CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS
DEL INSTITUTO POLITECNICO NACIONAL

DEPARTAMENTO DE INGENIERIA ELECTRICA
SECCION DE COMPUTACION

"SISTEMA DISTRIBUIDO CON MICROCONTROLADORES DE LA LINEA 8051"

Tesis que presenta el Ing. Alejandro Tinoco Alvarado para obtener el grado de MAESTRO EN CIENCIAS en la especialidad de INGENIERIA ELECTRICA con opción en COMPUTACION.

Trabajo dirigido por el Dr. Jan Janecek Hyan.

México D.F.
Septiembre 1990

Becario de
COSNET

A mis padres:

Jorge Tinoco y María Luisa Alvarado

A mis hermanos:

María Luisa,
Daniel,
Luis Antonio,
Susana Estela,
Benjamín y
Beatriz Graciela.

AGRADECIMIENTOS.

Estoy agradecido con mis compañeros y amigos cuyo ánimo y ayuda profesional convirtieron la realización de este trabajo en una tarea muy agradable. En particular deseo agradecer profundamente al Dr. Jan Janecek Hyan por sus enseñazas, consejos y su valiosa amistad, a Luis Rodolfo Rosado, Andrés Vega García, Luis Martin Rojas, Marcos Gómez Martinez y Octavio Juárez por sus aportaciones y sugerencias, asi como a Rosario Zamudio por su asistencia en la elaboración del texto.

También agradezco al Dr. Juan Manuel Ibarra Zannatha, al Dr. Armando Maldonado Talamantes, y al M. en C. José Oscar Olmedo Aguirre por sus valiosos comentarios y sugerencias.

Finalmente, me gustaría dar las gracias a la Sección de Computación del Departamento de Ingeniería Eléctrica de este Centro, por las facilidades ofrecidas durante el desarrollo de este trabajo.

I N D I C E .

Página:

OBJETIVO.

El presente trabajo de tesis tiene como objetivo diseñar y construir un sistema distribuido con microcontroladores de la línea 8051, el cual está compuesto de las siguientes partes:

- Un conjunto de tarjetas controladoras (módulos), interconectadas por una línea serie compartida.

- Un conjunto de programas que permiten: cargar programas de aplicación en cada uno de los módulos del sistema, ponerlos en ejecución y comunicarlos entre si.

- Un pequeño ejemplo de aplicación que muestre el uso y funcionamiento del sistema distribuido.

Este sistema está orientado a las aplicaciones del tipo control industrial, instrumentación y/o control de procesos. Básicamente, este sistema será una herramienta para el desarrollo de los programas de aquellas personas que trabajan en el control distribuido.

Se pretende además, que el sistema sea sencillo, económico y que su empleo resulte atractivo; aunque, debido a su sencillez, podrán existir aplicaciones donde el sistema no pueda ser usado.

PREFACIO.

Actualmente en la industria como en otros campos, el uso de las computadoras como medio para automatizar los procesos de fabricación, ha venido incrementándose cada vez más, haciendo el trabajo del hombre más fácil y productivo. Para esto se han desarrollado gran variedad de computadoras especializadas en el control industrial e instrumentación, recibiendo el nombre de "controladores".

Gracias al avance de la tecnologia, el costo de los "controladores" se ha reducido bastante, de tal forma que podemos colocarlos en cada etapa del proceso que deseamos controlar, interconectandolos entre si, formando lo que actualmente se conoce como: "sistemas de control distribuido".

En el presente trabajo se propone un sistema distribuido que será capaz de realizar tareas del tipo control industrial, el cuál podrá ser utilizado en una gran variedad de aplicaciones que van desde controlar una pantalla de desplegados, hasta controlar todo un proceso de fabricación, dependiendo esto, de las necesidades y la creatividad del usuario.

Para el estudio de nuestro sistema distribuido, se dividió el texto en 9 capitulos que tratan los siguientes temas:

El primer capitulo es introductorio y en él se proporcionan los conceptos fundamentales sobre microcontroladores y sistemas distribuidos.

El segundo capítulo da una vista general sobre el empleo de sistemas distribuidos en el control de procesos.

El tercer capítulo expone la arquitectura del sistema distribuido realizado.

El cuarto capítulo muestra la arquitectura BITBUS de Intel, de la cual se toman las bases para el desarrollo del sistema distribuido.

El quinto capítulo describe el diseño de las tarjetas controladoras.

En el sexto capítulo se define y establece el protocolo de comunicación utilizado.

El séptimo capítulo muestra y explica los programas de soporte desarrollados para la operación del sistema.

El octavo capítulo contiene un ejemplo de aplicación que muestra el funcionamiento del sistema.

Y por último, el noveno capitulo contiene las principales características del sistema distribuido.

NOTA:

Durante el desarrollo de este trabajo, todos los términos originales en idioma Inglés, se respetan y se escriben entre comillas para lograr mayor claridad en el texto.

La notación [#] indica la referencia bibliográfica.

C A P I T U L O 1

CAPITULO 1 - INTRODUCCION.

Los microprocesadores hoy en día, son verdaderamente una herramienta que facilita el diseño de equipo electrónico. Esta nueva dimensión literalmente expande el horizonte de aplicaciones más allá de la imaginación. No obstante, para aprovechar toda la potencialidad de los microprocesadores, uno debe entender los aspectos tanto de circuitería ("HARDWARE") como de programación ("SOFTWARE") de estas máquinas. Diferente a cualquier otra tecnología de computadoras, los microprocesadores están al alcance de todos por su bajo costo; pero desafortunadamente, existe un costo oculto en el desarrollo de "SOFTWARE".

Los microprocesadores los podemos encontrar en todas partes, en nuestra muñeca (reloj de pulso), en la cocina, en el sintonizador de un televisor, etc. En inumerables situaciones, hoy en día, disfrutamos de la potencialidad, flexibilidad y conveniencia de los microprocesadores. A continuación tenemos sólo algunas de las recientes aplicaciones de los microprocesadores:

Hornos de microondas.	Radios.
Relojes digitales.	Modems.
Osciloscopios inteligentes.	Teléfonos.
Terminales inteligentes.	Calculadoras.
Pequeñas computadoras.	Rastreadores de CB.
Controladores de procesos.	Juegos de TV.

La lista de aplicaciones ilustra dos puntos importantes: primero, las microcomputadoras (microprocesadores con otros dispositivos formando un sistema) no están reemplazando solo a las minicomputadoras en algunas aplicaciones, sino que además están creando muchas nuevas áreas de mercado. Segundo, los microprocesadores son verdaderamente una herramienta muy versátil para el diseño.

El bajo costo, tamaño reducido, y el bajo consumo de potencia de los microprocesadores, ha convencido a los diseñadores para utilizarlos en una amplia gama de aplicaciones. Además, la tendencia entre los usuarios a utilizar las microcomputadoras para el control local en vez de emplear grandes computadoras centrales controlando localidades remotas, ha venido incrementándose.

Los microprocesadores han venido reemplazando cada vez mas, grandes mini o maxi computadoras centrales previamente usadas en aplicaciones de control de procesos.

El uso de microprocesadores como microcontroladores locales en muchas de las etapas de un proceso tiene varias ventajas. Se eliminan kilómetros de cableado caro, los costos bajan, y la comunicación entre operadores locales y remotos se reduce, asi también se reduce la posibilidad de errores de comunicación.

En aplicaciones de control de procesos, típicamente encontramos cientos de puntos de medición, tales como: temperatura, presión, velocidad de líneas, y otros datos de procesos. En operación, numerosos circuitos analógicos pueden seleccionarse automáticamente y ser digitalizados bajo el control de un microprocesador. El microprocesador realiza los cálculos y operaciones de procesamiento a velocidades relámpago, conmutando entradas y digitalizando los datos analógicos.

Para que un microprocesador tenga todas estas capacidades es necesario dotarlo de algunos dispositivos extras, como circuitos de reloj, memorias, puertos de entrada/salida, etc. y todo este conjunto recibe el nombre de: **"microcomputador"**.

Intel ha desarrollado una familia de microcomputadores basados en el microprocesador (CPU) 8051 desarrollado también por Intel, para aplicaciones sofisticadas de tiempo real, tales como: instrumentación, control industrial y periféricos inteligentes de computadoras.

Estos microcomputadores se encuentran dentro de un solo empaquetado ("CHIP"), alojando en su interior una CPU 8051, puertos de entrada/salida, un puerto de comunicación serie, contadores, etc. superando a los microprocesadores en tamaño y espacio.

Agregando memorias externas y otros dispositivos a estos microcontroladores dentro de una tarjeta, obtendremos un **módulo controlador** con un tamaño bastante reducido y con toda la potencialidad de una computadora.

Ahora bien, si interconectamos varios módulos entre si, con el fin de intercambiar información, tendremos una red de controladores, que junto con un conjunto de programas ("SOFTWARE") que controlan el uso de esta red, obtendremos un "sistema distribuido".

Un **sistema distribuido** es aquel en el cual, varios procesadores autónomos y varias unidades de almacenamiento de datos, interactuan en forma cooperativa para realizar una meta común. Los procesos coordinan sus actividades e intercambian información por medio de una red de comunicación. [1]

Los atributos requeridos por un sistema distribuido son los siguientes:

1. Número arbitrario de sistemas y procesos de aplicación (recursos lógicos).

Debe ser posible soportar estos procesos y aceptar el incremento o decremento arbitrario de su número.

2. Arquitectura física modular (recursos físicos).

El sistema puede consistir de multiples elementos de procesamiento interconectados. Estos pueden estar físicamente distribuidos y pueden variar también en número durante el tiempo de vida del sistema.

3. Comunicación por envio de mensajes usando un sistema de comunicaciones compartido.

Este proporciona la transferencia de mensajes en una forma cooperativa, pudiendo soportar protocolos de más alto nivel.

4. Algún control sobre el sistema.

Este es necesario para integrar los elementos de procesamiento dentro de un sólo sistema coherente. Se da una política común para indicar como es usado el sistema, las facilidades dadas, el control de acceso, la protección, etc.

Los atributos anteriores fueron definidos por: Morris Sloman y Jeff Kramer en su libro: "Sistemas distribuidos y redes de computadoras".

Los beneficios potenciales de la distribución son muchos. Los principales objetivos de un sistema distribuido son descritos a continuación:

Reducir el costo del sistema.

Con la llegada de las micro- y minicomputadoras de bajo costo y con una alta relación: funcionalidad-costo, se ha impulsado el empleo del procesamiento distribuido. Sin embargo, el costo de los dispositivos periféricos no ha declinado tan dramáticamente. El compartir recursos caros como: impresoras laser, bases de datos, procesadores de propósito especial, etc. es otro de los mayores objetivos para los sistemas distribuidos.

En aplicaciones de control de procesos, en particular, puede ser un gran ahorro en el costo del cableado el colocar unidades de procesamiento junto a la planta, reemplazando cableado paralelo con cableado serie usando cable coaxial o un par torcido. En algunas aplicaciones es posible usar inteligencia local para reducir el costo de comunicación, reduciendo el volumen del tráfico.

Modularidad y simplicidad del "SOFTWARE".

Los sistemas distribuidos deben ser construidos en una forma muy modular, donde cada componente proporcione interfaces bien definidas o de servicios al resto del sistema. La modularidad nos conduce a simplificar el diseño del sistema, instalación y mantenimiento.

La reducción en el costo del "HARDWARE" ha hecho factible sub-utilizar un procesador y dedicarlo a tareas o funciones particulares, simplificando su programación.

Flexibilidad y extensibilidad.

Esta disciplina ha agregado la ventaja de facilitar la modificación o extensión de un sistema para adaptarlo a un ambiente cambiante, sin destruir su operación. La flexibilidad del "HARDWARE" permite reemplazar o agregar elementos de procesamiento y conexiones de comunicación; flexibilidad similar puede ser dada por el "SOFTWARE" sin afectar a los demás componentes en el sistema.

Usando protocolos de comunicación estandarizados, es posible incorporar equipos de diferentes fabricantes a la red (sistema distribuido), reduciendo la confianza sobre algún fabricante.

En fin, el empleo de un "sistema distribuido" para aplicaciones de control de procesos, es una de las mejores opciones que se tienen hoy en día.

C A P I T U L O 2

CAPITULO 2 - CONTROL DISTRIBUIDO.

Una de las aplicaciones más frecuentes de las computadoras es la de "controlar" dispositivos externos conectados a ellas, como: leer un teclado, encender una lampara, medir temperatura, encender un motor, ... , hasta controlar todo un proceso de fabricación, tan complicado como lo podamos imaginar.

Enfocándonos hacia el uso de las computadoras para el control industrial, nos encontramos con problemas de localización física de los dispositivos a controlar, los cuales se pueden encontrar a distancias bastante grandes para poder ser controlados por una computadora central; algunos de estos requerirán de mayor atención de la computadora que otros, y si el tiempo de respuesta es crítico, el sistema fallará.

La solución a este problema es colocar (distribuir) computadoras en áreas donde más se necesiten, e interconectar estas a través de una red de comunicación para la transferencia de datos (ver fig 2.1). De esta forma, cada computadora tiene menos dificultad en responder al medio ambiente externo porque tienen menos información que manejar, haciendo su programación más simple.

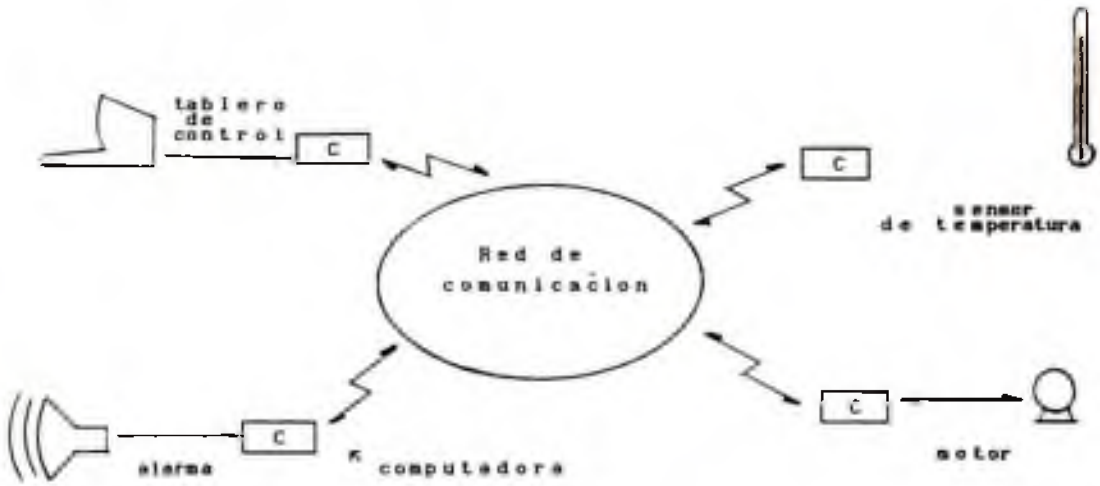


Figura 2.1 - Estructura general de un sistema de control distribuido.

Problemas en control.

Tipicamente, en los sistemas de control, se desea que una computadora anfitrión se comunique con microcontroladores en localidades remotas.

Esquemas de arbitraje del canal de comunicación ("BUS") como detección de portadora y múltiple acceso con detección de colisión, son muy usados para redes locales en muchas aplicaciones, pero estos no satisfacen algunas de las necesidades que se presentan en los ambientes industriales.

Los gastos involucrados en la conección del "BUS", pueden eliminarse a través de una relación Maestro-Esclavo entre la computadora anfitrión y los microcontroladores remotos; el maestro simplemente revisa ("POLLS") cada uno de los microcontroladores.

Hay algunas aplicaciones que resuelve el modelo "Maestro-Esclavo", como: adquisición de datos, automatización de fábricas y control de procesos.

C A P I T U L O 3

CAPITULO 3 - ARQUITECTURA DEL SISTEMA DISTRIBUIDO.

El sistema distribuido desarrollado en el presente trabajo, contiene un conjunto de controladores (módulos), los cuales están interconectados a través de un canal común de multiple acceso, con el fin de intercambiar información y poder coordinar sus actividades sobre el proceso que se desea controlar.

El uso de un canal común para la comunicación, facilita la instalación del sistema en el medio de operación, y hace más simple su manejo.

El sistema está formado por un módulo "maestro" llamado: "nodo maestro", y de un conjunto de módulos "esclavos" llamados también: "nodos esclavos", como se observa en la figura 3.1 (siguiente página). El nodo maestro se encarga de coordinar las comunicaciones entre los nodos esclavos, de tal forma que estos últimos se despreocupan de esta tarea.

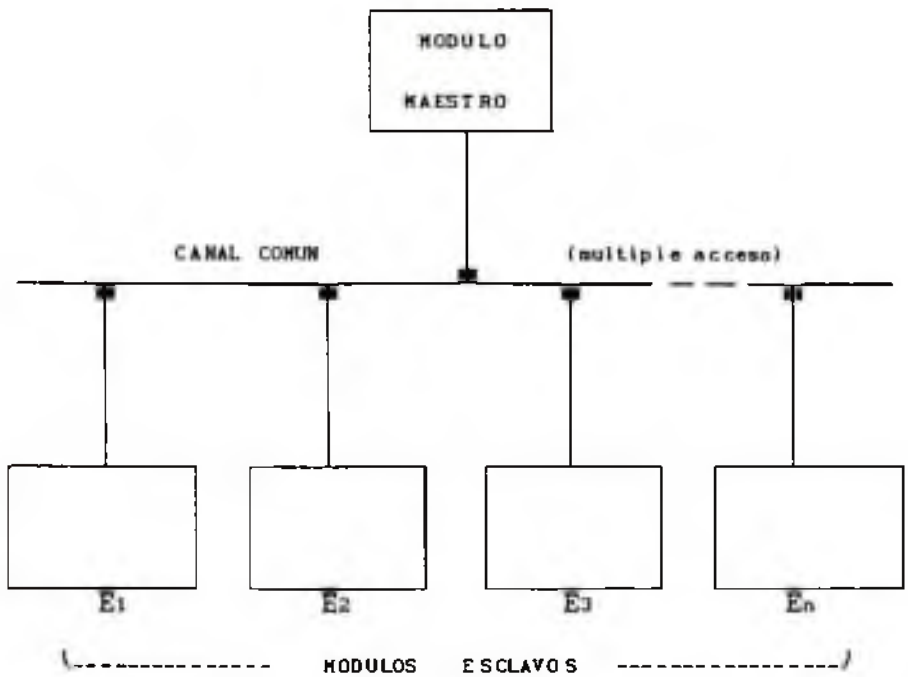


Figura 3.1 - Diagrama a bloques del sistema distribuido.

Cada uno de los nodos que componen nuestro sistema, tanto el nodo maestro como los nodos esclavos, son unas pequeñas computadoras mínimas basadas en el procesador: "8051" de Intel, el cual está orientado a aplicaciones del tipo control industrial, de aquí su nombre de "controladores". En el capítulo 5 del presente trabajo, veremos las características y diseño de estos controladores.

Para lograr que el sistema trabaje en forma distribuida, se requiere de una compartición de información entre aquellos nodos que realizan una tarea común, para esto, utilizan el canal de comunicación para enviar y recibir sus mensajes. Pero si dos nodos intentan transmitir su información a través del canal al mismo tiempo, estos se inteferirán y los mensajes se perderán, a este problema le llamamos "colisión" entre los mensajes.

Nuestro sistema distribuido está organizado de tal forma que las colisiones no son posibles, esto gracias a la coordinación de las operaciones de los transmisores.

Existen sistemas que utilizan: "TOKENS", "POLLING", técnicas de reservación y otras, para coordinar las comunicaciones entre los nodos que los conforman, evitando colisiones. De estas técnicas elegimos para nuestro sistema, el "POLLING", por su simplicidad.

Sistemas con "POLLING".

En estos sistemas, a los nodos se les permite usar el canal común en turnos. Mientras el canal está siendo usado por algún nodo, a todos los demás nodos se les prohibe introducir sus paquetes al canal. Después de que el nodo que está transmitiendo ha completado su ciclo de operación, se le permite al siguiente nodo usar el canal, mientras los restantes deben permanecer ociosos. [2]

Estos sistemas pueden trabajar tanto en forma síncrona como en asíncrona.

Descripción del sistema.

En este trabajo se consideró un sistema con "POLLING" asíncrono centralizado con un nodo maestro y los restantes como nodos esclavos.

El nodo maestro envía información de permiso, ξ_{per} ("POLLING") a cada uno de los nodos.

ξ_{per} puede tomar valores: ξ_m , $m = 1,2,...,M$,
donde: ξ_m significa que al m-ésimo nodo se le ha permitido iniciar la transmisión de paquetes.

Después que el m-ésimo nodo ha completado su transmisión, envía al nodo maestro información de fin de mensaje: ξ_{fin}, entonces el nodo maestro al recibirlo, envía la información de permiso al siguiente nodo: ξ_{m+1}. El proceso continua cíclicamente enviando ξ_1 después de ξ_K.

La regla de "POLLING" centralizado asíncrono (ACP) se define más precisamente por el conjunto de subreglas siguientes:

1 - Un páquete listo para transmitir dentro del m esimo nodo es tomado por éste e introducido a su "BUFFER".

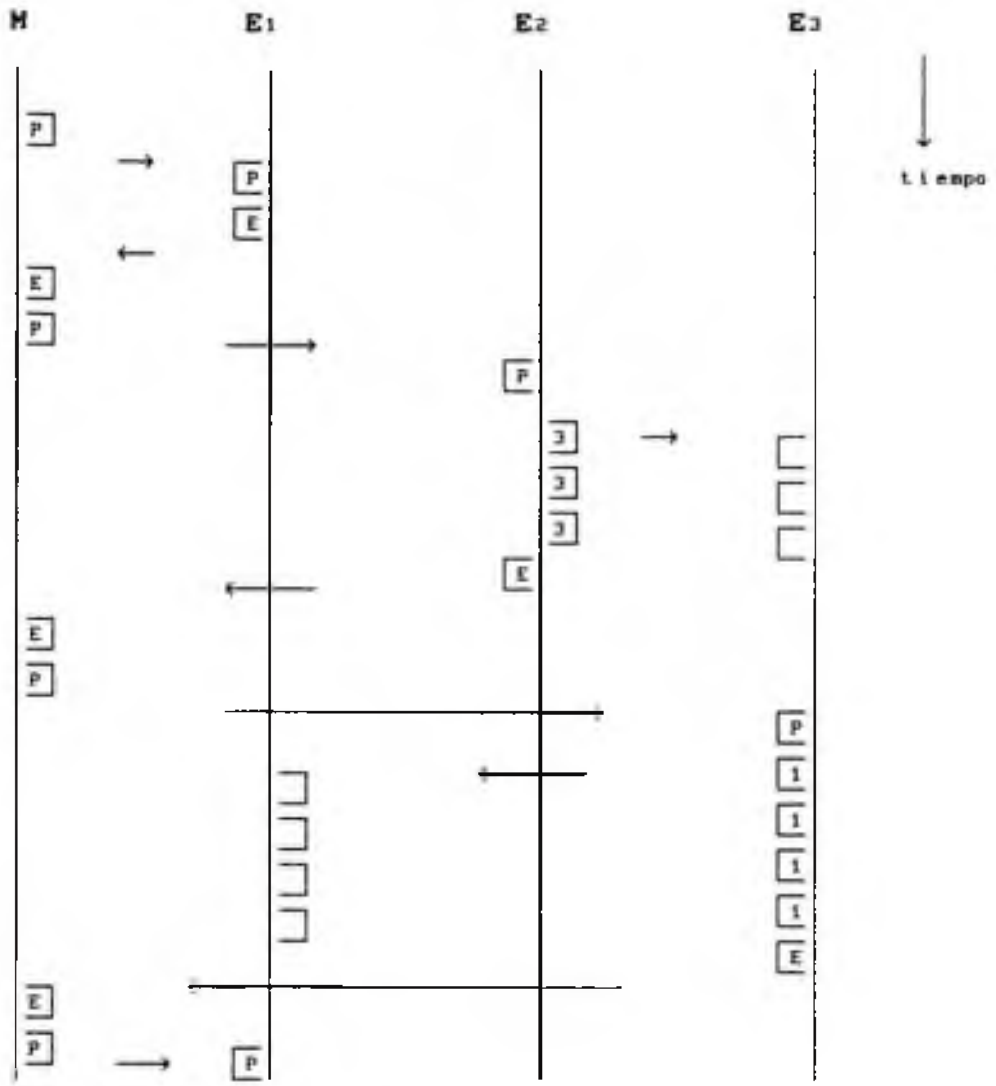
2 - Cuando $\xi_{per} = \xi_m$, el m-ésimo nodo remueve el paquete de su "BUFFER" y lo envía por el canal común.

3 - Al terminar de transmitir el m-ésimo nodo su paquete, envía la señal: ξ_{fin} al nodo maestro.

4 - Cuando el nodo maestro recibe la señal ξ_{fin}, envía la señal ξ_{per} al siguiente nodo.

5 - Si $\xi_{per} = \xi_m$ y el "BUFFER" en el m-ésimo nodo se encuentra vacío, ξ_{fin} se envía inmediatamente.

En la figura siguiente se tiene un diagrama que muestra un ejemplo del funcionamiento del sistema con "POLLING".



[P] - Señal de permiso ("POLLING").

[E] - Señal de fin de mensaje.

[n] - Paquete enviado al nodo n.

M - Nodo maestro.

En - Nodo esclavo n.

Figura 3.2 - Funcionamiento de un sistema con "POLLING".

Hasta ahora, tenemos definido de que elementos va a estar formado nuestro "sistema distribuido", y en que forma se va a realizar el intercambio de información entre estos. Pero es necesario resolver algunos otros problemas como:

- ¿ Qué norma de comunicación se va a utilizar ?
- ¿ De qué estan compuestos cada uno de los elementos ?
- ¿ Qué capacidades tienen estos elementos ?
- ¿ Qué protocolo de comunicación se va a utilizar ?

Resolveremos estos, como vayamos avanzando en el texto.

C A P I T U L O 4

CAPITULO 4 - ARQUITECTURA BITBUS.

Los microcontroladores han modernizado los sistemas mecánicos y eléctricos, reemplazando relevadores, ruedas y ganchos en aplicaciones de control de procesos en fabricación automatizada. Al incorporar tan solo un microcontrolador, el costo total del sistema de la mayoría de las aplicaciones orientadas al control, puede reducirse, agregando funcionalidad y reducción del espacio físico ocupado.

Sin embargo, si no se cuenta con un estándar de comunicación industrial, la mayoria de los microcontroladores de aplicación carecerán de un medio para conectarse a otros microcontroladores o a equipo de control.

A través de los avances en la tecnología de silicio, los sistemas de control han evolucionado. En un principio los sistemas funcionaban por secuencias de relevadores y simples alarmas indicadoras. La PDP-8 de Digital Equipment Corp. proporcionó la primera herramienta de uso común para coordinar varios controles en tiempo real. Los microprocesadores dieron un camino más simple para la reducción del costo de las computadoras, pero no cambiaron las arquitecturas de los sistemas hasta que los BUSes estandar fueron populares.

Las arquitecturas industriales estándar de BUS, como MULTIBUS y STD-BUS, han permitido a los diseñadores dividir las tareas de control entre varios procesadores, tomando las ventajas de módulos de varios fabricantes.

Los microprocesadores han pavimentado el camino para el próximo paso: "el control distribuido". Hay muchos caminos para distribuir el control: estructuras de BUS paralelo, conjuntos de señales de control sobre líneas individuales, enlaces de comunicación serie, o tecnologías habituales.

Cada una de las soluciones anteriores, requiere de un esfuerzo considerable en el diseño y frecuentemente se tienen limitantes en la interconexión.

En Febrero de 1984, Intel introdujo una nueva arquitectura de comunicación por "BUS", llamada: "BITBUS", enfocada a ampliar las aplicaciones basadas en microcontrolador. [4]

La interconexión BITBUS es un "BUS" serie optimizado para la transferencia de mensajes de control a altas velocidades en un sistema jerárquico. Se especifican las interfaces de alto nivel para dar una interconexión serie de alta funcionalidad transparente a las aplicaciones del programador. Estas interfaces incluyen: una estructura de mensajes y un protocolo (ver tabla 4.1), para ambientes multi-tareas, y un conjunto de comandos de alto nivel para accesos remotos de entrada/salida y para control de tareas de aplicación.

Tabla 4.1 - Interfaces del estandar BITBUS.

Interfaz:	Especificación:
Eléctrica.	RS-485.
Cable.	Par torcido o cable plano 10 cond.
Conector BACK-PLANE.	DIN estd. de 64-patas.
Conector terminal.	3M #3446-1302 hembra.
Control de enlace de datos.	Control de enlace de datos síncrono (SDLC).
Velocidad de transferencia de datos.	62.5K baud, 375K baud y 2.4M baud.
Formato de mensajes.	comando/respuesta/estado.

Especificaciones BITBUS.

BITBUS conecta un controlador maestro a un conjunto de controladores locales o remotos, con una configuración multiterminal ("MULTIDROP") conectando a los controladores a un "BUS" común.

Utiliza la interfaz RS-485 como enlace físico entre controladores, proporcionando características como inmunidad al ruido en ambientes industriales, y un cableado de bajo costo.

BITBUS es una variación aceptada de la interfaz común RS-422, permitiendo segmentos de cable más largos y más conexiones "MULTIDROP".

Soporta hasta 250 nodos, tres velocidades máximas de transferencia de datos y tres distancias máximas, como se muestra en la tabla 4.2.

Tabla 4.2 - Especificación vel/dist/#-nodos máx. BITBUS.

Velocidad:	Distancia:	No.de nodos:	Cable:	N seg.
2.4 Mbaud	30 metros	28 nodos	4-alambres	1
375 Kbaud	300 metros	56 nodos	par torcido	2
62.5 Kbaud	1200 metros	250 nodos	par torcido	9

28 nodos máximo por segmento.

Protocolo de transferencia y formato de los mensajes.

BITBUS se basa en el estandar "control de enlace de datos síncrono" (SDLC) de IBM, usado por muchos vendedores. SDLC es un protocolo de control para enlace de datos orientado a bit que define una estructura específica para cada tipo de intercambio de datos y de control.

Formato de las tramas de SDLC.

Tipo de trama:	**Formato:**	**Función:**
Trama-I	F A C -d.u.-FCS F	Transf. de inf.
Trama-S	F A C FCS F	Contr. supervisor
Trama-U	F A C FCS F	Sincr. Tr/Rec

Donde:

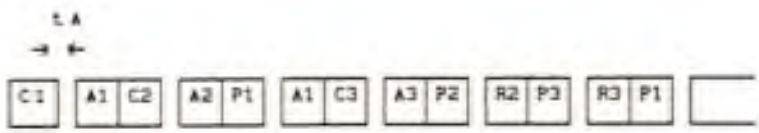
F - Bandera SDLC.
A - Dirección de estación esclavo.
C - Campo de control.
d.u.- Datos del usuario.
FCS - Secuencia de chequeo de trama.

Como se muestra en la tabla anterior, cada tipo de transmisión (Trama-I, Trama-S y Trama-U) es dividido en campos identificables. Cada campo contiene uno o más bytes de datos y/o información de control para realizar la función correspondiente.

Las transmisiones normales entre maestro y esclavo inician con un intercambio de Tramas-U para sincronizar los contadores de secuencia de tramas y otros controles. El intercambio de datos se realiza con la Trama-I y la Trama-S. La Trama-I contiene datos para el esclavo o una respuesta de un esclavo, las Tramas-S son usadas para rastrear un esclavo o para datos.

Los datos de usuario contenidos en las Tramas-I, conforman el formato de mensajes estandar BITBUS. Todos los mensajes contienen una cabecera de 5 bytes, que describen la dirección fuente y destino, junto con otra información de estado y control.

Los controladores BITBUS requieren que cada mensaje sea contestado y no solo reconocido. La figura siguiente muestra la conversación entre maestro y esclavo BITBUS.



Donde:

C1 = Cn

Mensaje de comando enviado por el maestro para identificar al esclavo.

Al = An

Reconocimiento hecho por el esclavo identificado al mensaje del maestro.

Am

Reconocimiento hecho por el maestro para un mensaje de un esclavo.

Rl = Rn

Mensaje de respuesta enviado por el esclavo identificado al maestro. El maestro reconoce esta respuesta con el rastreo o comando siguiente enviado al esclavo identificado.

Pl = Pn

Rastreo (POLL) enviado por el maestro al esclavo identificado preguntando por un mensaje de respuesta.

tA

Tiempo tomado para reconocer un mensaje o rastreo del maestro.

La arquitectura BITBUS, proporciona un método de conexión de controladores en aplicaciones de control distribuido. Usando una interconexión estandar es posible interconectar controladores que antes se encontraban aislados, con otras partes de un sistema de control distribuido, sin importar si estas partes son de diferentes fabricantes.

De la arquitectura BITBUS, tomamos las bases para desarrollar nuestro sistema distribuido haciendo algunas modificaciones. Tomamos de éste, la interfaz eléctrica. El formato de las tramas asi como el protocolo de comunicación son parecidos a los definidos en nuestro sistema, como se verá en el capítulo 6. En nuestro sistema utilizamos comunicación asíncrona, un poco diferente a la comunicación síncrona utilizada por BITBUS (SDLC), pero más facil de utilizar.

La idea principal del presente trabajo fué la de simplificar la arquitectura de un sistema de control distribuido, asi como su manejo.

C A P I T U L O 5

CAPITULO 5 - DISEÑO DEL CONTROLADOR 8051.

El sistema distribuido está formado por un conjunto de controladores semejantes entre si, pero con diferentes tareas en la aplicación, es decir, la circuiteria ("HARDWARE") es la misma para cada nodo del sistema incluyendo al nodo maestro, pero la programación ("SOFTWARE") es distinta. En este capitulo enfocaremos nuestro estudio hasia el "HARDWARE" de nuestro sistema distribuido.

El diseño del controlador "8051" está dividido en 3 partes:

a) Especificaciones.
b) Criterios de diseño.
c) Diseño final.

a) Especificaciones.

Para el diseño del módulo controlador "8051" requerimos primeramente observar las especificaciones que éste debe cumplir para satisfacer nuestras necesidades, y estas son:

1- El módulo controlador deberá estar orientado a las aplicaciones de control industrial.

2- Que disponga de memorias para almacenar un programa monitor residente y almacenar los programas de aplicación del usuario.

3- Que cuente con de un puerto de comunicación que permita intercomunicarlo, tanto con otros módulos semejantes a éste, como con otros equipos.

4- Que cuente además con un puerto de entrada/salida para interconectar los dispositivos a controlar (mundo exterior).

5- Que su tamaño sea pequeño y su costo reducido.

6- Que permita enviar voltaje de alimentación a cada uno de los módulos conectados a la red (sistema) a través del mismo canal o "BUS" de comunicación.

7- Que todos los controladores que forman la red sean iguales sin importar su uso o aplicación.

b) Criterios de diseño.

Para obtener un módulo controlador de tamaño pequeño, buscamos los circuitos en el mercado que en un solo empaquetado ("CHIP") contengan el mayor número de circuitos necesarios para nuestro sistema, haciendolo pequeño y económico.

Consultando manuales de "Intel" encontramos la línea de microcontroladores "8051", la cual es una familia compuesta por 3 "CHIPS": el 8031, el 8051 y el 8751, los cuales son unos microcomputadores diseñados especialmente para aplicaciones de control, de aqui su nombre "microcontroladores", los cuales trabajan a una velocidad bastante aceptable (1 useg./instr. en promedio).

El término genérico "8051" es usado para referir colectivamente al 8031, al 8051 y al 8751.

Estos microcontroladores de 8 bits, cuentan internamente con una CPU "8051", puertos de entrada/salida, un puerto de comunicación serie, contadores, temporizadores, circuitos de reloj, etc. (ver apéndice A), y tales características son las que andamos buscando, eligiendo así uno de estos "CHIP".

Las diferencias entre estos "CHIPS" es la siguiente:

El 8031 no tiene memoria interna de programa.
El 8051 tiene memoria interna de programa de sólo lectura.
El 8751 tiene memoria interna de programa de sólo lectura borrable (EPROM).

Elegimos entre estos al "8031" por tratarse nuestro proyecto de un sistema que se está desarrollando, al cual continuamente se le modifica el programa monitor residente, lo cual no podriamos lograr utilizando un "CHIP" 8051, y para utilizar un 8751 necesitaríamos de un grabador especializado en este tipo de circuitos. En cambio utilizando el 8031, sólo necesitamos agregar algunas memorias de acuerdo a nuestras especificaciones y de los medios de grabado a nuestro alcanze. Aunque gracias al "PIN" 31 del 8031 (ver apéndice A) podemos emplear cualquiera de los tres "CHIPS" sin hacer modificaciones al diseño.

El "8051" maneja dos áreas de memoria diferentes: área de programas y área de datos.

Como área de memoria de programa utilizamos una "EPROM" **2764** de 8 K bytes, tamaño suficiente para almacenar nuestro programa monitor residente que se encargará de inicializar el controlador, cargar y ejecutar los programas de aplicación.

Como área de memoria de datos utilizamos una "RAM" **6264** de 8 Kbytes, también suficiente para nuestras necesidades, ya que este controlador no está destinado al manejo de grandes cantidades de información.

Para poder conectar estas memorias al microcontrolador "8051" necesitamos desmultiplexar la parte baja del "BUS" de direcciones, que se encuentra multiplexado con el "BUS" de datos dentro del puerto 0, para esto utilizamos una compuerta "LATCH" **74LS373** la cual puede atrapar 8 líneas de entrada, siendo estas

líneas los 8 bit de la parte baja del canal de direcciones (16 bits para direccionar hasta 65,536 localidades), y obteniendo los 8 bits restantes directamente del puerto 2 del "8051".

Ahora bien, para poder ejecutar programas de aplicación almacenados dentro del área de memoria de datos (memoria "RAM") necesitamos hacerle creer al procesador que esta seleccionando área de programa, mientras que realmente estará seleccionando área de datos.

Lo anterior se resuelve colocando un pequeño circuito de selección que habilite la memoria "RAM" cuando se seleccionen localidades superiores a 8 Kbytes (2000H). Para el diseño de este circuito utilizamos la tabla de habilitación mostrada en la figura 5.1.

$\overline{PSEND}$	$\overline{RD}$	A13	$\overline{OE}$ ("RAM") Se habilita como:
0	1	1	Area de programa
1	0	0	Area de datos
otros	casos		Deshabilitada

Donde: $\overline{PSEN}$ - Se pone en 0 cuando se realiza lectura de código.

$\overline{RD}$ - Se pone en 0 cuando se realiza lectura de datos.

A13 - Se pone en 1 cuande se accesa una localidad superior a 8 Kbytes.

Figura 5.1 - Tabla de habilitación del área de memoria de datos "RAM" como área de programa.

El circuito de selección se realizó utilizando cuatro compuertas NOR contenidas dentro de un solo "CHIP", el **74LS02** como se podrá observar en el diagrama del controlador mostrado en el apéndice A.

Finalmente, para poder interconectar y comunicar el conjunto de controladores de que está compuesto el sistema, utilizamos el puerto de comunicación serie con que cuenta el microcontrolador "8051", el cual puede trabajar tanto en forma sincrona como en forma asincrona, eligiendo entre las dos a la "comunicación serie asíncrona" ("USART") por ser simple, por la facilidad que da para interconectar otros equipos, y por ser fácil de manejar.

Para apegarnos a la norma eléctrica RS-485 recomendada por BITBUS de Intel para sistemas de control industrial con microcontroladores, consultamos los manuales buscando "DRIVERS" y "RECEIVERS" que manejaran esta norma, y poder así conectar el "USART" del "8051" al canal de comunicación que trabaja bajo esta norma, y encontramos el "TRANSCEIVER" 75176, el cual internamente contiene a ambos circuitos, con una salida balanceada que conectada a un "BUS" del tipo: Par Torcido, evita que existan interferencias por ruido; y que a través de dos pines de este circuito podemos habilitar la transmisión o la recepción de información, obteniendo un sistema de comunicación del tipo: "HALF DUPLEX", donde se permite transmitir o recibir información, pero no ambas (ver apéndice A).

c) Diseño final.

Una vez que tenemos definidos los elementos de que va a estar compuesto nuestro sistema, sólo resta montarlos e interconectarlos entre si siguiendo las especificaciones de los manuales y agregando los elementos necesarios para su funcionamiento sugeridos por el fabricante, como son: el cristal de 12 MHz, el circuito de "RESET", la fuente de alimentación, etc.

Para alimentar a aquellos módulos controladores que se encuentran en localidades físicas donde el proporcionarles alimentación resulta complicado, se proporsiona un voltaje de alimentación a través del mismo "BUS". Esto se logra agregando un par de alambres a la linea BITBUS (RS485) sobre los cuales se envia un voltaje de +12V y Tierra, como se muestra en la figura siguiente.

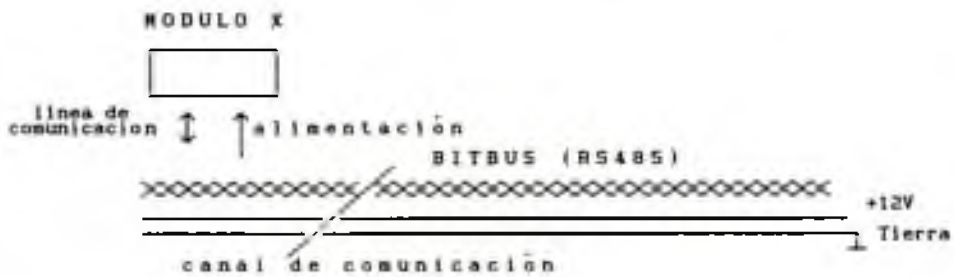


Figura 5.2 - Envio de alimentación a través del canal de comunicación.

El voltaje enviado es regulado a +5V dentro de cada módulo controlador utilizando un circuito integrado 7805, el cual es un regulador de voltaje a +5V, esto siempre y cuando se aplique a su entrada un voltaje dentro de +7V y +35V.

El diagrama y la construcción de estos módulos controladores se presenta en los apéndices A y C.

C A P I T U L O 6

CAPITULO 6 - PROTOCOLO DE COMUNICACION.

Como mencionamos en un principio, la necesidad de comunicar computadoras de diferentes fabricantes ha conducido al desarrollo de estandares de comunicación para sistemas distribuidos. El modelo ISO (Open System Interconnection Reference) fué definido como un marco de referencia para el desarrollo de estándares de comunicación. [1]

Una red de comunicación puede ser extremadamente compleja, y tanto, que ésta es organizada en una jerarquía de capas las cuales dividen el problema total en pequeñas piezas. Cada una agrega valores al servicio dado por el conjunto de capas menores, de tal forma que la capa más alta ofrece el conjunto de servicios necesarios para correr aplicaciones distribuidas.

El modelo de ISO define las 7 capas siguientes:

7 - Aplicación:
procesamiento de todas las aplicaciones especificas.

6 - Presentación:
responsable de la transformación y representación de la información.

5 - Sesión:
mantiene la asociación entre entidades de aplicación y realiza el control de diálogo.

4 - Transporte:
control de flujo y de error entre terminal-terminal.

3 - Red:
consideraciones de enrutamiento y conmutación.

2 - Enlace de datos:

control de flujo y error a través de un solo enlace de datos.

1 - Física:

transferencia de bit y señalización.

Un **protocolo** es el conjunto de reglas (semánticas y sintácticas) que gobiernan las comunicaciones entre entidades las cuales constituyen una capa en particular (Fig. 6.1). Esto es, el protocolo N define como una entidad en el nivel N en un sistema, intercambia información con su correspondiente pareja en otra estación, de acuerdo a los servicios provistos en el nivel N. [1]

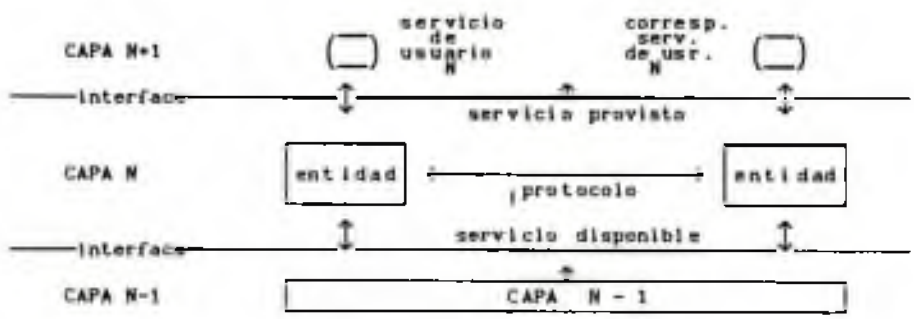


Figura 6.1 - Conceptos de capas.

Las reglas sintácticas del protocolo definen el formato de la información intercambiada. Por ejemplo: "el tercer byte en un mensaje representará el número de secuencia".

Las reglas semánticas del protocolo definen las operaciones a realizarse por el transmisor y el receptor, es decir, bajo que condiciones los datos deben ser retransmitidos, reconocidos, rechazados, etc.

El protocolo se diseñó en base a la facilidad que ofrece el microcontrolador "8051" de interrumpir a la CPU, cuando se recibe una trama de 11 bits por el puerto serie, en la cual, el bit 8 (noveno bit) está en "uno", realizando una rutina de servicio a esta interrupción (vector 0023H); esto se realiza programando el UART del "8051" en el modo 3.

Si se utiliza la primera trama para direccionar el mensaje transmitido con el 9o. bit puesto (indicador de dirección), se interrumpirá a todos los módulos controladores conectados a la red, los cuales en su rutina de servicio checarán si la dirección del mensaje recibido es la suya, aquel que sea el direccionado reconfigurará su puerto con el fin de continuar recibiendo las subsecuentes tramas del mensaje, mientras que los demás, continuan trabajando sin hacer caso de las tramas siguientes.

Reglas sintácticas del protocolo diseñado.

- La primera trama contiene la dirección del destino del mensaje, poniendo el noveno bit de esta trama en uno.

- La segunda trama contiene la dirección del nodo que envia el mensaje (dirección fuente), con el fin de que más tarde se devuelva el acuse de recibo o reconocimiento correspondiente.

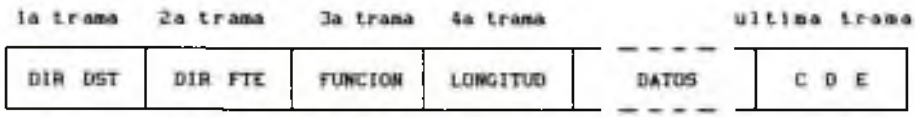
- La tercera trama contiene un byte de función. el cual indica el significado del mensaje, por ejemplo, si se trata de un rastreo ("POLLING"), si es un reconocimiento, si es un fin de mensaje, si es código de programa, o si son datos.

- La cuarta trama contiene la longitud de los datos que serán enviados después de esta trama.

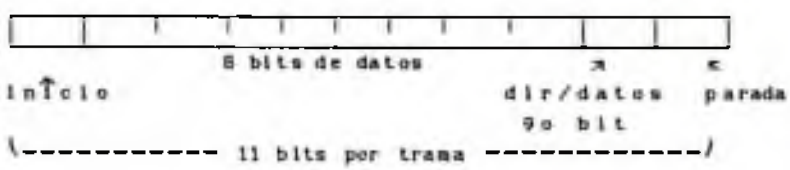
- Las tramas siguientes a la cuarta, contienen la información enviada, cuando la longitud es diferente de cero.

- La última trama contiene la suma de todos los bytes transmitidos en este mensaje, con el fin de checar si se recibió correctamente (Código de Detección de Error: Check-Sum).

Esquemáticamente el protocolo tiene el siguiente formato:



Y cada trama tiene el siguiente formato:



El procedimiento para la recepción-transmisión de las tramas de un mensaje es el siguiente:

1 - Esclavo: Configura el puerto serie para interrumpir a la CPU sólo cuando el noveno bit de la trama recibida sea uno. (SETB SM2)

2 - Maestro: Transmite trama conteniendo la dirección en los primeros 8 bits de datos y pone el noveno bit de datos en uno (SETB TB8, indicando que es trama de direccionamiento).

3 - Esclavo: El puerto serie interrumpe a la CPU de todos los computadores en la red cuando la trama de direccionamiento se ha recibido. El programa que sirve la interrupción compara la dirección recibida con su dirección. El esclavo que fué direccionado reconfigura su puerto serie para interrumpir a la CPU en las subsecuentes transmisiones. (CLR SM2)

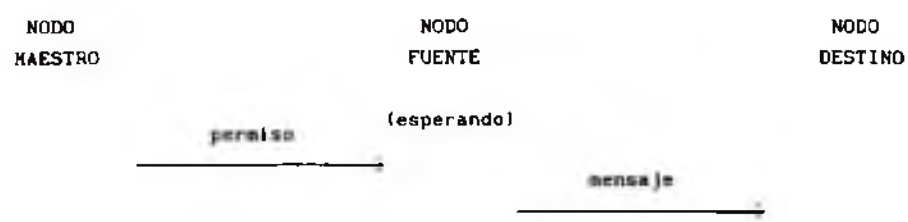
4 - Maestro: Transmite tramas de: dirección fuente, función, longitud, datos y código de detección de error (éstas son sólo aceptadas por el esclavo previamente direccionado).

5 - Esclavo: Al recibir cada trama el esclavo direccionado, va sumando cada byte recibido, y al final checa su resultado con el código de error recibido, asi sabrá si recibió correctamente el mensaje.

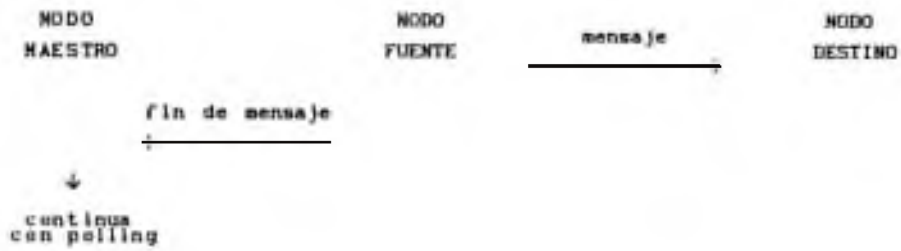
Reglas semánticas del protocolo diseñado.

Las reglas semánticas del protocolo se diseñaron en base al método de "POLLING" definido en el capítulo 3, y son:

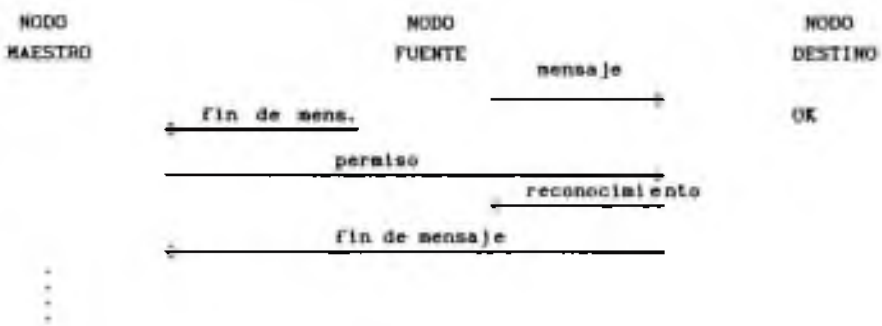
1 - Los mensajes podran ser enviados sólo cuando se reciba la señal de permiso ("POLLING") dada por el nodo maestro.



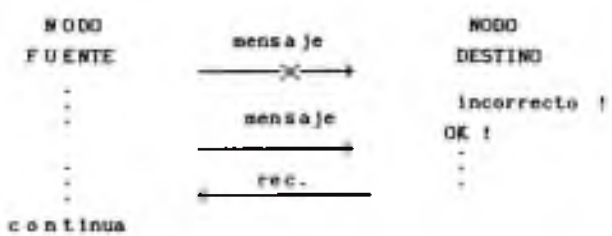
2 - Al final de cada mensaje transmitido, se envia la señal de fin de mensaje hasia el nodo maestro, para que éste continue su ciclo de rastreo.



3 - Cada mensaje enviado por algún nodo esclavo hacia otro nodo esclavo, requiere de un acuse de recibo o reconocimiento, por parte del nodo destinatario, el cual es enviado sólo cuando se recibe la señal de permiso.



4 - Si el mensaje recibido está incorrecto, no se envía reconocimiento, y se espera volver a recibir el mensaje, el cual se envía nuevamente al no recibir respuesta por parte del destinatario.



5 - Si el reconocimiento no se recibe dentro de un número determinado de permisos recibidos ("POLLINGS"), el nodo que se encontraba esperando la respuesta, sale de este estado a través de un registro de "TIME OUT" (tiempo terminado),pudiendo entonces tomar una decisión y continuar con su trabajo.

Este protocolo es bastante simple, pero para nuestro sistema distribuido, es suficiente para realizar la comunicación en forma aceptable, y en el futuro puede ser ampliado o mejorado.

Con esto, nuestro sistema distribuido alcanza a cubrir hasta la tercera capa de ISO: 1 - Física, 2 - Enlace de datos, y 3 - Red.

C A P I T U L O 7

CAPITULO 7 - PROGRAMACION DEL SISTEMA DISTRIBUIDO.

Hasta ahora sólo conocemos las herramientas físicas que forman nuestro sistema distribuido, y de algunos conceptos teóricos que definen como funcionará. Necesitamos entonces conocer el "SOFTWARE" (programas) que se encargará de explotar estas características, y de permitir al usuario de este sistema, desarrollar sus programas con facilidad.

Los programas que se desarrollaron fueron tres:

1 - Programa monitor I.

Se encarga de inicializar el controlador, programar el puerto de comunicación serie (UART), y permite cargar los programas de usuario, depurarlos, ejecutarlos y comunicarlos. Este programa corre en cada uno de los controladores, está escrito en lenguaje ensamblador "8051", y se encuentra grabado en memoria de programa (EPROM).

2- Programa monitor II.

Este interactúa con el programa monitor I, corre en una microcomputadora PC, se comunica con el sistema distribuido a través del puerto serie RS-232C utilizando la interfaz que convierte RS-232C a RS-485, y está escrito en lenguaje C. Por medio de este programa el usuario puede transferir sus programas ya ensamblados, a cada uno de los controladores conectados al sistema distribuido, depurarlos y ejecutarlos.

3- Programa de usuario.

En su primera parte este programa contiene las rutinas de librería que se encargan de la transferencia de información mediante el "Sistema de POLLING", y después de estas rutinas se encuentra el programa de aplicación. El programa de usuario también corre en cada uno de los controladores, está escrito en lenguaje ensamblador "8051" y se carga en memoria de datos externa.

Una representación esquemática de la distribución física de los programas desarrollados es la siguiente:

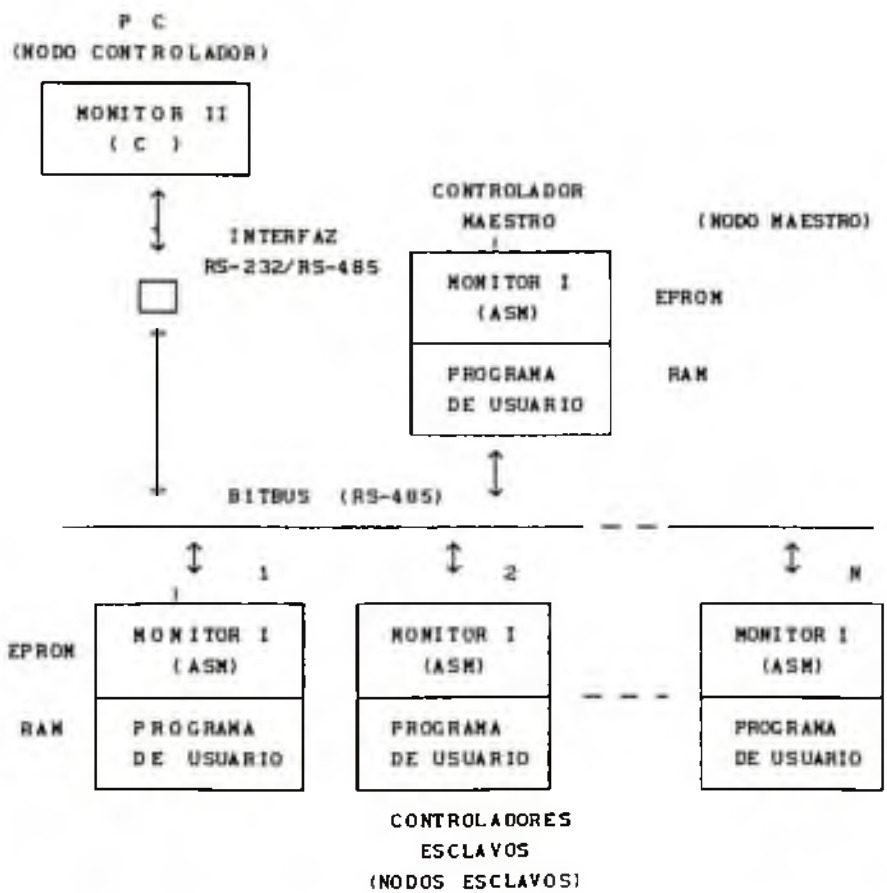


Figura 7.1 Distribución física de los programas del sistema distribuido 8051.

Como se observa en la figura 7.1, el programa monitor I es el mismo para todos los controladores incluyendo al nodo maestro, la única diferencia entre estos es: "la dirección (física/lógica) asignada a cada uno de ellos".

Establecimos las siguientes direcciones, las cuales no pueden ser utilizadas por otros nodos del sistema, porque el "SOFTWARE" que se desarrolló se diseñó de esta forma.

Dirección:

00 H	Dirección del nodo controlador (monitor II que corre en la PC).
FE H	Dirección del nodo maestro.
FF H	Dirección de "BROADCAST".

La dirección de "BROADCAST" es utilizada para enviar un mensaje a todos los nodos y es recibido por estos. Es utilizada por el nodo controlador (PC) para detener la ejecución de los programas de usuario que corren en todos los nodos, enviando un sólo comando.

Además de la diferencia en direcciones entre los nodos del sistema, los programas de usuario también son diferentes en parte, esto es, las rutinas del "Sistema de POLLING" que se encuentran dentro, son las mismas para todos los nodos esclavos, no así para el nodo maestro, el cual tiene como programa de usuario las rutinas que se encargan de realizar el "POLLING" siendo esta su única tarea.

A continuación veremos lo que realizan cada uno de los programas desarrollados.

1- PROGRAMA MONITOR I.

El programa monitor I se encuentra grabado en memoria de programa (EPROM 2764) a partir de la dirección: 0000H, a la cual salta el módulo controlador cada vez que se enciende o se reestablece; y está escrito en lenguaje ensamblador 8051. [3,5]

Primeramente, se tiene el traslado de todos los vectores de interrupción hacia las localidades 8K bytes adelante, entrando al área de memoria de datos externa (área de programa de usuario), a excepción del vector de interrupción por puerto de comunicación serie, el cual es atendido por la rutina "CINT" localizada el final del programa monitor I.

Después de este traslado, se tiene una tabla de saltos hacia rutinas de funciones. Actualmente sólo se tienen dos:

MRECV (0030H) preparación de recepción de un mensaje, y
MSEND (0033H) preparación del envío de un mensaje.

En seguida se hace la programación del puerto de comunicación serie, utilizando el TIMER 1, y estableciendo una velocidad de 10,416 bits/seg. siendo la más alta y más cercana entre la PC y los módulos controladores trabajando en el modo 3, esto se logra con el código siguiente:

```
MOV     TMOD,#20H       ; T1 8-BITS, AUTORECARGABLE.
MOV     SCON,#0F0H      ; MODO 3, MULTIPROCESADOR.
MOV     TH1,#-3         ; VEL. 10,416 BITS/SEG.
MOV     TCON,#40H       ; T1 ENCENDIDO.
```

Después se tiene una etapa de almacenamiento del contexto actual, almacenando el contenido de todos los registros y el contenido de la memoria de datos interna (RAM interna de 256 bytes), en el área de memoria de datos externa (RAM externa de 8192 byte) que empieza en la dirección: 1E80H y termina: 1FFFH, parte final de esta memoria.

El almacenamiento del contexto actual tiene como finalidad el de almacenar el ambiente inicial o actual sobre el cual va a operar el programa de usuario cuando éste se mande ejecutar.

A continuación se tiene un esquema de la disposición de las memorias de programa y de datos externas:

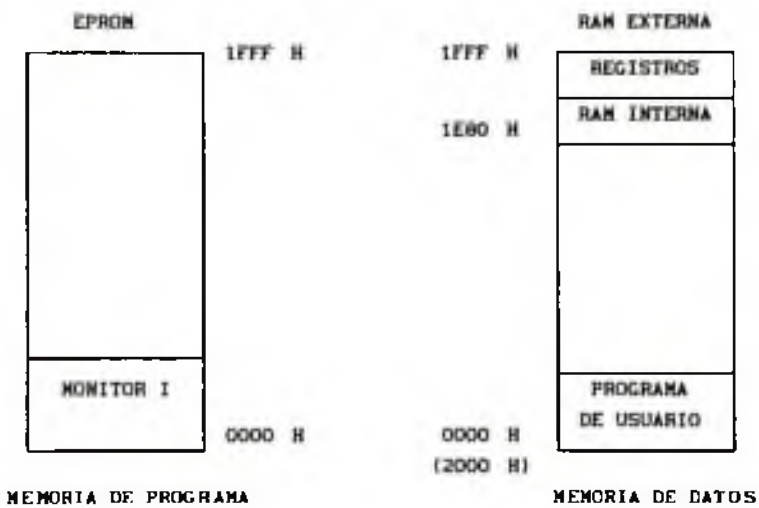


Figura 7.2 - Disposición de las memorias externas.

Una vez hecho lo anterior, el monitor entra en un ciclo de espera de comandos (funciones), los cuales son enviados por el programa monitor II que corre en la microcomputadora PC; para esto se tiene una serie de rutinas que se encargan de leer, modificar y ejecutar el programa de usuario, todo esto siguiendo el protocolo de comunicación del "Sistema Distribuido 8051".

Funciones: las desarrolladas en esta sección son:

FUNCION:	DESCRIPCION:
00 H	Modificación del área de registros almacenados.
01 H	Lectura del área de registros almacenados.
02 H	Modificación del programa en memoria.
03 H	Lectura del programa en memoria.
04 H	Modificación del área de RAM interna almacenada.
05 H	Lectura de la área de RAM interna almacenada.
06 H	Modificación de la memoria de datos externa.
07 H	Lectura de la memoria de datos externa.
08 H	Re-ejecución de programa (cambio de contexto).
09 H	Sensado de procesador.
0A H	Interrupción del procesador.
0F H	Reconocimiento.

Las funciones 00H y 01H permiten hacer modificaciones sobre el área de memoria de datos externa localizada entre las direcciones: 1F80Hy 1FFFH, con el fin de modificar el contenido de los 21 registros de función especial (SFR) en el momento de hacer el cambio de contexto y ejecutar el programa de usuario.

Las funciones 02H y 03H permiten hacer modificaciones sobre la memoria de datos externa, la cual se convierte en memoria de programa en el momento de invocar la función: 08H. Gracias a estas funciones podemos cargar nuestros programas (programas de usuario) dentro de cada módulo controlador.

Las funciones 04H y 05H permiten realizar modificaciones sobre el área de memoria de datos externa localizada entre las direcciones: 1E80H y 1F7FH, con el fin de modificar el contenido de la memoria de datos interna (256 bytes) en el momento de hacer el cambio de contexto y ejecutar el programa de usuario. Hay que notar que dentro de estos 256 bytes de memoria de datos interna se encuentran los 21 registros de función especial modificados por las funciones 00H y 01H.

Es importante notar que las 6 funciones anteriores solamente estan realizando modificaciones sobre áreas de memoria de datos externa (RAM externa), y solamente se realizan las modificaciones descritas, en el momento de hacer el cambio de contexto y ejecutar el programa de usuario.

Las funciones 06H y 07H permiten modificar el contenido de la memoria de datos externa, desde la localidad: 0000H hasta la 1FFFH, por lo tanto con estas dos funciones es posible realizar las mismas operaciones que con las 6 funciones anteriores, pero para lograr esto sería necesario recordar perfectamente las direcciones donde se encuentran cada una de las áreas de almacenamiento.

La función 08H es la que se encarga de hacer el cambio de contexto, antes de saltar al área de memoria de datos externa donde se encuentra localizado el programa de usuario previamente cargado, normalmente éste se encuentra a partir de la dirección: 2000H del área de memoria de programa, la cual corresponde a la dirección: 0000H del área de memoria de datos, y esto se debe al efecto de ESPEJO dado por el circuito de selección de memorias diseñado.

La función 09H permite saber si un módulo controlador dado, se encuentra conectado al sistema, y si éste se encuentra en estado de espera de comandos.

La función 0AH se encuentra aquí sólo por precaución, pues este comando es utilizado cuando un programa de usuario se encuentra en ejecución y deseamos interrumpirlo, pero que si en un momento dado olvidamos que no lo está, y enviamos este comando, no ocasionará ningún problema.

La función 0FH se utiliza para responder a las funciones anteriores en forma positiva.

A continuación tenemos el mapa de memoria del sistema:

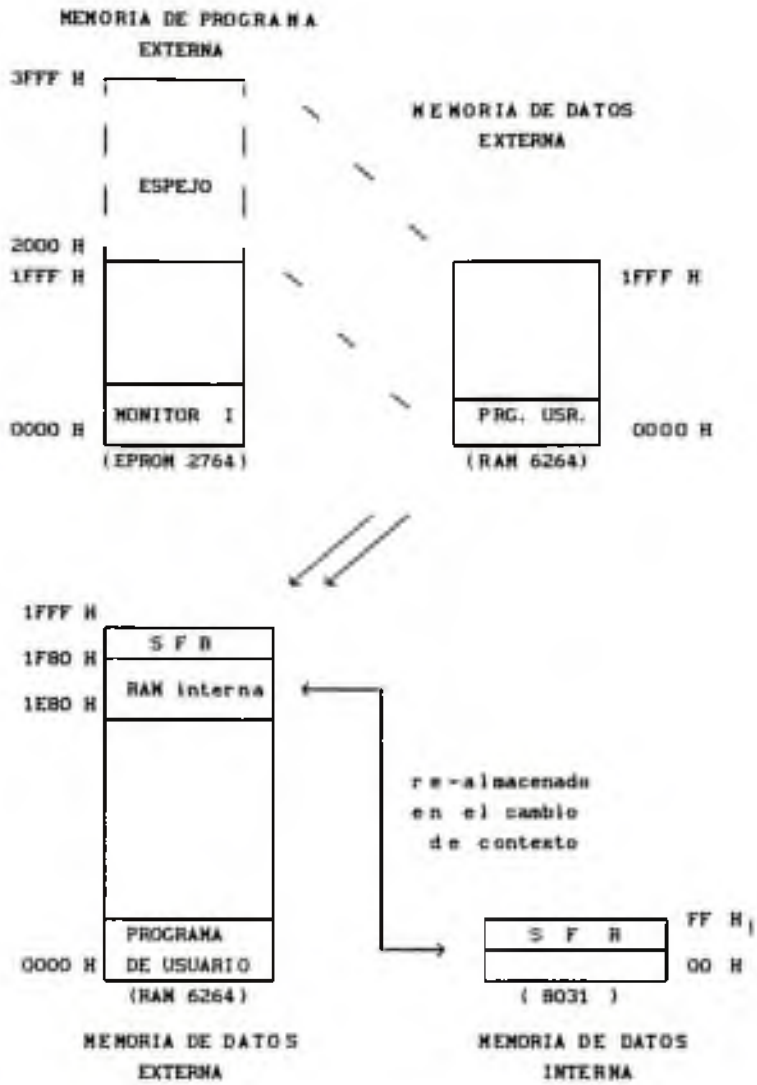


Figura 7.3 - Mapa de memoria del sistema.

Para la transmisión y recepción de bytes basandose en el protocolo de comunicación establecido, se desarrollaron 3 rutinas:

RECV - Rutina para recibir un byte.

SEND - Rutina para transmitir un byte.

SHEAD - Rutina para transmitir el encabezado del mensaje.

La rutina SHEAD se desarrolló para transmitir el encabezado de cada mensaje, el cual está formado por: el byte de dirección destino = 00H (dirección del nodo controlador) con el 9o. bit puesto y el byte de dirección fuente = XXH (dirección propia del nodo), porque estos bytes son los mismos para todos los mensajes enviados por el nodo en cuestión.

A continuación muestro dos ejemplos de como el nodo controlador (programa monitor II corriendo en la microcomputadora PC) envía las peticiones de estas funciones a los módulos controladores conectados al sistema y lo que éstos devuelven.

EJEMPLO 1: Modificación del programa de usuario (función 02H).

En este ejemplo se envía un bloque de 2 bytes de código de programa hacia el módulo controlador con dirección: 5DH, con dirección inicial: 2030H.

El nodo controlador PC envía:

Dir. destino	=	5D H	(9o bit puesto).
Dir. fuente	=	00 H	(Dir. propia (PC)).
Función	=	02 H	
Longitud	=	05 H	(3 + 2 bytes de código).
Info.	=	02 H	(Long. código).
Info.	=	30 H	(DPL).
Info.	=	20 H	(DPH).
Info.	=	12 H	(Código).
Info.	=	BF H	(Código).
C D E	=	14 H	(Check sum).

El Código de Detección de Error (C D E) se obtiene sumando cada byte que se transmite o se recibe, enviando o recibiendo éste en forma negativa, cada vez que se llama a las rutinas: SHEAD, SEND y RECV, alojando su valor en la localidad de memoria interna: 7FH, la cual no debe ser usada por el usuario.

Si el mensaje se recibió correctamente, es decir, la suma del CDE calculado en la recepción, con el valor del CDE recibido es cero, entonces:

El módulo controlador 5DH responde:

Dir. destino	=	00 H	(9o bit puesto).
Dir. fuente	=	5D H	(Dir. propia de módulo).
Función	=	0F H	(Reconocimiento).
Longitud	=	00 H	
C D E	=	94 H	

En caso contrario, no se responde a la petición, y se entra de nuevo al ciclo de espera de comandos.

EJEMPLO 2: Ejecución del programa de usuario (función 08H).

En este ejemplo se envia la petición de ejecución junto con la dirección en la cual se va a iniciar la ejecución del programa que previamente ha sido almacenado en el módulo controlador con dirección: 3E H. La dirección de inicio es: 2050H.

El nodo controlador PC envia:

Dir. destino	=	3E H	(9o bit puesto).
Dir. fuente	=	00 H	(Dir. propia (PC)).
Función	=	08 H	
Longitud	=	02 H	(2 bytes de dirección).
Info.	=	50 H	(PCL).
Info.	=	20 H	(PCH).
C D E	=	48 H	(Check sum).

Si el mensaje se recibió correctamente, entonces:

El módulo controlador 3EH responde:

Dir. destino	=	00 H	(9o bit puesto).
Dir. fuente	=	3E H	(Dir. propia del módulo).
Función	=	0F H	(Reconocimiento).
Longitud	=	00 H	
C D E	=	B3 H	

En caso contrario, no se responde a la petición, y se entra de nuevo al ciclo de espera de comandos.

Como se observa en los ejemplos, cada módulo controlador tiene una dirección asociada, la cual se encuentra grabada en EPROM, y es verificada cada vez que se recibe el primer byte de cada comando, con esto, sólo el módulo controlador direccionado recibirá los subsecuentes bytes que forman el comando, gracias a la facilidad dada por el microcomputador 8031 de habilitar o deshabilitar la recepción de tramas de 11 bits (1 byte por trama) en la cual el noveno bit está en cero.

Al final del programa monitor I, y en forma separada, se encuentra la rutina "CINT" que sirve a las interrupciones debidas al puerto de comunicación serie (UART), utilizando el protocolo de comunicación que hemos diseñado. Las interrupciones permanecen deshabilitadas mientras se encuentra en ejecución el programa monitor I, y son habilitadas cuando se ejecuta el programa de usuario.

La rutina de servicio "CINT" se encuentra dividida en 2 partes:

a)- Atención a la interrupción por un byte recibido.

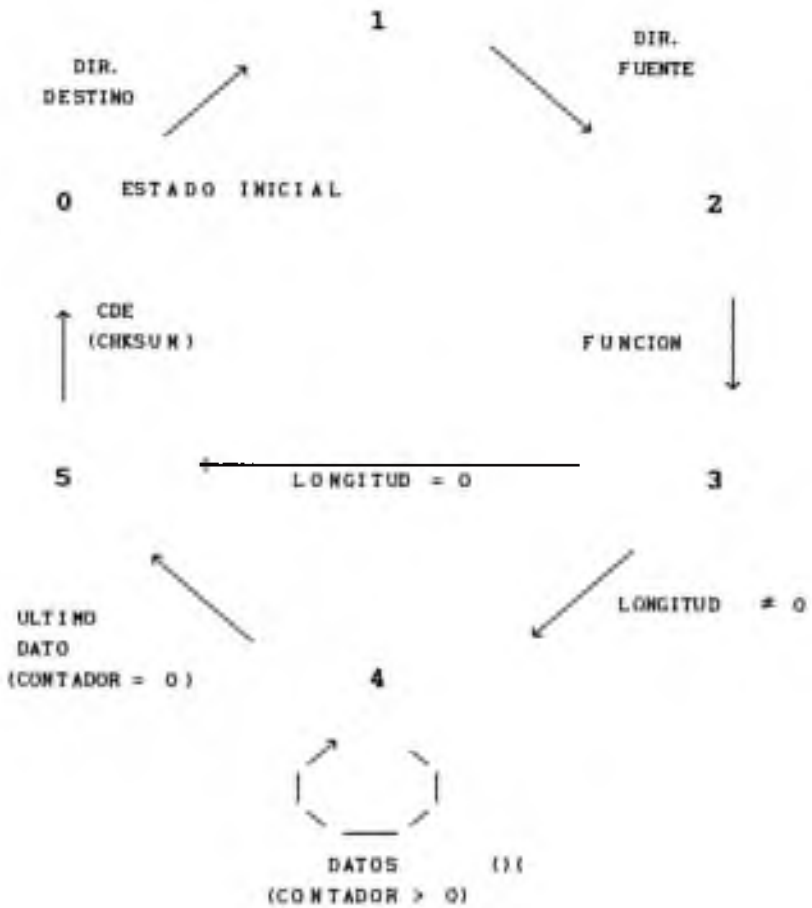
b)- Atención a la interrupción por un byte transmitido.

Se logra dividirlas por medio del bit T1 (bit 5 del puerto 3, no utilizado por el "TIMER 1"), el cual indica que se esta transmitiendo un mensaje, y además permite habilitar el "DRIVER" de transmisión de la interfaz RS-485 (75176).

a)- **Rutina CINT 1a parte:** se encarga de ir recibiendo cada uno de los bytes que forman el mensaje, el primero de ellos contiene la dirección del módulo controlador al cual va dirigido el mensaje, se verifica, y si coincide con la dirección propia entonces se hacen los arreglos necesarios para recibir los siguientes bytes también por interrupción, y cada uno de estos bytes se van almacenando en un área de memoria de datos externa que denominamos: "BUFFER" de recepción.

La forma en como se van recibiendo y analizando cada uno de los bytes que forman el mensaje, es controlado por un autómata finito deterministico mostrado en la figura 7.4.

Figura 7.4 - Diagrama de estados del autómata utilizado para la recepción de mensajes.



Para el funcionamiento del autómata y de ambas partes de la rutina de servicio "CINT", se utilizan los 8 registros contenidos en el segundo banco de registros del "8031" (banco 1), seleccionando este banco cada vez que se entra a la rutina de servicio por medio de los bits: RS1 y RS0 (0,1).

La siguiente lista nos muestra el uso dado a cada uno de los registros contenidos en el segundo banco de registros.

BANCO 1 (RS1,RS0 = 01)

- R0 - Almacenamiento del registro PSW.
- R1 - Almacenamiento del Acumulador.
- R2 - Almacenamiento del registro DPL.
- R3 - Almacenamiento del registro DPH.
- R4 - Apuntador bajo al "BUFFER" de recepción.
- R5 - Apuntador alto al "BUFFER" de recepción.
- R6 - Contador de bytes de datos.
- R7 - Estado actual del autómata.

Dado el uso que se le dá a este banco de registros, el usuario no podrá utilizarlo en sus programas si es que desea establecer comunicaciones a través del UART del "8051" por interrupción.

Además de la restricción anterior, si el usuario desea utilizar las rutinas del "Sistema de POLLING" contenidas en la parte del programa de usuario, las cuales se veran más adelante, también se le restringirá el uso del primer banco de registros.

El "BUFFER" de recepción se encuentra inicialmente en las localidades que inician en la dirección: 1D7AH de memoria de datos externa (RAM externa), pero puede ser cambiada a otra área por medio de la pequeña rutina: "MRECV" que sólo modifica el contenido de las localidades: 1D78H y 1D79H las cuales apuntan hacia el "BUFFER" de recepción. Esta rutina se puede invocar haciendo un llamado a la localidad: 0030H.

Un ejemplo de como asignar una nueva área de memoria al "BUFFER" de recepción, colocándolo a partir de la dirección 1500H usando esta rutina, se tiene a continuación:

```
MRECV    EQU   0030H
         ORG   2000H
           .
           .
           .
         MOV   DPTR,#1500
         CALL  MRECV
           .
           .
           .
```

Al terminar de almacenar el mensaje que se ha recibido, se verifica si éste llegó correctamente por medio del "CHECK SUM" (CDE), si hubo error, se reinicializa la recepción deshabilitando la interrupción por tramas en las cuales el 9o. bit esté en cero, y se regresa al estado 0 del autómata controlador.

Si el mensaje llegó correctamente, se analiza éste con el fin de saber si se trata de un comando del nodo controlador (PC) para detener el programa de usuario que se encuentra actualmente en ejecución, y trasladarse entonces al programa monitor.

Si no se trata de un comando del nodo controlador, entonces se hace el llamado a la rutina "LRINT" que se encuentra en la dirección: 2023H, entrando a los dominios de las rutinas del "Sistema de POLLING" las cuales veremos más adelante en los programas del usuario.

Al finalizar la ejecución de estas rutinas, se reinicializa la recepción deshabilitando la interrupción por tramas en las cuales el 9o. bit es cero, se regresa al estado 0 del autómata controlador, y se regresa el control del procesador al programa de usuario que fué interrumpido, esto a través del siguiente código.

```
SETB  SM2          ; Deshab. rec. 9o bit = 0.
MOV   R7,#0        ; Estado autómata = 0.
MOV   DPH,R3       ; Cambio de contexto.
MOV   DPL,R2
MOV   A,R1
MOV   PSW,R0
CLR   RS0          ; Sel. banco 0
RETI               ; Regreso de la interrupción.
```

b)- Rutina **CINT 2a parte**: se encarga de ir transmitiendo cada uno de los bytes que forman el mensaje, sin contar el primero, puesto que la interrupción se genera sólo cuando un byte ha sido transmitido. Por lo tanto, cuando se desea transmitir un mensaje, primeramente debemos transmitir el primer byte del mensaje, el cual contiene la dirección del módulo controlador a quien va dirigido, y al ser transmitido, entra en operación la rutina de interrupción transmitiendo los bytes restantes.

Para transmitir el primer byte del mensaje, hay que habilitar la transmisión (T1 = 1) y deshabilitar la recepción, poner en uno el noveno bit de la trama que va a contener este byte, colocar en los registros R4 y R5 la dirección donde se localiza nuestro "BUFFER" de transmisión, y entonces enviar el byte.

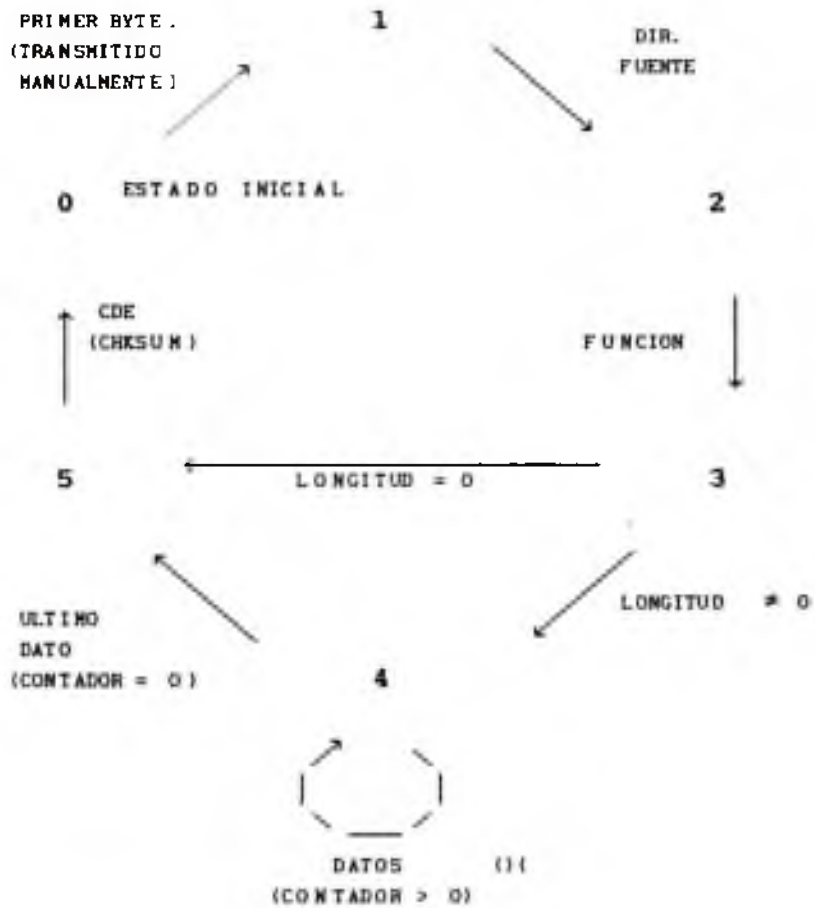
Lo anterior se realiza fácilmente utilizando la rutina "MSEND", la cual se ejecuta haciendo un llamado a la dirección: 0033H. A continuación tenemos un ejemplo del llamado a esta rutina, donde deseamos transmitir un mensaje previamente almacenado en la dirección: 1C00H.

EJEMPLO:

```
MSEND   EQU     0033H
        ORG     2000H
          .
          .
          .
        MOV     DPTR,#1C00H
        CALL    MSEND
```

La transmisión de mensajes se realiza en una forma muy parecida a la recepción, pues se utiliza el mismo protocolo, y el autómata que la controla es prácticamente el mismo (ver figura 7.5), con la única diferencia de que el primer byte, se transmite en forma manual, y los bytes restantes se van transmitiendo automáticamente por interrupción.

Figura 7.5 - Diagrama de estados del autómata utilizado para la transmisión de mensajes.



Al enviar el último byte del mensaje, se deshabilita la transmisión y se hace una llamado a la rutina "LTINT" la cual se encuentra en la dirección: 2028H. Al igual que la rutina "LRINT", "LTINT" se encuentra dentro de las rutinas del "Sistema de POLLING", las cuales veremos más adelante en los programas de usuario.

Al finalizar la ejecución de estas rutinas, se reinicializa la recepción deshabilitando la interrupción por tramas en las cuales el 9o. bit es cero, se regresa al estado 0 del autómata controlador, y se regresa de la interrupción; igual que en la parte a) de estas rutinas.

Rutinas de transferencia de mensajes por interrupción.

Finalmente, tenemos dos rutinas de funciones utilizadas para las comunicaciones por interrupción, las cuales hemos venido mencionando. Por ser de las más importantes y bastante simples, las muestro a continuación.

Rutina de preparación de "BUFFER" de recepción.

```
MRECV:  PUSH  DPH              ; Coloca en las direcciones
        PUSH  DPL              ; 1D78H y 1D79H la dirección
        MOV   DPTR,#1D78H      ; contenida en el apuntador
        POP   ACC              ; a datos DPTR.
        MOVX  @DPTR,A
        INC   DPTR
        POP   ACC
        MOVX  @DPTR,A
        RET
```

Rutina de envío de mensaje.

```
MSEND:  MOV   A,#60H          ; Espera un momento antes de
MSEND0: DJNZ  ACC,MSEND0      ; empezar a transmitir.
        SETB  T1              ; Habilita transmisión.
        CLR   REN             ; Deshab. recepción.
        SETB  TB8             ; Pone el 9o. bit en uno
        MOVX  A,@DPTR         ; por ser byte de dirección.
        INC   DPTR            ; Apunta al sig. byte-mensg.
        CLR   RS1             ; Selecciona banco 1.
        SETB  RS0
        MOV   R7,#1           ; Se pone en edo 1 de autóm.
        MOV   R5,DPH          ; Guarda dir. sig. byte
        MOV   R4,DPL
        CLR   RS0             ; Selecciona banco 0
        MOV   SBUF,A          ; Transmite 1er byte.
        MOV   7FH,A           ; Lo pone en CDE.
        RET
```

2- PROGRAMA MONITOR II.

El programa monitor II está escrito en lenguaje C, fue compilado con Turbo C (compilador de Borland) y corre en una microcomputadora PC (IBM-PC o compatible). Este programa interactúa con el programa monitor I que corre en cada uno de los módulos que componen el sistema distribuido.

La PC se conecta al BUS del sistema como si se tratase de otro módulo, a través del puerto de comunicación serie RS-232, con la ayuda de la interfaz construida, que convierte RS-232C a RS-485.

El programa monitor II se encarga de interactuar con el usuario del sistema distribuido de una manera amable, permitiéndole cargar, depurar y mandar a ejecutar, sus programas de aplicación, previamente ensamblados, dentro del sistema distribuido.

El usuario puede desarrollar sus programas en lenguaje ensamblador sobre cualquier editor de texto que no introduzca caracteres especiales, ensamblarlos con algún ensamblador 8051 que esté a sus alcance, y cargar el código hexadecimal obtenido, en el/los módulo/s del sistema distribuido utilizando el programa monitor II (nodo controlador).

Los problemas a resolver para la realización de este programa fueron los siguientes:

a)- La programación del puerto de comunicación serie RS-232C de la PC.

b)- La transmisión y recepción de bytes.

c)- Las funciones brindadas por el monitor II.

A continuación veremos como se resolvieron cada uno de ellos.

a)- Programación de puerto de comunicación serie.

El puerto de comunicación serie RS-232C de una microcomputadora PC está compuesto por un C.I. 8250. Este circuito es programado a través de 7 registros, los cuales son accesados como puertos utilizando las siguientes funciones de Turbo C:

```
<byte> = inportb ( <word> );      /* lee un <byte> del
                                     puerto <word>      */
outportb ( <word>, <var> );       /* escribe un <byte>
                                     en el puerto <word> */
```

Para accesar el puerto de comunicación serie 1 (COMM 1), se utiliza la dirección: 3F8 H, la cual corresponde al registro 0, accesando los registros de la siguiente forma:

REG_0 - 3F8	REG_3 - 3FB	REG_6 - 3FE
REG_1 - 3F9	REG_4 - 3FC	REG_7 - 3FF
REG_2 - 3FA	REG_5 - 3FD	

Las funciones de estos registros son:

REG_0 - "BUFFER" de transmisión/recepción de bytes.
REG_1 - Habilitación de interrupción.
REG_2 - Identificación de interrupción.
REG_3 - Formato de los datos.
REG_4 - Control salidas de RS-232.
REG_5 - Estado del puerto.
REG_6 - Estado de entradas de RS-232.
REG_7 - "Scratch Pad".
REG_0/8 - Divisor de velocidad de transm. LSB.
REG_1/9 - Divisor de velocidad de transm. MSB.

El registro 0 tiene 3 usos: recibir bytes, transmitir bytes, y colocar el byte bajo (R8) del valor del divisor con el cual se va dividir la frecuencia del reloj de referencia, para generar la velocidad de comunicación.

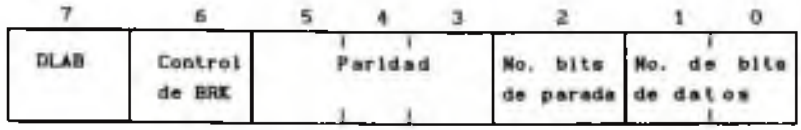
El registro 1 tiene 2 usos, habilitar la interrupciones y colocar el byte alto (R9) del valor del divisor. El valor del divisor (alto y bajo) se obtiene con la fórmula:

$$\text{DIVISOR} = \frac{\text{FRECUENCIA DEL RELOJ DE REFERENCIA}}{16 \times \text{VELOCIDAD DE COM. DESEADA}}$$

Para una microcomputadora IBM-PC, la frecuencia del reloj de referencia es de: 1.8432 MHz.

La velocidad de comunicación máxima obtenida en este sistema, fue de: 10.4 K bits/seg. y esto debido a que las diferencias entre las velocidades de comunicación de la PC y los módulos son muy grandes. Asi el valor del Divisor = 000B H.

El formato que se le da a las tramas de comunicación es dado por el registro 3, el cual tiene la siguiente forma:



Donde:

DLAB - indica que se va a enviar los bytes bajo y alto del valor del divisor, a través de los registros R0/8 y R1/9 del siguiente acceso.

Control

BRK - indica si se envía o no (ON/OFF) señal de BREAK (no usada).

Paridad - para seleccionar si se utiliza bit de paridad o no, si ésta es par o impar, o es puesto o limpiado por nosotros mismos.

No. bits de parada - selecciona el número de bits de parada que va a contener la trama.

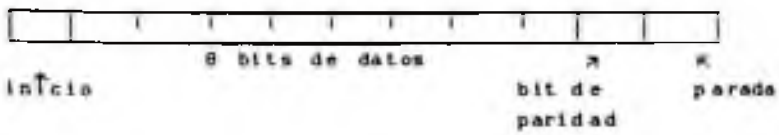
No. bits de datos - selecciona el número de bits de datos (5, 6, 7 u 8 bits de datos).

Como el 8250 no maneja un noveno bit, como el que utilizamos en los módulos controladores 8051 para direccionar los mensajes, nosotros utilizamos el bit de paridad, el cual controlamos a través de los bits 3, 4 y 5 del registro 3, poniendo las combinaciones:

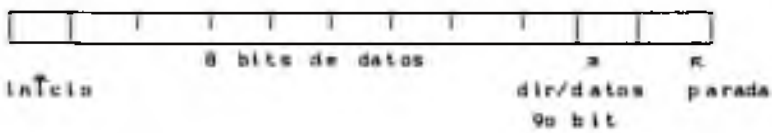
bit 5	bit 4	bit 3	
1	0	1	poner en uno (marca).
1	1	1	poner en cero (espacio).

De esta forma, obtenemos el siguiente formato de tramas, el cual es similar al de los módulos controladores.

Formato de las tramas manejadas por el monitor II (PC):



Formato de las tramas manejadas por los módulos controladores:



Para tener un control sobre la habilitación de la interfaz, construída para interconectar la microcomputadora PC al sistema distribuido, se utilizan las líneas DTR (pin 20) y RTS (pin 4); las cuales son controladas por medio del registro 4, el cual tiene el siguiente formato:

7	6	5	4	3	2	1	0
0	0	0	LLP	GPO2	GPO1	RTS	DTR

Poniendo o limpiando los bits RTS y DTR, obtenemos el control de las lineas fisicas del puerto de comunicación serie.

Además, es importante notar que al leer el "BUFFER" de recepción a través del registro 0, lo limpiamos.

La siguiente rutina fué realizada para programar el puerto de comunicacion, de acuerdo a nuestras necesidades.

```
#define   COMM   0x3F8              /* dirección del 8250        */

void ComInit (void)
{
    outportb (COMM+3,0x80);         /* pone bit 7 -DLAB-         */
    outportb (COMM+0,0x0B);         /* d.p.b.                    */
    outportb (COMM+1,0x00);         /* d.p.a.  10.472 K b/s      */
    outportb (COMM+3,0x3B);         /* 8 datos, 1 par, espacio   */
    outportb (COMM+4,0x08);         /* limpia RTS y DTR          */
    inportb  (COMM+0);              /* limpia BUFFER             */
}
```

Para obtener información detallada del uso de los registros del C.I. 8250, puede consultarse algún texto de "Comunicaciones en IBM-PC". [6]

b)- Transmisión y recepción de bytes.

Para realizar la transferencia de bytes se realizaron 3 rutinas:

1- SendCI (<byte>):

Transmite un byte por el puerto serie poniendo el bit de paridad en uno, indicando que se trata de un byte de direccionamiento de mensaje, y habilita el transmisor de la interfaz RS-232/RS-485. Se utiliza para enviar el primer byte de cada mensaje (comando) hacia los módulos controladores del sistema. Después de haber enviado el byte, limpia el bit de paridad y deshabilita el transmisor.

2- SendC (<byte>):

Transmite un byte por el puerto serie sin modificar el bit de paridad, manteniendolo en cero; y habilita el transmisor de la interfaz RS-232/RS-485, y lo deshabilita después de transmitir el byte. Se utiliza para enviar los bytes siguientes al byte de dirección.

3- RecvC ():

Recibe un byte por el puerto serie, unicamente revisando el estado del puerto hasta que el byte se recibe.

Las tres rutinas utilizan la variable "bcc" para almacenar la suma de cada byte que se transmite o se recibe en cada mensaje, con la finalidad de transmitirlo como byte de C.D.E. para el caso de envio de mensaje; y para detectar errores en la comunicación para el caso de recepción de mensaje. A continuación tenemos estas rutinas.

```
void SendCI(byte b)
{
  inportb(COMM+0);                              /* Limpia BUFFER    */
  outportb(COMM+3,inportb(COMM+3) & 0xEF); /* Coloca 9no bit   */
  outportb(COMM+4,0x0A);                        /* Habilita Transm. */
  outportb(COMM+0,b);                           /* Envia byte       */
  bcc = b;                                      /* Inic. CHEKSUM    */
  while((inportb(COMM+5)&0x060) != 0x060)
      ;
  outportb(COMM+4,0x08);                        /* Deshab. Transm.  */
  outportb(COMM+3,inportb(COMM+3) | 0x10); /* Limpia 9no bit   */
  inportb(COMM+0);
}
```

```
void SendC(byte b)
{
  inportb(COMM+0);                    /* Limpia BUFFER de Tr.   */
  outportb(COMM+4,0x0A);              /* Habilita Transmisión   */
  outportb(COMM+0,b);                 /* Envía byte             */
  bcc = (bcc + b) % 256;              /* Calcula CHEKSUM        */
  while((inportb(COMM+5)&0x060) != 0x060)
      ;
  outportb(COMM+4,0x08);              /* Deshabilita Transm.    */
  inportb(COMM+0);
}



byte RecvC()
{
  byte ca;
  int  tmout=0;

  while((inportb(COMM+5)&0x01)!=0x01 && tmout++<16383)
      ;
  if (tmout >= 16383)                 /* Pregunta si se terminó   */
     return(0x00);                    /* el tiempo de espera.     */
  ca = inportb(COMM+0);               /* Obtiene el byte recibido */
  bcc = (bcc + ca) % 256;             /* Obtiene CHEKSUM          */
  return(ca);
}
```

c)- Funciones brindadas por el monitor II.

Utilizando las anteriores rutinas de comunicación, podemos entonces enviar y recibir mensajes, siguiendo el protocolo de comunicación establecido (capitulo 6). Las funciones brindadas en el monitor II, son las mismas que nos permite realizar el monitor I, diseñadas con este fin.

La funciones son seleccionadas por medio de las teclas de funciones: F1, F2, ... ,F8 y F10, asi como las teclas: ESC y M del teclado de la microcomputadora PC, y son:

F1 - Ayuda.

Muestra una explicación breve de las funciones que brinda el programa monitor.

F2 - Archivo.

Permite cargar archivos en disco que contienen código hexadecimal de programas de aplicación 8051, en los módulos controladores 8051 conectados al sistema.

F3 - Memorias.

Por medio de esta función podemos leer y modificar el contenido de las 3 áreas de memoria con que cuenta cada módulo controlador del sistema, y estas son:

MemExt.
Memoria externa, localidades desde: 0000H a la 1FFFH.

MemInt.
Memoria interna, localidades desde: 00H a la FFH.

MemProg.
Memoria de programa de usuario, localidades desde: 2000H a la 3FFFH.

La operación realizada por "MemProg" también puede ser realizada utilizando la función "MemExt", a causa del efecto "espejo" producido por los circuitos de selección de memorias (capítulo 5).

F4 - Registros.

Esta función permite leer y modificar el contenido del área de los 21 registros, de cada módulo controlador conectado al sistema distribuido.

F5 - Congelar.

Permite detener la ejecución de los programas que corren en cada uno de los módulos controladores del sistema, enviando un solo comando. Se utiliza la dirección de "BROADCAST": FFH, la cual es reconocida por todos los módulos del sistema, como si fuera su propia dirección.

F6 - Ejecutar.

Esta función permite mandar a ejecutar programas previamente cargados en los módulos controladores del sistema.

F7 - Interrumpir.

Utilizando esta función podemos detener el funcionamiento de algún programa que se encuentre en ejecución, en alguno de los módulos del sistema.

F8 - POLLING.

Esta función es especial, y sólo se utiliza cuando se han cargado y mandado ha ejecutar todos los programas de aplicación en cada uno de los controladores, incluyendo principalmente, al programa "maestro", el cual se encarga de coordinar las comunicaciones del sistema.

Por medio de esta función le enviamos al programa "maestro" el número y direcciones de los módulos controladores conectados en la red, de esta forma el nodo maestro inicia su trabajo de "POLLING" para coordinar y permitir las comunicaciones del sistema distribuido.

Además, esta función nos permite convertir a la computadora PC en un nodo más del sistema distribuido, con dirección: 05, una vez que se han enviado las direcciones de POLLING.

F10 - Salir.
ESC - Salir.

Con cualquiera de estas dos teclas podemos finalizar la ejecución del programa monitor II, el cual ya no es necesario para el funcionamiento del sistema distribuido 8051, una vez que éste se encuentra en operación.

M - Módulo.

Con esta función podemos verificar si algún módulo controlador específico se encuentra conectado al sistema, y además si éste se encuentra en estado de espera de comandos.

La tecla "ESC" también es utilizada para regresar al menu principal, cuando nos encontramos dentro de algunas funciones; y las teclas de movimiento del cursor, nos permiten desplazarnos dentro de los mapas de memoria, cuando estamos modificando el contenido de alguna de estas áreas.

A continuación tenemos un ejemplo de como es enviado el comando: ejecución de programa (Función 08H), desde el monitor II hacia uno de los módulos del sistema, y como se recibe la respuesta a este comando:

```
    SendCI (DesAddr);              /* dir. del módulo destino */
    SendC (MyAddr);                /* dir. PC = 00 H          */
    SendC (0x08);                  /* función 08H             */
    SendC (0x02);
    SendC (pcl);                   /* dir. inicio de programa */
    SendC (pch);
    SendC (256 - bcc);             /* envio del ChkSum        */

    bcc = 0x00;
    if (RecvC() == MyAddr)
      if (RecvC() == DesAddr)
        if (RecvC() == 0x0F)       /* reconocimiento          */
          if (RecvC() == 0x00)
            {
             RecvC();
             if (bcc == 0)         /* verificación del ChkSum */
               return(CORRECTO);
            }
    return(ERROR);
}
```

Así, el monitor I y el monitor II serán las herramientas para el desarrollo de programas de aplicación para este sistema distribuido.

3- PROGRAMA DE USUARIO.

El programa de usuario corre en cada uno de los módulos controladores que componen el "sistema distribuido 8051", está escrito en lenguaje ensamblador "8051", y se carga en memoria de datos externa, en el área de programas de usuario.

Este programa contiene las rutinas de librería que se encargan de la transferencia de información entre los módulos del sistema mediante el "Sistema de POLLING" (capítulo 3), las cuales son usadas por el programa de aplicación que se escribe a continuación de estas.

El programa de usuario debe iniciar a partir de la dirección: 2000 H, colocando en las direcciones 2023H y 2028H las rutinas que sirven a las interrupciones de mensaje recibido y mensaje transmitido respectivamente, debido a que la transferencia de mensajes en el sistema se realiza por interrupción.

La estructura que tiene el programa de usuario se puede representar con la figura siguiente:

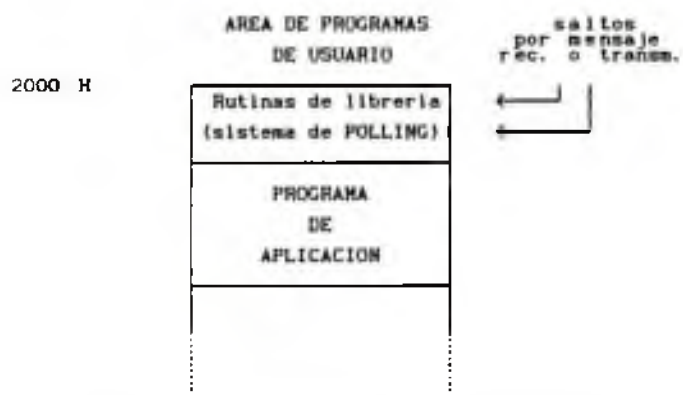


Figura 7.6 - Estructura del programa de usuario dentro de su área de memoria.

Las rutinas de librería utilizadas por cada uno de los módulos del sistema, son exactamente las mismas, a excepción del módulo "maestro", el cual es el encargado de realizar la comunicación por "POLLING", conteniendo en sus rutinas de librería el mecanismo de coordinación.

Nuestro estudio, en este capítulo, lo dedicaremos unicamente a describir el funcionamiento de las rutinas de librería realizadas; y en el capítulo siguiente, se mostrará- un ejemplo de aplicación del sistema distribuido 8051, donde se emplean estas rutinas.

Primeramente veremos como estan construídas las rutinas de librería que utilizan los módulos "esclavos", y después las utilizadas por el módulo "maestro".

1- Rutinas de librería de los módulos "esclavo".

Las rutinas de librería en los módulos esclavos son 2, la que responde a la llegada de un mensaje: **LRINT**, y la que responde a la terminación del envío de otro mensaje **LTINT**.

Al inicio de estas rutinas se definen las funciones que el sistema de "POLLING" maneja, utilizadas para indicar el tipo de mensaje que se envía o se recibe, y éstas son:

0F H - Reconocimiento ("ACKN").
10 H - Señal de "POLLING" ("POLL").
11 H - Direcciones para el "POLLING" ("ADDPOL").
12 H - Señal de fin de mensaje ("TREND").
15 H - Paquete de datos ("DATOS").

También se definen algunas otras constantes como:

MAESTRO = FE H - Dirección del módulo "maestro".
MYADDR = XX H - Dirección propia del módulo donde se carga el programa de usuario.
DSTADDR = XX H - Dirección de alguno de los módulos del sistema, a quien se le envía el mensaje.

Después se definen ciertas áreas de memoria externa donde se almacenan mensajes previamente preparados, con el fin de agilizar la comunicación. Los mensajes que se preparan son:

Señal de fin de mensaje:
dirección de inicio: 1DF8 H = "MEND"
contenido: MAESTRO,MYADDR,TREND,00H

Reconocimiento:
dirección de inicio: 1DFC H = "RECON"
contenido: MAESTRO,MYADDR,ACKN,00H

Paquete de información:
dirección de inicio: 1E40 H = "MENSG"
contenido: DSTADDR,MYADDR,DATOS,01H

Para la recepción y transmisión de los mensajes, se definieron ciertas áreas de memoria externa, para almacenarlos en forma temporal:

Los mensajes que serán enviados (BUFFER de transmisión).
Los mensajes que se reciben (BUFFER de recepción).
Los mensajes previamente preparados.
Y el "BUFFER" de llegada.

La distribución del área de memoria externa se muestra en las siguientes figuras.

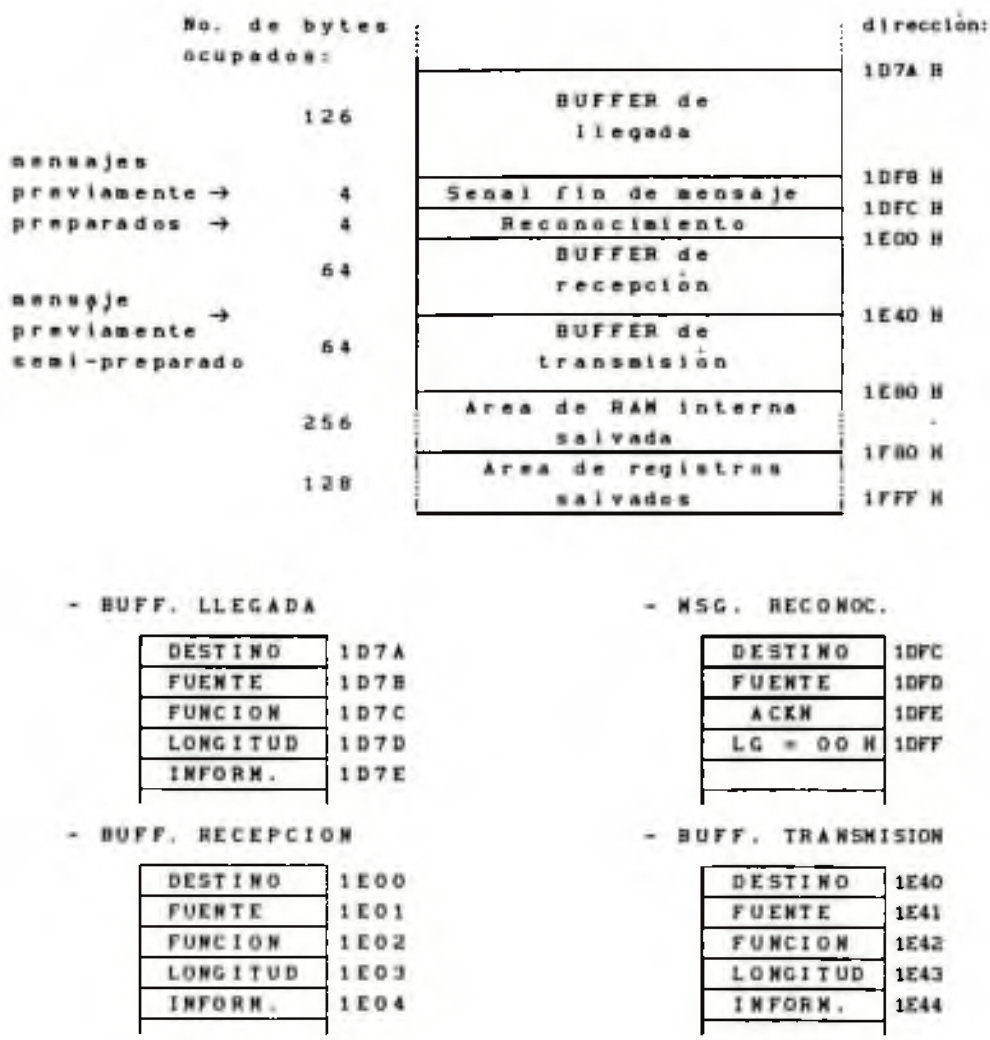


Figura 7.7 - Distribución del área de memoria de datos/programa externa.

Rutina LRINT:

Esta rutina inicia en la dirección 2023 H, a la cual se salta, cada vez que un mensaje ha sido recibido. Esta función tiene la siguiente forma:

```
El mensaje recibido
proviene del módulo
MAESTRO ?
     Si:
        Se trata de una
        señal de "POLLING" ?
                Si:
                   Hay un mensaje
                   de reconocimiento
                   esperando a ser
                   transmitido ?
                          Si:
                             Envía"RECON".
                             Pone bandera de envío de
                                 fin de mensaje.
                             Retorno de interrupción.
                          No:
                             Hay un mensaje esperando
                             en el BUFFER de transmisión ?
                                  Si:
                                     Envía "MENSG".
                                     Pone bandera de envio
                                         fin de mensaje.
                                     Retorno de interrupción.
                                  No:
                                     Envía "MEND".
                Si:                  Retorno de interrupción.
     No:
        Es un mensaje
        de reconocimiento ?
                Si:
                   El BUFFER de transmisión
                   está lleno ?
                          Si:
                             Quita bandera de
                                BUFFER de transmisión lleno.
                          No:Retorno de interrupción.
                No:
                   El BUFFER de recepción
                   está vacio ?
                          Si:
                             Guarda el mensaje recibido
                               en el BUFFER de recepción.
                             Prepara reconocimiento.
                             Pone Bandera de reconocimiento.
                          No:
                             Retorno de interrupción.
```

Rutina **LTINT:**

Esta rutina inicia en la dirección 2028 H, a la cual se salta, cada vez que un mensaje ha sido transmitido. Esta función tiene la siguiente forma:

```
La bandera de BUFFER
de transmisión está en lleno ?
     Si:
        El contador de
        "TIME OUT" ha
        finalizado ?
          Si:
             Pone bandera de BUFFER de transmisión en vacio.
             Pone contador de TIME OUT en cero.
          No:
             Incrementa contador en 1.
     No:
Hay que enviar fin
de mensaje ?  (bandera de fin de mensaje puesta)
     Si:
        Envia "MEND"
        Quita bandera de fin de mensaje.
     No:
Retorno de interrupción.
```

2- Rutinas de librería del módulo "maestro".

Aquí utilizamos el área de memoria: "BUFFER de transmisión", donde se almacena el mensaje de permiso a transmitir, previamente semi-preparado, con el fin de agilizar la comunicación. El mensaje se prepara de la siguiente forma:

Señal de "POLLING" (permiso):

dirección de inicio: 1E40 H = **"POLLNG"**

contenido: XXH,FEH,POLL,00H

Para la recepción y transmisión de los mensajes, se utilizan las mismas áreas de memoria externa, definidas en los módulos esclavos.

Las rutinas de librería del módulo maestro también son 2, la que responde a la llegada de un mensaje: **LRINT**, y la que responde a la terminación del envío de otro mensaje **TRINT**. Estas inician en las mismas direcciones que las rutinas de librería de los módulos esclavos; A continuación se muestra lo que cada una de ellas realiza.

Rutina **LRINT**:

```
Se recibió una señal
de fin de mensaje ?
     Si:
          Obtiene la dirección del siguiente
                módulo a rastrear.
          Envia señal de permiso ("POLLING")
                al módulo direccionado.
     No: Retorno de interrupción.
```

Rutina **TRINT**:

```
Apunta al siguiente módulo a rastrear.
Se acabaron los módulos
de la lista ?
     Si:
          Apunta al primero de la lista.
     No: Retorno de la interrupción.
```

Además se agregó una rutina temporizadora la cual envía la siguiente señal de "POLLING" si después de un determinado tiempo no se recibe la señal de "fin de mensaje", para esto se utiliza el "TIMER" 0, el cual se encuentra libre, y a través de su sobreflujo, se interrumpe al procesador y se ejecuta la rutina temporizadora la cual es muy parecida a la rutina LRINT.

Después de las rutinas de librería, en el módulo maestro, se tiene un programa que nos permite recibir del módulo controlador (programa monitor II corriendo en la microcomputadora PC), las direcciones que van a ser rastreadas ("POLLING"), a través de la función: 11 H = direcciones para "POLLING" ("ADDPOL"), colocando éstas en una lista, enviando después el primer mensaje de permiso ("POLLING"), y finalmente quedando en un ciclo sin fin, ya que todo el proceso se lleva a cabo por interrupción.

Las rutinas descritas en este capítulo se encuentran listadas en el apéndice D.

IMPORTANTE:

Las rutinas de librería de los módulos esclavos, así como la rutina del módulo maestro, utilizan el primer banco de registros del 8051 (banco 0: RS1,RS0 = 00H), para almacenar los apuntadores a datos, direcciones, banderas, contadores, y demas. Por lo tanto, el programador usuario del sistema, no debe utilizar el banco 0 ni el banco 1 (usado por las rutinas de trans./rec. de mensajes: "monitor I"), para otros usos que no sean los definidos.

C A P I T U L O 8

CAPITULO 8 - EJEMPLO DE APLICACION.

Para mostrar el uso y funcionamiento del sistema distribuido, se desarrolló un pequeño ejemplo de aplicación, utilizando unicamente 3 módulos controladores. Además se desarrollaron algunas rutinas de comunicación extras, muy simples y que dependiendo de la aplicación pueden ser mejoradas.

El ejemplo consiste en controlar un sistema que proporciona un vaso de agua a una temperatura dada.

Los problemas a resolver son:

- Trasladar el agua de un depósito de almacenaje hacia uno de calentamiento.
- Calentar el agua a la temperatura deseada.
- Vaciar el agua ya caliente en un vaso en la cantidad adecuada.

El sistema está formado por dos depósitos de agua: uno grande y otro pequeño:

En el depósito grande se encuentra colocada una bomba de agua la cual translada el agua hacia el depósito pequeño.

En el depósito pequeño (depósito de calentamiento) tenemos:
- Un flotador para detectar el nivel del agua.
- Un calentador.
- Un sensor de temperatura.
- Una válvula de paso que permite sacar el agua del depósito para vaciarla dentro de un vaso (salida).

Y en forma separada tenemos:

- Un teclado, el cual permite introducir el nivel de temperatura que se desea tenga el agua.
- Una pantalla de desplegados de 2 dígitos.
- Un motor de pasos el cual mueve un brazo mecánico para colocar el vaso en la posición adecuada para recibir el agua.
- Una alarma, la cual nos avisa si el proceso se ha iniciado o ha sido terminado.
- Una computadora PC, la cual sólo muestra como se está llevando a cabo la operación.

Los dispositivos a controlar se distribuyen de la siguiente manera:

Módulo 1:

- Flotador detector de nivel (F).
- Sensor de temperatura (T).
- Teclado (K).
- Pantalla de 2 digitos (D).

Módulo 2:

- Bomba de agua (B).
- Calentador (C).

Módulo 3:

- Válvula de paso (V).
- Motor de pasos (M).
- Alarma (A).

A continuación se muestra un esquema de este sistema:

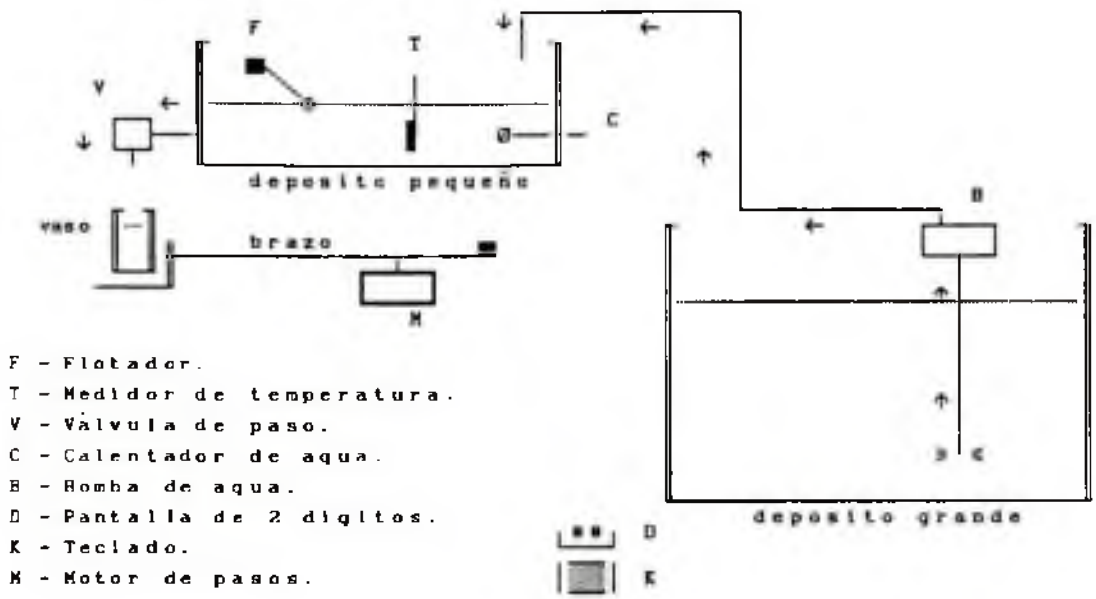


Figura 8.1 - Esquema del sistema del ejemplo de aplicación.

La distribución de los módulos dentro del sistema que deseamos controlar se realizó en base a que el sistema es pequeño y no es necesario colocar controladores a grandes distancias, por lo que:

El módulo 1 se encarga de controlar los dispositivos de detección de nivel, temperatura, la entrada de datos por el teclado y la salida de información por la pantalla de 2 digitos.

El módulo 2 controla los dispositivos de encendido/apagado de mayor voltaje como son: el calentador y la bomba.

El módulo 3 se encarga de: controlar la válvula de paso para el llenado del vaso, el motor de pasos para colocar el vaso en la posición y el momento adecuado, y hacer sonar la alarma en momentos en que suceden algunos eventos importantes.

A continuación veremos como cada uno de los módulos del sistema distribuido realizan su tarea, incluyendo la computadora PC (módulo PC) que se encarga de monitorear el estado del sistema.

Módulo 1.

Primeramente, despliega una señal de espera de información sobre la pantalla de dos dígitos, enviando este estado al módulo "PC", cada vez que recibe un dato a través del teclado lo muestra en la pantalla y envia éste al módulo "PC".

El teclado está formado por 4 teclas, "+", "-", "C" y "X", cuando se recibe un "+" o un "-" se incrementa o decrementa un valor inicial, con el fin de poder seleccionar el valor de temperatura que deseamos tenga el vaso de agua que se va a preparar.

Al introducir el dato "X" se inicia el proceso con la temperatura indicada, enviando la señal de inicio a los módulos "2", "3" y "PC", junto con el valor de temperatura que se desea.

Entonces, se entra en la etapa de detección de nivel de agua del depósito calentador y de su temperatura, manteniendo estos valores almacenados para cuando le sean solicitados por el módulo "2" o por el módulo "3", mostrando en la pantalla una señal de "proceso en ejecución".

Cuando se envian los valores de nivel y temperatura al módulo "2" también le son enviados al módulo "PC". El programa regresa a su estado inicial cuando recibe el mensaje de "terminación de proceso" dado por el módulo "3".

Módulo 2.

Este módulo al recibir la señal de inicio junto con el valor de temperatura que se desea, procedente del módulo "1", envía un mensaje de solicitud de datos (temperatura y nivel) al módulo "1", y de acuerdo al nivel leído y al nivel establecido se enciende y/o se apaga la bomba, hasta obtener el nivel adecuado, solicitando los valores de nivel y temperatura tantas veces como sea necesario.

Después de esto, se compara el valor de temperatura actual con el valor deseado, y también de acuerdo a esto, se enciende y/o se apaga el calentador, hasta obtener la temperatura deseada.

Una vez logrado lo anterior, se envía un mensaje de "agua lista" al módulo "3" así como al módulo "PC", regresando al estado inicial del programa.

Módulo 3.

Inicialmente este módulo coloca el brazo en una posición de espera sosteniendo el vaso que está vacío (responsabilidad del usuario debido a la sencillez del ejemplo de aplicación), y el programa entra a un ciclo de espera.

Cuando se recibe el mensaje de "agua lista", proveniente del módulo "2", se coloca el vaso en la posición adecuada para recibir el agua caliente que viene de la válvula cuando ésta se abra, la cual debe permanecer cerrada desde que se alimenta el circuito.

Después de esto, se solicita el valor de nivel de agua actual al módulo "1", para entonces abrir la válvula y entrar en un ciclo de solicitudes de niveles de agua, para saber cuando el vaso se ha llenado y entonces cerrar la válvula.

Una vez lleno el vaso y cerrada la válvula, el vaso es trasladado a un punto de entrega y finalmente se envía un mensaje de "terminación de proceso" al módulo "1" y se hace sonar la alarma.

Módulo PC.

El módulo "PC" unicamente recibirá información de los otros módulos, mostrando en forma gráfica una representación del proceso que se está llevando a cabo.

Todos los circuitos que se diseñaron y desarrollaron para interconectar los módulos controladores a los dispositivos a controlar (interfaces), no se describen en este trabajo, por su simplicidad y por no agrandar demasiado el texto.

En el apéndice E se muestran los listados de los programas de cada uno de los módulos (escritos en lenguaje ensamblador "8051"). En estos listados se pueden observar las rutinas de comunicación desarrolladas para este ejemplo ("SEND" y "RECV") como mencionamos al inicio de este capítulo.

En este ejemplo de aplicación, la velocidad no es un factor crítico, por lo que el sistema funciona bastante bien, además de estar compuesto de sólo 4 módulos (3 módulos 8051 y la computadora PC).

C A P I T U L O 9

CAPITULO 9 - CARACTERISTICAS DEL SISTEMA.
(Pruebas y resultados)

El sistema distribuido "8051" desarrollado en el presente trabajo está formado por una red de controladores "8051", en configuración maestro-escalvo.

Cada módulo controlador cuenta con los puertos o canales siguientes:

* Un canal de comunicación serie (comunicación asíncrona) utilizando un par torcido, permitiendo el uso del canal por un solo nodo o controlador a la vez ("HALF DUPLEX").

* Un canal o puerto paralelo de 8 bits, el cual se utiliza para enviar y/o recibir información a los dispositivos externos a controlar, con la capacidad de controlar cada uno de estos bits en forma individual, y

* Tres líneas de entrada/salida que pueden ser utilizadas como entradas de interrupción (INT0,INT1) o entrada al contador 0 (T0), o bien como salidas individuales.

Además cuenta con la facilidad de enviar voltaje de alimentación a cada uno de los módulos a través del mismo "BUS" o canal de comunicación. Este voltaje es utilizado por cada uno de los módulos controladores y por la interfaz RS485/RS232.

La velocidad de comunicación es de 10.416 K bits/seg., siendo ésta, la velocidad máxima lograda en el modo programado.

Es posible conectar hasta 28 nodos cubriendo una distancia máxima de 300 metros. El sistema desarrollado en el presente trabajo se formó con 10 nodos y 15 metros de "BUS" aproximadamente.

En la siguiente página se tiene una tabla con los resultados obtenidos al hacer mediciones sobre los controladores, y estos resultados concuerdan aproximadamente con los cálculos que se tienen a continuación.

Tenemos que: vel_com = 10,416 bits/seg.
con tramas de: 11 bits
y mensajes de: 5 tramas en promedio.

Por lo tanto tenemos: 55 bits por mensaje.
y contando otros 55 bits del mensaje de respuesta
tenemos: 110 bits/intercambio.

Así, estos dos mensajes forman un ciclo de "POLLING" dado por el nodo maestro tomandole un tiempo de:

110 bits/polling ÷ 10,416 bits/seg = 10.56 mseg.

" TIEMPOS DE POLLING "

Número de nodos.	Permisos dados por el maestro.	Permisos recibidos por un esclavo.	
Sin tráfico:			
6	11 mseg.	66 mseg.	
5	11 mseg.	55 mseg.	
4	11 mseg.	44 mseg.	
3	11 mseg.	33 mseg.	
2	11 mseg.	22 mseg.	
1	11 mseg.	11 mseg.	(91 Hz)
4	(con 1 sin funcionar)	89 mseg.	
3	''	78 mseg.	
2	''	67 mseg.	(14.9 Hz)
Con tráfico (1/byte de datos):			
4	15 mseg.	66 mseg.	
3	15 mseg.	48 mseg.	
2	15 mseg.	31 mseg.	
1	15 mseg.	15 mseg.	(63 Hz)

El tiempo de "espera de respuesta" (Time out) dado por el nodo maestro cada vez que envía una señal de permiso es: 15 mseg, permitiendo con esto enviar hasta 11 bytes de datos entre módulos.

C O N C L U S I O N E S

CONCLUSIONES.

El "sistema distribuído 8051" es sencillo y puede ser utilizado en una gran variedad de aplicaciones de control e instrumentación. Tiene algunas limitaciones en cuanto a velocidad, tanto de comunicación como de respuesta, pero si se planea correctamente su utilización en el proceso a controlar, en muchas de las aplicaciones, estas limitantes no afectarán su funcionamiento, aprovechando el bajo costo de este sistema.

Los resultados obtenidos en el ejemplo de aplicación, fueron satisfactorios, porque al utilizar las facilidades ofrecidas por el sistema distribuido "8051", como el colocar un mensaje para otro proceso dentro de un "BUFFER" y despreocuparse de como éste es transmitido, facilita bastante su manejo; y creo que las personas que tengan ligeros conocimientos de lenguaje ensamblador y de microprocesadores, podrán emplear el sistema para sus aplicaciones.

Ahora bien, si comparamos nuestro sistema distribuido "8051" con la línea de productos "iDCM" (módulos de control distribuido) desarrollados por Intel, compuesta por productos de "SOFTWARE" y "HARDWARE" [4], encontraremos que nuestro sistema es pequeño, demasiado simple, no cuenta con las capacidades de multitareas (nucleo), ni cuenta con convertidores Analógicos/Digitales y Digital/Analógicos incluidos en el mismo controlador, como los módulos "iRCB" de la línea de productos "iDCM" de Intel.

Con un poco de trabajo (tal vez otro trabajo de tesis), es posible incorporar un "nucleo de concurrencia" a nuestro sistema; y en lo que respecta a los convertidores A/D y D/A, el nuestro es más modular, ya que podemos contruir tarjetas convertidoras, las cuales podemos conectar a cualquiera de nuestros módulos en donde se vayan a utilizar, permitiéndole al usuario (o al cliente) modificar su sistema de una manera económica.

B I B L I O G R A F I A

BIBLIOGRAFIA.

1 - Distributed Systems and Computer Networks
Morris Sloman, Jeff Kramer
Prentice-Hall
1987

2 - Principles of Computer Communication Network Design
J. Seidler
Ellis Horwood Limited
1983

3 - MCS-51 Family of Single-Chip Microcomputers
USER'S MANUAL
INTEL
July 1981

4 - OEM Systems Handbook
Intel
1986

5 - 8-Bit Embedded Controller Handbook
Intel
1989

6 - C Programmer's Guide to Serial Communications
Campbell
Howard W. Sams
& Company

A P E N D I C E

APENDICE A.

DESCRIPCION DEL CONTROLADOR 8051.

La descripción del controlador 8051, está dividido en 4 partes:

a) Descripción de "PINES".
b) Arquitectura,
c) Memoria, y
d) Registros.

a) Descripción de "PINES".

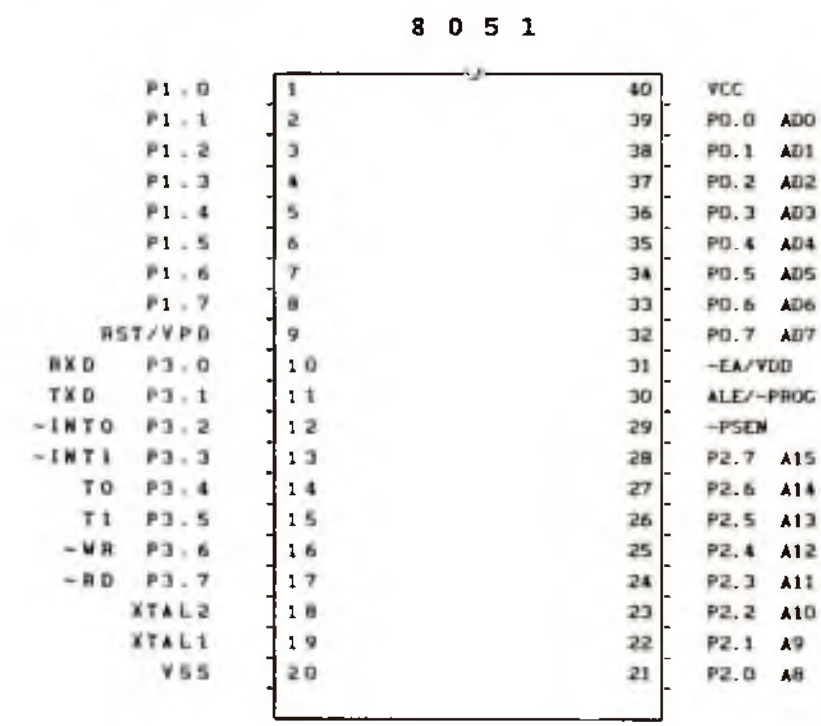


Figura A.1 - Distribución de "PINES".

Vss

Potencial circuito tierra.

Vcc

Alimentación de +5V durante: operación, programación y verificación.

Port 0

El puerto 0 es un puerto de entrada/salida bidireccional de 8 bits con drenaje abierto. También es el canal de direcciones de bajo orden multiplexadas con el canal de datos cuando se usa memoria externa. Este puerto es usado para la entrada y salida de datos durante la programación y verificación (8751). Su abanico de salida es igual a 2 cargas TTL.

Port 1

El puerto 1 es un puerto de E/S cuasi-bidireccional de 8 bits. Es usado para las direcciones de bajo orden durante la programación y verificación (8751). Su abanico de salida es de 1 carga TTL.

Port 2

El puerto 2 es un puerto de E/S cuasi-bidireccional de 8 bits. También emite los 8 bits de direcciones de alto orden cuando se accesa memoria externa. Es usado para las direcciones de más alto orden y las señales de control durante la programación y verificación (8751). Su abanico de salida es de 1 carga TTL.

Port 3

El puerto 3 es un puerto de E/S cuasi-bidireccional de 8 bits. También contiene las interrupciones, timers, puerto serie y los "PINS" RD y $\overline{WR}$ que son usadas por varias opciones. Su abanico de salida es de 1 carga TTL. Las funciones secundarias son asignadas a los "PINS" del puerto 3, como a continuación:

RXD/data (P3.0). Entrada de datos del puerto serie (asíncrono) o entrada/salida de datos (síncrono).

TXD/clock (P3.1). Salida de datos del puerto serie (asincrono) o salida de reloj (síncrono).

INT0 (P3.2). Entrada de interrupción 0 ó entrada de la compuerta de control para el contador 0.

INTI (P3.3). Entrada de interrupción 1 ó entrada de la compuerta de control para el contador 1.

T0 (P3.4). Entrada al contador 0.

T1 (P3.5). Entrada el contador 1.

$\overline{WR}$ (P3.6). Señal de control de escritura que sujeta el byte de datos del puerto 0 dentro de una memoria de datos externa.

$\overline{RD}$ (P3.7). Señal de control de lectura que habilita la memoria de datos externa al puerto 0.

RST/VPD

Una transición de bajo a alto en este "PIN" (aproximadamente de +3V) resetea el 8051. Si VPD es mantenido dentro de este rango (aproximadamente +5V), mientras Vcc cae debajo del rango, VPD proporcionará la alimentación a la RAM por un momento.

ALE/PROG

Proporciona la salida de habilitación para sujetar las direcciones en la memoria externa durante la operación normal. Recibe el pulso de entrada de programa durante la programación de la EPROM (8751).

PSEN

La salida de habilitación de programa almacenado es una señal de control que habilita la memoria de programa externa hacia el canal ("BUS") durante las operaciones normales de "FETCH".

EA/VDD

Cuando se mantiene en un nivel TTL alto esta terminal, el 8051 ejecuta instrucciones de la ROM/EPROM interna (8051/8751) siempre y cuando el PC (contador de programa) es menor que 4096. Cuando se mantiene en un nivel TTL bajo, el 8051 obtiene todas las instrucciones de la memoria externa de programa. Este "PIN" también recibe el voltaje de programación de 21V para la EPROM (8751).

XTAL1

Entrada hacia el oscilador de alta ganancia. Puede usarse un cristal o una fuente externa.

XTAL2

Salida del oscilador de alta ganancia. Es requerida cuando se utiliza un cristal.

b) Arquitectura.

El microcontrolador 8031 (microcontrolador de la familia "8051" utilizado en el presente trabajo) internamente está compuesto de una CPU 8051, una memoria de datos de lectura/escritura de 128 bytes, 32 líneas de entrada/salida distribuidas en cuatro puertos de 8 bits, 2 timer/contadores de 16 bits, una estructura de interrupciones de 5 fuentes con 2 niveles de prioridad, un puerto serie de entrada/salida: expansión de I/O o UART duplex completo, y oscilador y circuitos de reloj.

En la figura A.2 tenemos un diagrama a bloques del microcontrolador 8031.

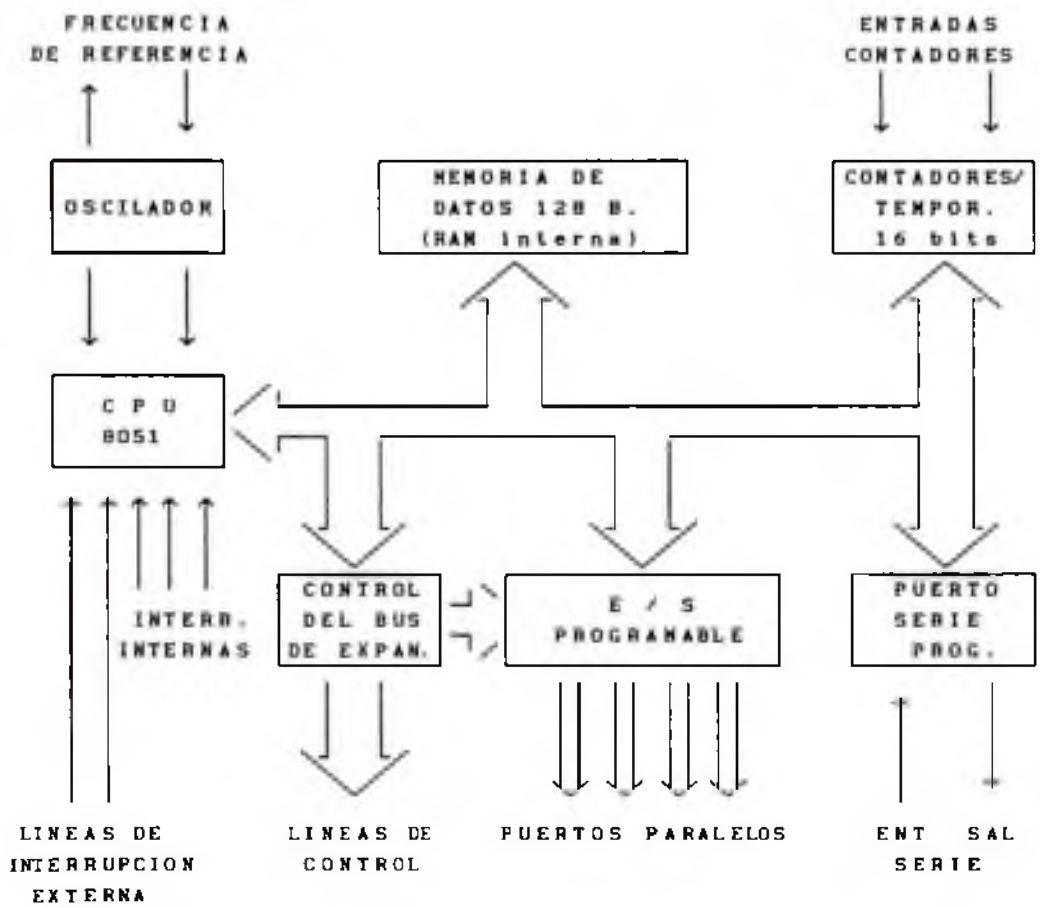


Figura A.2 - Diagrama a bloques del 8031.

El conjunto de instrucciones del "8051" es una ampliación del conjunto de instrucciones del MCS-48. Este fué ampliado para permitir la expansión de periféricos sobre el mismo "CHIP" y optimizar la velocidad de ejecución.

El conjunto de instrucciones está compuesto de:

49 instrucciones de un sólo byte,
45 instrucciones de 2 bytes, y
17 instrucciones de 3 bytes.

total: 111 instrucciones (ver apéndice B).

Cuando se usa un oscilador de 12 MHz, 64 instrucciones se ejecutan en 1 μseg. y 45 instrucciones se ejecutan en 2 μseg. Las instrucciones restantes (multiplicación y división) se ejecutan en sólo 4 μseg.

El "8051" tiene instrucciones que tratan las 32 líneas de entrada/salida como 32 bits individualmente direccionables, y como 4 puertos paralelos de 8 bits, direccionables por Puerto 0, 1, 2 y 3. Los puertos 0, 2 y 3 asumen otras funciones.

El puerto 0 proporciona el "BUS" de direcciones de bajo orden multiplexado con el "BUS" de datos, usados para expandir el "8051" con memorias estándar y periféricos.

El puerto 2 proporciona el "BUS" de direcciones de alto orden usado para expandir el "8051" con más de 256 bytes de memoria externa.

El puerto 3 puede ser configurado individualmente para proporcionar entradas de requerimientos de interrupción externas, entradas de contadores, entrada al receptor y salida del transmisor del puerto serie, y para generar las señales de control usadas para la lectura y escritura de memorias externas.

Para obtener información detallada sobre el microcontrolador "8051" se pueden consultar la bibliografía: [3,4 y 5].

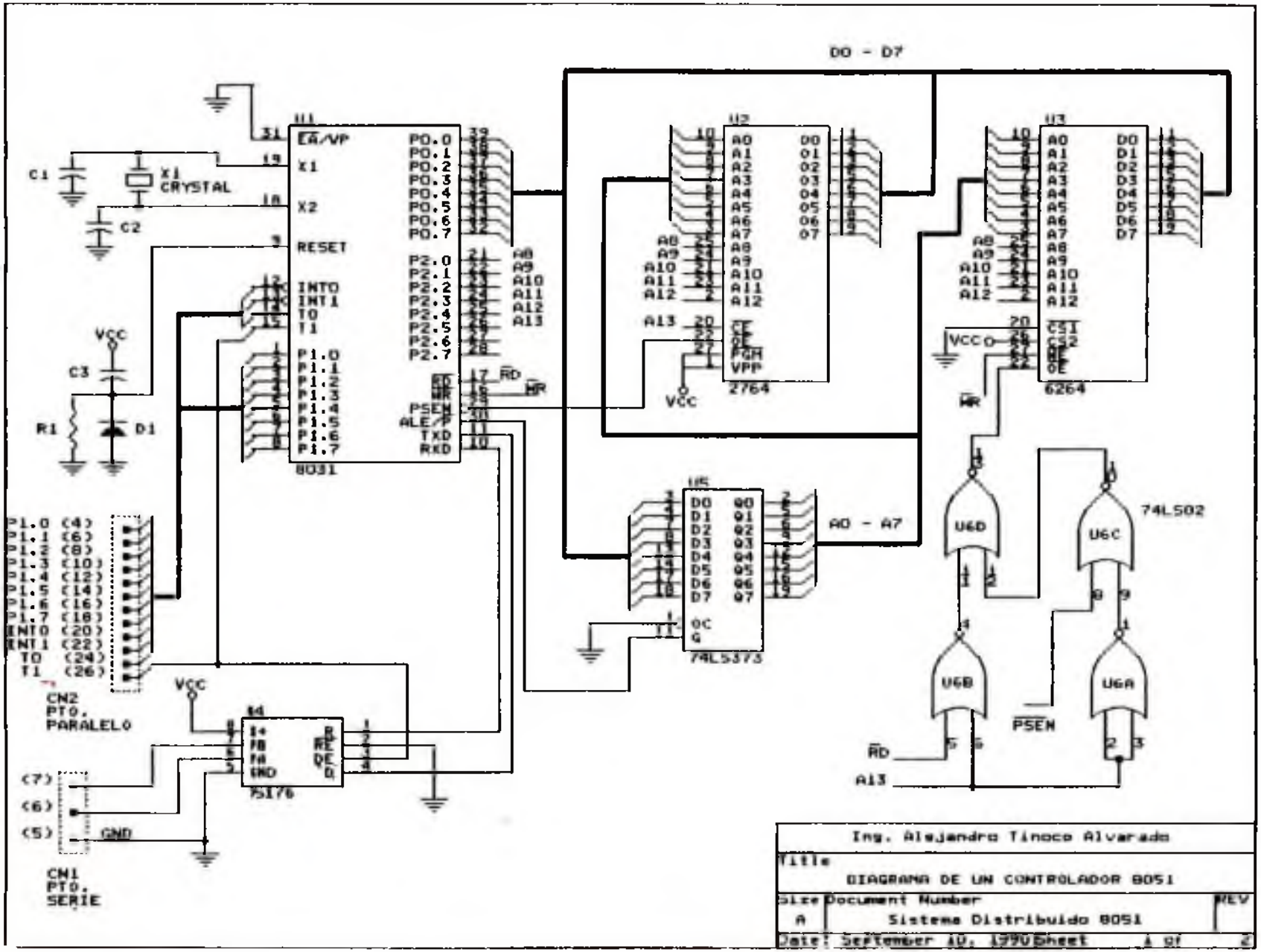
Además del C.I. 8031, el controlador "8051" está formado por los siguientes circuitos:

- Una memoria "EPROM" 2764 de 8 K bytes para el almacenamiento del código de programa,
- Una memoria "RAM" 6264 de 8K bytes, para el almacenamiento de datos o programa,
- Una compuerta 74LS373 (latch) para desmultiplexar las direcciones del puerto 0,
- Una compuerta 74LS02 (NOR) para la selección de memoria de datos/programa (6264), y
- Un "TRANSEIVER" 75176 para la comunicación serie en la norma RS-485.

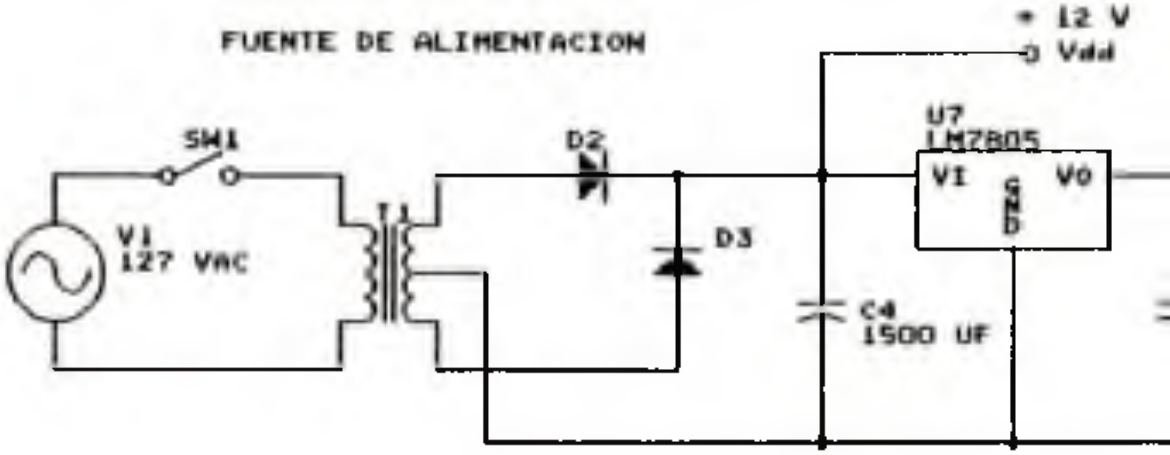
La compuerta 74LS02 (NOR) se utiliza para poder almacenar programas de aplicación en la memoria externa de datos RAM, haciendole creer al 8031 que esta accesando áreas de memoria de programa, haciendo accesible esta área en ambas formas, pero con direccionamientos diferentes, es decir, para accesarla como memoria de datos usamos direcciones a partir de la 0000H y como memoria de programa a partir de la 2000H, creando un efecto de espejo, como se puede observar en la figura A.3.

El circuito 75176 se utiliza para realizar la comunicación serie dentro de la norma "eléctrica": RS-485 recomendada por BITBUS de Intel, para aplicaciones de control industrial, por utilizar una línea balanceada, protegiendola del ruido del ambiente exterior. La comunicación a través de la línea usando esta interfaz es "HALF-DUPLEX", donde sólo se permite transmitir o recibir, pero no ambas al mismo tiempo.

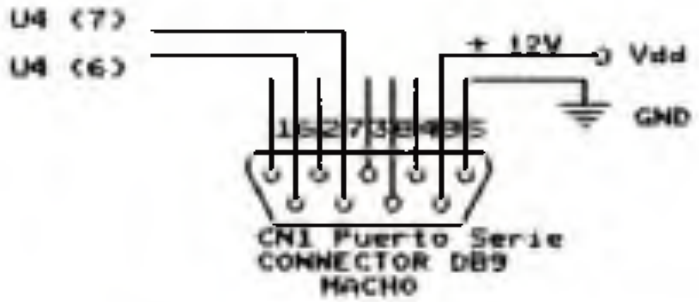
A continuación se muestra el diagrama del circuito de uno de estos controladores.

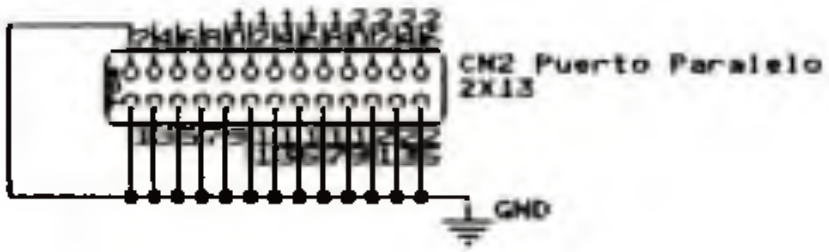

D0 - D7
U1
8031
EA/VP
X1
X2
RESET
INT0
INT1
T0
T1
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P0.0
P0.1
P0.2
P0.3
P0.4
P0.5
P0.6
P0.7
P2.0
P2.1
P2.2
P2.3
P2.4
P2.5
P2.6
P2.7
PSEN
ALE/P
TXD
RXD
C1
C2
X1
CRYSTAL
C3
R1
D1
VCC
U2
2764
U3
6264
U5
74LS373
A0 - A7
74LS02
U6D
U6C
U6B
U6A
PSEN
RD
A13
CN2
PTO.
PARALELO
CN1
PTO.
SERIE
GND
U4
75176
Ing. Alejandro Tinoco Alvarado
Title
DIAGRAMA DE UN CONTROLADOR 8051
Size
A
Document Number
Sistema Distribuido 8051
REV
Date: September 10, 1990
Sheet
1 of
2

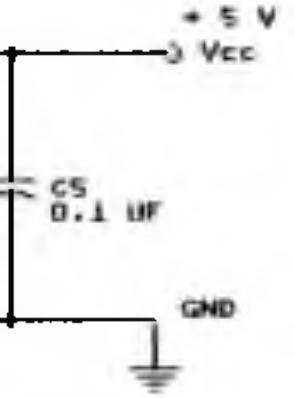
FUENTE DE ALIMENTACION



DISTRIBUCION DE PINES
DE LOS CONECTORES:







LISTA DE DISPOSITIVOS:

C1 = 10 PF
C2 = 10 PF
C3 = 10 UF - 10V
C4 = 1500 UF - 16V
C5 = 0.1 UF
R1 = 1.2 KOHMS
D1 = 1N914
D2 = 1N4004
D3 = 1N4004
X1 = 12 MHZ
T1 = 127 - 9+9 VCA
SW1 = 1P1T

TABLA DE ALIMENTACION

DISPOSITIVO	PIN Vcc	PIN GND
U1 = 8031	40	20
U2 = 2764	28	14
U3 = 6264	28	14
U4 = 75176	8	5
U5 = 74LS373	20	10
U6 = 74LS02	14	7
U7 = LM7805	-	-

Ing. Alejandro Tinoco Alvarado	
Title DISPOSITIVOS, FUENTE Y TABLA DE ALIMENTACION	
Size A	Document Number Sistema Distribuido 8051
Date: September 10, 1990	Sheet 2 of 2

c) **Memoria.**

El C.I. 8031 maneja operandos de 4 espacios de memoria, estos son: 64 K bytes de memoria externa de programa, 64 K bytes de memoria externa de datos, 384 bytes de memoria de datos interna y espacios para el contador de programa de 16 bits.

El espacio de direccionamiento de datos interna se divide además, en 256 bytes de datos internos RAM y 128 bytes de registros de funciones especiales (SFR), 4 bancos seleccionables de 8 registros cada uno, 128 bits direccionables, y la pila residente en la RAM de datos internos, cuya profundidad está limitada sólo por el espacio disponible, y estas localidades estan determinadas por los 8 bits del apuntador de pila SP.

La figura A.3 muestra en forma general la distribución de la memoria manejada por el microcontrolador.

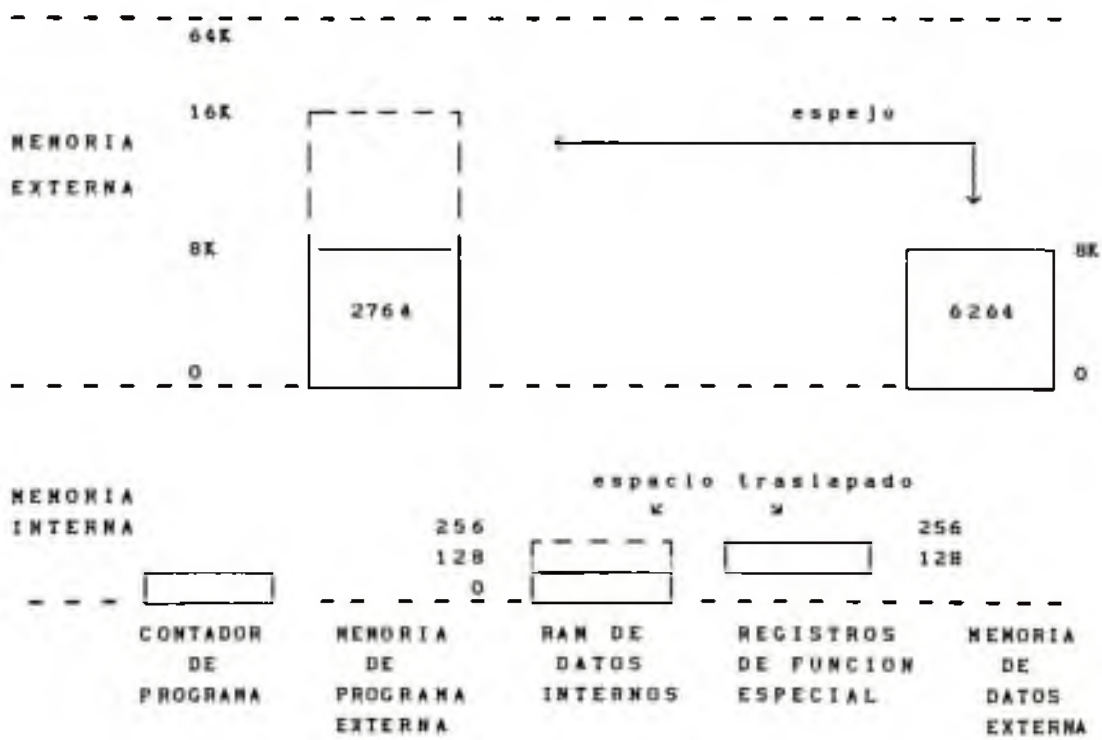


Figura A.3 - Mapa de memoria del microcontrolador 8051.

De los 4 bancos de registros con que cuenta el microcontrolador "8051", sólo uno es seleccionado a la vez, y accesado a través de los símbolos terminales del lenguaje ensamblador:

R0 R1 R2 R3 R4 R5 R6 R7

La selección del banco sobre el cual se está trabajando se realiza a través de los bits RS1 y RS0 del registro PSW, como se verá en la parte de "registros".

Los vectores de interrupción son direcciones de localidades de memoria, a la cual salta la ejecución del programa cuando una interrupción es reconocida; en estas direcciones se inicia la rutina de servicio que atiende la interrupción, previamente habilitada.

En el 8031, las interrupciones son provocadas por 5 fuentes diferentes. La tabla siguiente muestra las fuentes que las provocan y el vector de interrupción asociado a ellas.

Fuente interrupción	Dirección de inicio de la rutina de servicio.
Reestablecimiento ("RESET")	0000 H
Externa 0 (INT0)	0003 H
Temporizador/Contador 0 (T0)	000B H
Externa 1 (INT1)	0013 H
Temporizador/Contador 1 (T1)	001B H
Puerto serie (TI/RI)	0023 H

d) Registros.

Todos los registros excepto el contador de programa y los 4 bancos de registros, residen en el espacio de direcciones de los Registros de Función Especial (SFR: 21 registros).

Estos registros mapeados en memoria, incluyen registros aritméticos, apuntadores, puertos de I/O, registros para el sistema de interrupciones, timers, y el canal serie. Estos se muestran en la tabla siguiente:

REGISTRO	DIRECCION	FUNCION
P0	80H	Puerto 0
SP	81H	Apuntador de pila
DPL	82H	Apuntador de datos (low)
DPH	83H	Apuntador de datos (high)
PCON	87H	Reg. de control de potencia
TCON	88H	Control de timers
TMOD	89H	Reg. Modo de timer
TL0	8AH	Timer 0 L-byte
TL1	8BH	Timer 1 L-byte
TH0	8CH	Timer 0 H-byte
TH1	8DH	Timer 1 H-byte
P1	90H	Puerto 1
SCON	98H	Reg. cntr. pto. serie
SBUF	99H	Buffer datos pto. serie
P2	A0H	Puerto 2
IE	A8H	Reg. habil. interrupciones
P3	B0H	Puerto 3
IP	B8H	Reg. prior. interrupciones
PSW	D0H	Palabra de estado de prg.
ACC	E0H	Acumulador
B	F0H	Registro B

Algunos de los registros anteriores se utilizan para programar, habilitar y configurar los diferentes puertos, timers, contadores, etc., veremos a continuación con más detalle algunos de estos registros.

TCON - Registro de control/estado de los Timers/contadores.

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1 - Bandera de sobreflujo del timer 1, puesta por "HARDWARE" cuando hay sobreflujo en el timer/contador 1, limpiado por "SOFTWARE" cuando la interrupción es procesada.

TR1 - Bit de control de timer 1 corriendo., puesta/limpiado por "SOFTWARE" para encender/apagar el timer/contador 1.

TF0 y TR0 - Igual que los dos anteriores pero para el timer/contador 0.

IE1 - Bandera de interrupción 1 por flanco, puesta por "HARDWARE" cuando un flanco externo de interrupción es detectado, limpiado cuando la interrupción es procesada.

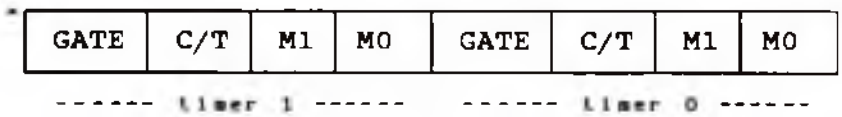
IT1 - Bit de control de tipo de interrupción 1, puesto/limpiado por software para especificar el disparo externo de interrupción por flanco de bajada o por nivel bajo.

IE0 y IT0 - Igual que las dos anteriores pero para la interrupción 0.

EJEMPLO:

```
MOV  TCON,#00010100B      ; Timer 1 encendido,
                          ; Interrupción 1 (ext) por flanco.
```

TMOD - Registro de modo de los timers/contadores.



GATE - Compuerta de control, cuando está puesta, el timer/contador "x" es habilitado sólo mientras el pin "INTx" esta en alto y el bit de control "TRx" esté puesto. Cuando es limpiado, el timer/contador está habilitado siempre que el bit de control esté puesto.

C/T - Selector de timer o contador, limpiado para la operación del timer, puesto para operación de contador.

M1 M0 - Modo de operación.

0 0 - MCS-48 timer, "TLx" sirve como un preescalador de 5 bits.

0 1 - 16 bit timer/contador, "THx" y "TLx" están en cascada, no hay preescalador.

1 0 - Timer/contador de auto-recarga de 8 bits, "THx" mantiene un valor el cual será recargado en "TLx" cada vez que éste se sobrefluye.

1 1 - (timer 0)

TL0 es un timer/contador de 8 bits, controlado por los bits de control estandar del timer 0.

TH0 es un timer de 8 bits controlado sólo por los bits de control del timer 1.

1 1 - (timer 1)

El timer/contador 1 es detenido.

EJEMPLO:

```
MOV  TMOD,#00000101B      ; Selecciona contador 0 de 16 bits
                          ; habilitado por TR0.
```

SCON - Registro de control/estado del puerto serie.

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0,SM1 - Bits de control 0 y 1 para programar el modo del puerto serie, puesto/limpiado por "SOFTWARE" (ver nota).

SM2 - Bit de control 2 del modo del puerto serie, puesto por software para deshabilitar la recepción de tramas en las cuales el bit 8 es cero.

REN - Bit de control de habilitación de recepción, puesto/limpiado por software para habilitar/deshabilitar la recepción de datos serie.

TB8 - Bit 8 transmitido, puesto/limpiado por "SOFTWARE" para determinar el estado del noveno bit transmitido en el modo UART de 9 bits.

RB8 - Bit 8 recibido, puesto/limpiado por "HARDWARE" para indicar el estado del noveno bit de datos recibido.

TI - Bandera de interrupción de transmisión, puesto por "HARDWARE" cuando un byte es transmitido, limpiado por "SOFTWARE" después del servicio.

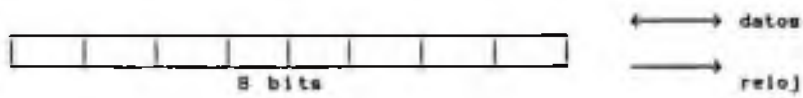
RI - Bandera de interrupción de recepción, puesto por "HARDWARE" cuando un byte es recibido, limpiado por "SOFTWARE" después del servicio.

Nota: el estado de (SM0,SM1) selecciona:

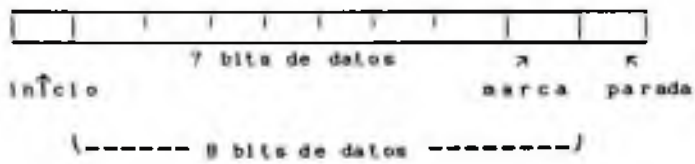
modo 0 (0,0) - Registro de corrimiento para expansión de E/S.
modo 1 (0,1) - UART 8-bits con velocidad de datos variable.
modo 2 (1,0) - UART 9-bits con velocidad de datos fija.
modo 3 (1,1) - UART 9-bits con velocidad de datos variable.

Los formatos de las tramas que se transmiten o reciben de acuerdo al modo programado, son las siguientes:

Modo 0: Expansión de entrada/salida, usando la frecuencia del oscilador de cristal.



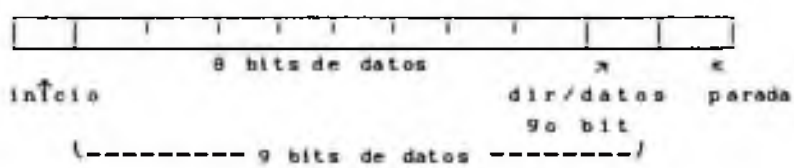
Modo 1: UART 8 bits con velocidad variable, utilizando el sobre flujo del TIMER 1.



Modo 2 y 3: UART de 9 bits con:

velocidad fija - modo 2 usando la frecuencia del oscilador de cristal.

velocidad variable - modo 3 usando el sobre flujo del TIMER 1.



En estos 2 modos se tiene el control del noveno bit a través del bit TB8 del registro SCON.

EJEMPLO:

```
MOV  SCON,#01010000B      ; UART 8-bits usando Timer 1 y
                          ; habilita recepción.
```

IE - Registro de habilitación de interrupciones.

EA	-	-	ES	ET1	EX1	ET0	EX0

EA - Bit de control de habilitación de toda interrupción, limpiada por "SOFTWARE" para deshabilitar todas las interrupciones, independientemente del estado de los otros bits.

ES - Bit de control de habilitación del puerto serie, puesto/limpiado por "SOFTWARE" para habilitar/dehabilitar interrupciones por las banderas TI o RI.

ET1 - Bit de control de habilitación del timer 1, puesto/limpiado por "SOFTWARE" para habilitar/deshabilitar interrupciones debidas al timer contador 1.

EX1 - Bit de control de habilitación de inter. 1 externa, puesto/limpiado por "SOFTWARE" para habilitar/deshabilitar interrupciones de INT 1.

ET0 y EX0 - Bits de control iguales a ET1 y a EX1 pero para el timer 0 y la entrada INT 0 respectivamente.

EJEMPLO:

```
MOV  IE,#10001001B      ; Habilita interrupciones por
                        ; el timer 1 y por entrada INT0.
```

La mayoria de los registros de función especial son también accesados a través de operaciones de bit como:

```
SETB  bit
CLR   bit
```

P3 - Funciones alternas de I/O del puerto 3.

RD	WR	T1	T0	INT1	INT0	TXD	RXD

RD - Salida de control de lectura de datos, activa en bajo, generada por hardware cuando un dato en memoria externa es leído.

WR - Salida de control de escritura de datos, activa en bajo, generada por "HARDWARE" cuando un dato en memoria externa es escrito.

T1 - Entrada externa o pin de prueba del timer/contador 1.

T0 - Entrada externa o pin de prueba del timer/contador 0.

INT1 - Pin de entrada de interrupción 1, disparado por nivel bajo o por flanco de bajada.

INT0 - Pin de entrada de interrupción 0, disparado por nivel bajo o por flanco de bajada.

TXD - Pin de transmisión de datos para el puerto serie en modo UART. Salida de reloj en modo de registro de corrimiento.

RXD - Pin de recepción de datos para el puerto serie en modo UART. Pin de datos de entrada/salida en modo de registro de corrimiento.

EJEMPLO:

```
RWAIT:   JNB  RI,RWAIT      ; Permanece en el ciclo
         CLR  RI            ; hasta que un byte
                            ; sea recibido.
```

PSW - Palabra de estado del programa.

CY	AC	FO	RS1	RS0	OV	-	P

CY - Bandera de acarreo.
AC - Bandera de acarreo auxiliar.
FO - Bandera 0, disponible al usuario para propósito general.
RS1 - Registro selector de banco bit 1. (ver sig. nota)
RS0 - Registro selector de banco bit 0. (ver sig. nota)
OV - Bandera de sobreflujo.
\- - Bandera definible por el usuario.
P - Bandera de paridad. Puesta/limpiada por "HARDWARE" en cada ciclo de instrucción para indicar un número par o impar de bits '1' en el acumulador.

Nota:

A través de los registros RS1 y RS0 es posible seleccionar el banco de registros sobre el cual se está trabajando. Los registros contenidos en cada banco son 8, y se accesa a éstos a través de los símbolos: **R0, R1, R2, R3, R4, R5, R6 y R7** del lenguaje ensamblador 8051. (ver apéndice B)

Para seleccionar el banco de trabajo utilizamos la tabla siguiente:

RS1	RS0	Banco:	Direccion:
0	0	0	00H - 07H
0	1	1	08H - 0FH
1	0	2	10H - 17H
1	1	3	18H - 1FH

Para obtener mayor información de estos y otros registros, se puede consultar el manual del microcomputador "8051". [3,5]

APENDICE B.

CONJUNTO DE INSTRUCCIONES DEL MCS-51.

OPERADORES ARITMETICOS.

Mnemónico:		Descripción:
ADD	A,Rn	Suma el registro al Aumulador.
ADD	A,directo	Suma el byte direc. al Acumulador.
ADD	A,@Ri	Suma RAM indirec. al Acumulador.
ADD	A,#dato	Suma el dato inmed. al Acumulador.
ADDC	A,Rn	Suma el reg. al Ac. con acarreo.
ADDC	A,directo	Suma byte dir. al Ac. con acarreo.
ADDC	A,@Ri	Suma RAM ind. al Ac con acarreo.
ADDC	A,#dato	Suma dato inmed. al Ac. c/acarreo.
SUBB	A,Rn	Substrae reg. del Ac. c/prestamo.
SUBB	A,directo	Subs. byte dir. del Ac. c/prest.
SUBB	A,@Ri	Subs. RAM ind. del Ac. c/prest.
SUBB	A,#dato	Subs. dato inmed. del Ac c/prest.
INC	A	Incrementa el Acumulador.
INC	Rn	Incrementa el registro.
INC	directo	Incrementa el byte directamente.
INC	@Ri	Incrementa RAM indirectamente.
DEC	A	Decrementa el Acumulador.
DEC	Rn	Decrementa el registro.
DEC	directo	Decrementa el byte directamente.
DEC	@Ri	Decrementa RAM indirectamente.
INC	DPTR	Incrementa el apuntador a datos.
MUL	AB	Multiplica A y B.
DIV	AB	Divide A por B.
DA	A	Ajuste decimal del Acumulador.

OPERACIONES LOGICAS.

Mnemónico:		Descripción:
ANL	A,Rn	AND registro a Acumulador.
ANL	A,directo	AND byte dir. a Acumulador.
ANL	A,@Ri	AND RAM indir. a Acumulador.
ANL	A,#dato	AND dato inmed. a Acumulador.
ANL	directo,A	AND Acumulador a byte dir.
ANL	directo,#dato	AND dato inmed. a byte dir.
ORL	A,Rn	OR registro a Acumulador.
ORL	A,directo	OR byte dir. a Acumulador.
ORL	A,@Ri	OR RAM indir. a Acumulador.
ORL	A,#dato	OR dato inmed. a Acumulador.
ORL	directo,A	OR Acumulador a byte dir.
ORL	directo,#dato	OR dato inmed. a byte dir.
XRL	A,Rn	OR excl. registro a Acumulador.
XRL	A,directo	OR excl. byte dir. a Acumulador.
XRL	A,@Ri	OR excl. RAM indir. a Acumulador.
XRL	A,#dato	OR excl. dato inmed. a Acumulador.
XRL	directo,A	OR excl. Acumulador a byte dir.
XRL	directo,#dato	OR excl. dato inmed. a byte dir.
CLR	A	Limpia el Acumulador.
CPL	A	Complemento de Acumulador.
RL	A	Rota a la izq. el Acumulador.
RLC	A	Rota izq. Ac. a través del C. F.
RR	A	Rota a la der. el Acumulador.
RRC	A	Rota der. Ac. a través del C. F.
SWAP	A	Intercambia nibbles del Ac.

TRANSFERENCIA DE DATOS.

Mnemónico:		Descripción:
MOV	A,Rn	Mueve el registro al Acumulador.
MOV	A,directo	Mueve el byte dir. al Acumulador.
MOV	A,@Ri	Mueve RAM ind. al Acumulador.
MOV	A,#dato	Mueve dato inmed. al Acumulador.

TRANSFERENCIA DE DATOS (continuación).

Mnemónico:		Descripción:
MOV	Rn,A	Mueve el Acumulador al registro.
MOV	Rn,directo	Mueve el byte dir. al registro.
MOV	Rn,#dato	Mueve el dato inmed. al registro.
MOV	directo,A	Mueve el Acumulador al byte dir.
MOV	directo,Rn	Mueve el registro al byte dir.
MOV	direc,direc	Mueve el byte dir. al byte dir.
MOV	directo,@Ri	Mueve RAM indir. al byte dir.
MOV	directo,#dato	Mueve el dato inmed. al byte dir.
MOV	@Ri,A	Mueve el Acumulador a RAM indir.
MOV	@Ri,directo	Mueve el byte dir. a RAM indir.
MOV	@Ri,#dato	Mueve el dato inmed. a RAM indir.
MOV	DPTR,#dato16	Carga DPTR con constante de 16-bits.
MOVC	A,@A+DPTR	Mueve byte-código rel. a DPTR al A.
MOVC	A,@A+PC	Mueve byte-código rel. a PC al A.
MOVX	A,@Ri	Mueve RAM ext. (8bits-dir) a A.
MOVX	A,@DPTR	Mueve RAM ext. (16bits-dir) a A.
MOVX	@Ri,A	Mueve el A. a RAM ext. (8bits-dir).
MOVX	@DPTR,A	Mueve el A. a RAM ext. (16bits-dir).
PUSH	directo	Mete el byte dir. en la pila.
POP	directo	Saca el byte dir. de la pila.
XCH	A,Rn	Intercambia registro con el A.
XCH	A,directo	Intercambia byte dir. con el A.
XCH	A,@Ri	Intercambia RAM indir. con el A.
XCHD	A,@Ri	Interc. dig. - sign RAM ind. con A.

MANIPULACION DE VARIABLES BOOLEANAS.

Mnemónico:		Descripción:
CLR	C	Limpia la bandera de acarreo.
CLR	bit	Limpia el bit directamente.
SETB	C	Pone la bandera de acarreo.
SETB	bit	Pone el bit directamente.
CPL	C	Complementa bandera de acarreo.

MANIPULACION DE VARIABLES BOOLEANAS (continuación).

Mnemónico:	Descripción:
CPL bit	Complementa el bit directamente.
ANL C,bit	AND bit dir. a bandera de acarreo.
ANL C,/bit	AND compl. bit dir. a band. acarreo.
ORL C,bit	OR bit dir. a bandera de acarreo.
ORL C,/bit	OR compl. bit. dir. a band. acarreo.
MOV C,bit	Mueve el bit dir. a la band. acarreo.
MOV bit,C	Mueve la band. acarreo al bit dir.

CONTROL DE FLUJO DE PROGRAMA.

Mnemónico:	Descripción:
ACALL dir11	Llamada absoluta a subrrutina.
LCALL dir16	Llamada larga a subrrutina.
RET	Retorno de subrrutina.
RETI	Retorno de interrupción.
AJMP dir11	Salto absoluto.
LJMP dir16	Salto largo.
SJMP rel	Salto corto (relativo).
JMP @A+DPTR	Salto ind. relativo a DPTR.
JZ rel	Salta si el A. es cero.
JNZ rel	Salta si el A. no es cero.
JC rel	Salta si b. acarreo esta puesta.
JNC rel	Salta si b. acarreo no esta puesta.
JB bit,rel	Salta si el bit dir. esta puesto.
JNB bit,rel	Salta si el bit dir. no esta puesto.
JBC bit,rel	Salta si esta puesto el bit y limpia.
CJNE A,directo,rel	Compara d.byte con A y salta si son <>.
CJNE A,#dato,rel	Compara dato con A y salta si son <>.
CJNE Rn,#dato,rel	Compara dato con reg. y salta si <>.
CJNE @Ri,#dato,rel	Compara dato con ind. y salta si <>.
DJNZ Rn,rel	Dec. reg. y salta si no es cero.
DJNZ directo,rel	Dec. byte dir. y salta si no es cero.
NOP	No realiza ninguna operación.

Notas sobre los modos de direccionamiento:

Rn	Registros de trabajo R0 - R7.
directo	Localidades de la RAM interna, algún puerto de E/S, de control, o estado.
@Ri	Localidad de RAM interna indirectamente direccionada por los registros R0 ó R1.
#dato	Constante de 8-bits incluida en instrucción.
#dato16	Constante de 16-bits incluidas como instrucciones de 2 y 3 bytes.
bit	128 banderas de software, algún pin de E/S, bit de control o bit de estado.

Notas sobre modos de direccionamiento de programa:

dir16	Dirección destino para LCALL y LJMP, pudiendo ser algún lugar dentro de los 64-Kbytes del espacio de direcciones de la memoria de programa.
dir11	Dirección destino para ACALL y AJMP pudiendo estar dentro de la misma página de 2-Kbytes de la memoria de programa.
rel	SJMP y todos los saltos condicionales incluyen un "OFFSET" de 8-bits. El rango es de -127 a +127 bytes relativos al primer byte de la siguiente instrucción.

APENDICE C.

CONSTRUCCION DE LAS TARJETAS CONTROLADORAS.

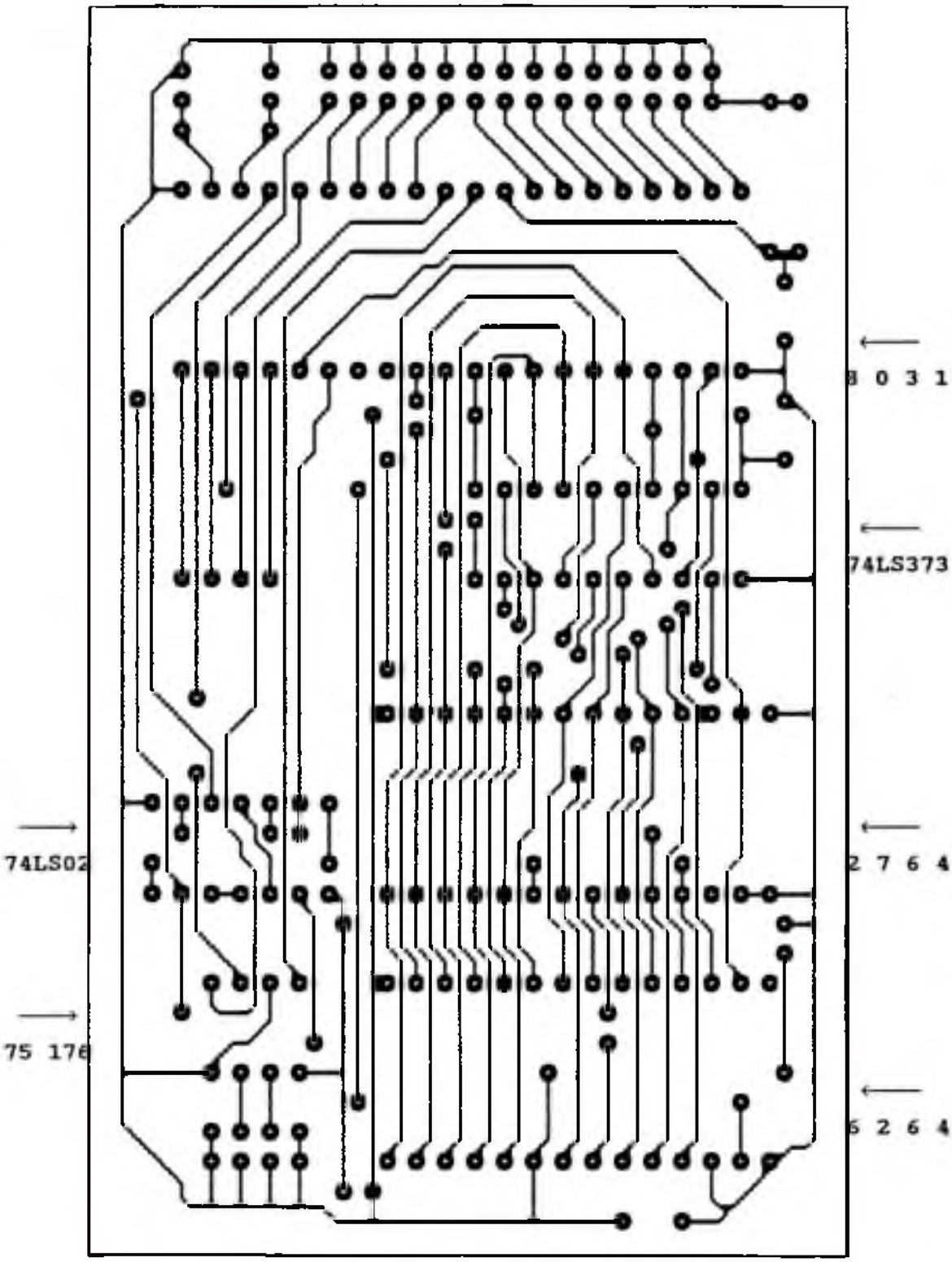
Para construir físicamente el diseño de los controladores realizado en el capítulo 5, primeramente se construyó el circuito con "WIRE-WRAP" sobre una tablilla perforada. Después de obtener buenos resultados con este diseño, se realizó el diseño del circuito impreso usando el paquete "SMARTWORK", realizando el estencil (Serigrafía) del circuito para poder obtener varios circuitos impresos.

La distribución de los circuitos integrados en la tarjeta controladora es similiar a como se encuentra dibujada en el diagrama del diseño, mostrado en el apéndice A.

La siguiente página muestra una copia del diseño del circuito impreso realizado con "SmartWork", compuesto unicamente de una cara, esto gracias al uso de sólo algunas conexiones extras ("JUMP") en puntos donde no se pudo encontrar ruta de conexión.

La tarjeta ya terminada, con los dispositivos montados, funcionó correctamente sin mayores problemas, pero su calidad no fué muy buena, en cuanto se refiere a presentación.

Con la ayuda de las autoridades de la Sección de Computación del Departamento de Ingeniería Eléctrica de este Instituto, se mandaron hacer 10 tarjetas controladoras, en una empresa dedicada a este tipo de trabajos, modificando un poco el diseño original, reduciendo su tamaño, incluyendo la fuente de alimentación dentro de la misma tarjeta y eliminando el circuito de "RESET" manual.



Circuito impreso diseñado con Smartwork.

(X 2)

Al final de este apendice se tiene una fotografía de uno de los controladores ya terminados, utilizados en el sistema distribuido "8051", con una etiqueta (6DH) que nos dice cual es su dirección dentro de la red.

Además del diseño de estas tarjetas, fué necesario diseñar y construir una interfaz que nos permitiera conectar una microcomputadora PC (IBM PC o compatible), a la red del sistema distribuido (RS-485 - recomendación de BITBUS), a través del puerto de comunicación serie RS-232C de la PC.

Esta interfaz convierte niveles RS-485 a niveles RS-232C y viceversa, utilizando:

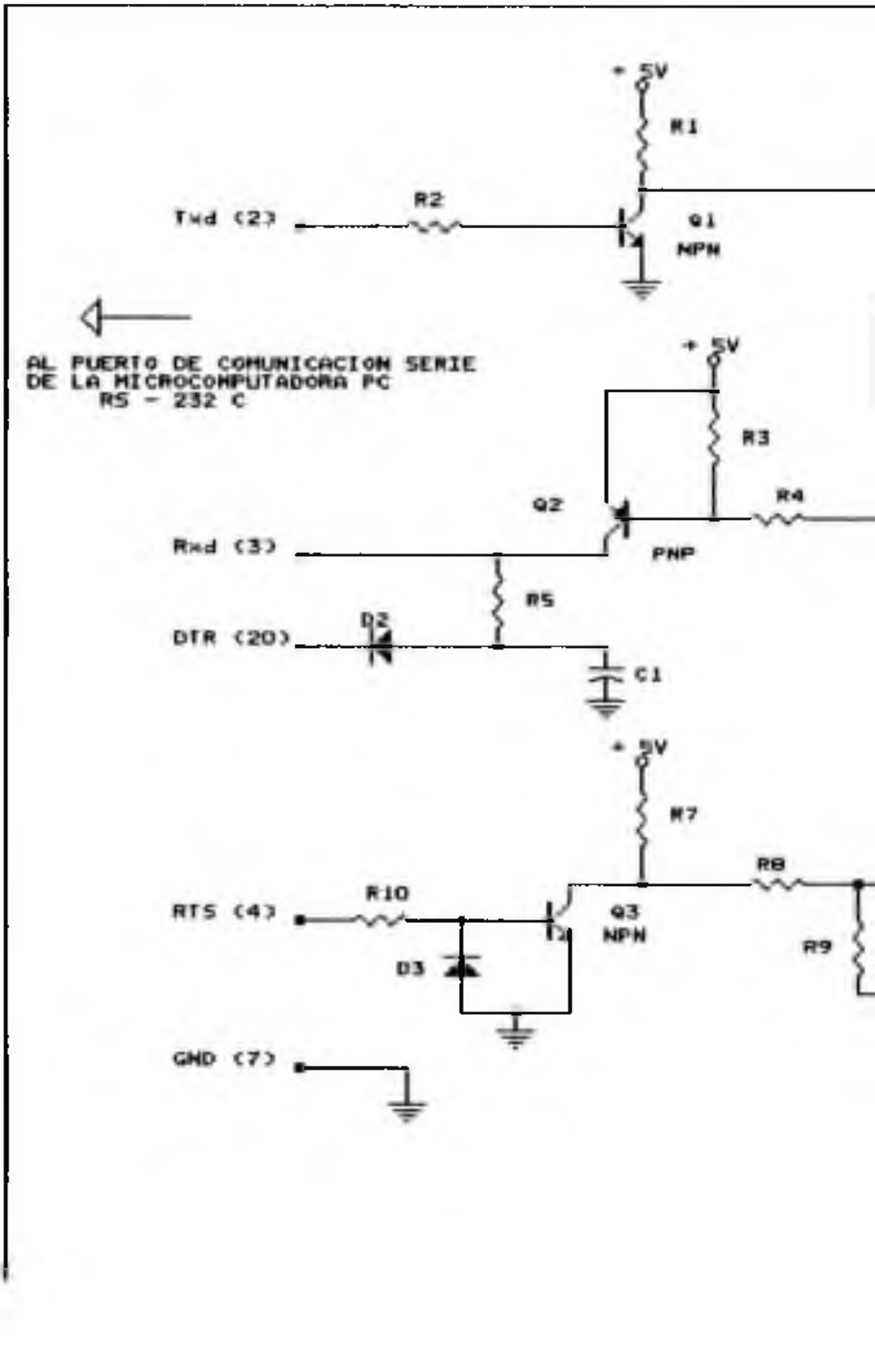
- El circuito 75176, el cual maneja niveles TTL por un extremo, y niveles RS-485 por el otro,

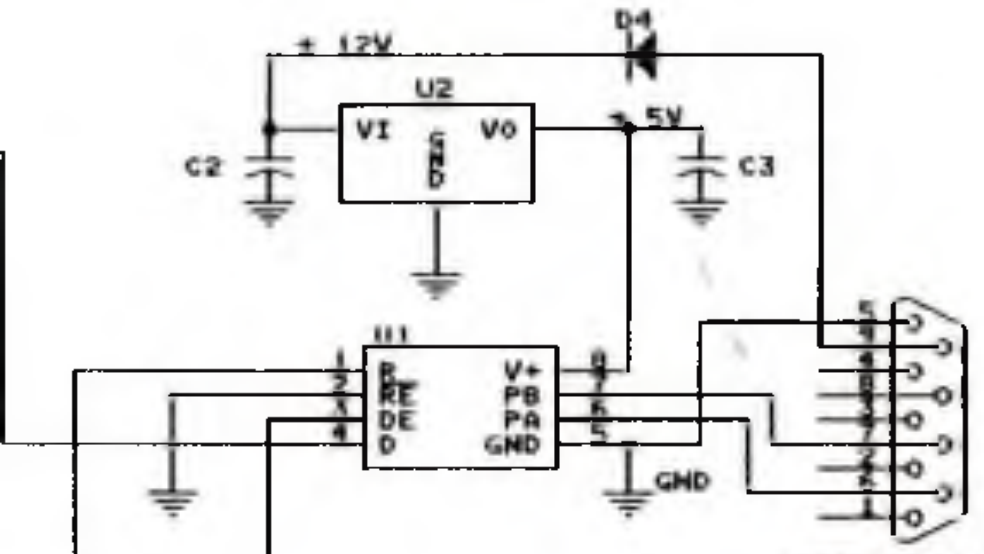
- Tres transistores para obtener unos inversores que manejan niveles de RS-232C (-12V - +12V), y los convierten a niveles TTL (0 - 5V), y

- Una compuerta NOT: 74LS04 para manejar la habilitación de transmisión del 75176; como se puede observar en el diagrama de la interfaz mostrada en la siguiente página.

De las señales de RS-232C utilizamos:

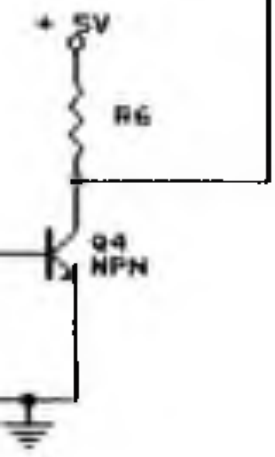
TXD - Para recibir los bytes de la PC.
RXD - Para enviar bytes a la PC.
DTR - Para tomar un voltaje de la PC que nos ayude a polarizar el transistor transmite a la PC.
RTS - Para habilitar el transmisor del transeiver 75176.


+ 5V
R1
R2
Txd (2)
Q1
NPN
AL PUERTO DE COMUNICACION SERIE
DE LA MICROCOMPUTADORA PC
RS - 232 C
+ 5V
R3
R4
Q2
Rxd (3)
PNP
R5
D2
DTR (20)
C1
+ 5V
R7
R8
R10
RTS (4)
Q3
NPN
R9
D3
GND (7)



Lista de dispositivos:

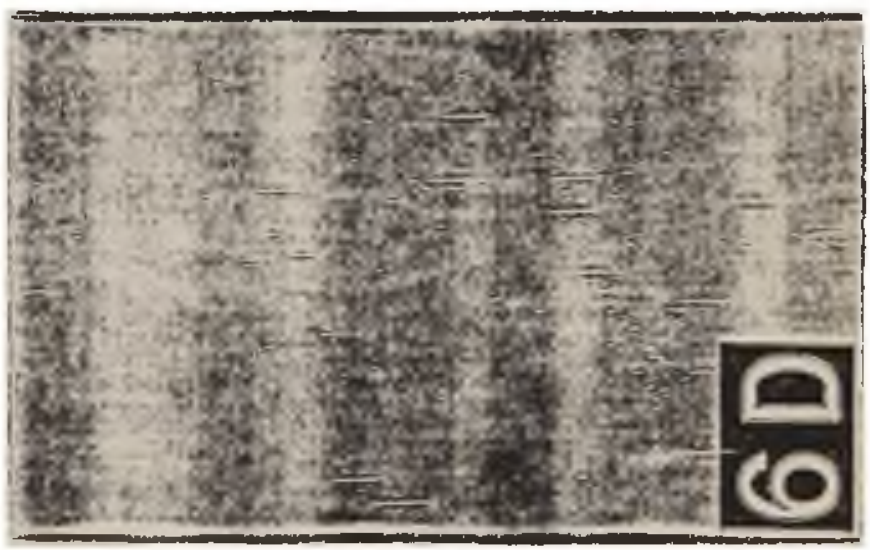
R1	=	2 K 2 ohms
R2	=	15 K "
R3	=	1 K "
R4	=	4 K 7 "
R5	=	4 K 7 "
R6	=	3 K 3 "
R7	=	2 K 2 "
R8	=	3 K 3 "
R9	=	1 K "
R10	=	15 K "
C1	=	4.7 uf
C2	=	0.1 uf
C3	=	0.1 uf
D1	=	1N 914
D2	=	1N 914
D3	=	1N 914
D4	=	1N 4004
Q1	=	2N 2222
Q2	=	2A 257
Q3	=	2N 2222
Q4	=	2N 2222
U1	=	75 176
U2	=	7805



Ing. Alejandro Tinoco Alvarado		
Title: Interfaz RS-232 / RS-485		
Size: A	Document Number: Sistema Distribuido 8051	REV
Date: September 10, 1990	Sheet 1 of 1	

El control de la interfaz es realizado por la PC que se conecta al sistema (Programa monitor II - capítulo 7).

Gracias a esta interfaz, es posible desarrollar programas en la microcomputadora PC y transferirlos a cada uno de los controladores del sistema distribuido, depurarlos, iniciar y controlar su ejecución.



Tarjeta controladora 8051 terminada.

APENDICE D.

LISTADOS DE LAS RUTINAS DE LIBRERIA.

```
;=================================================================
;
;                  SISTEMA DISTRIBUIDO 8051
;
;               (c) 1990   CINVESTAV del I.P.N.
;
;           Programa: Programa Maestro.
;           Archivo:  MAES.ASM
;           Función:  Se encarga de coordinar las comunicaciones
;                     mediante el sistema de POLLING (permiso).
;           Autor:    Ing. Alejandro Tinoco Alvarado
;           Asesor:   Dr. Jan Janecek Hyan
;           Fecha:    Julio  1990
;=================================================================
CTRADDR EQU   00H
MYADDR  EQU   0FEH
;
ACKN    EQU   0FH
POLL    EQU   10H
ADDPOL  EQU   11H
TREND   EQU   12H
DATOS   EQU   15H
TMOUTL  EQU   8CH
TMOUTH  EQU   3CH
;
MRECV   EQU   0030H
MSEND   EQU   0033H
;
CNT0    EQU   7EH
CNT1    EQU   7DH
CNT2    EQU   7CH

        ORG   1E40H                     ; Señal de POLLING
POLLNG  DB    01H,MYADDR,POLL,00H
        ;
        ORG   2000H
START:  JMP   MAIN
        ;
        ORG   200BH
        JMP   LTF0                      ; Servicio TMOUT
        ;
        ORG   2023H                     ; Mensaje recibido
        JMP   LRINT
        ;
```

```
        ORG   2028H                     ; Mensaje transmitido
        JMP   LTINT
        ;
;=======================================================================
;       Rutina de servicio a la llegada de un mensaje
;=======================================================================
        ORG   2030H
LRINT:
        PUSH  PSW
        CLR   RS0
        MOV   DPTR,#1D7CH
        MOVX  A,@DPTR
        CJNE  A,#TREND,LRINTE           ; Es fin de mensaje ?
        MOV   DPL,R4
        MOV   DPH,#1EH
        MOVX  A,@DPTR
        MOV   DPTR,#POLLNG              ; Envía siguiente POLLING
        MOVX  @DPTR,A
        CALL  MSEND
LRINTE:
        POP   PSW
        RET
        ;
;=======================================================================
;       Rutina de servicio de un mensaje transmitido
;=======================================================================
LTINT:
        PUSH  PSW
        CLR   RS0
        CLR   T1
        CLR   TB8
        SETB  REN
        INC   R4
        DJNZ  R3,LTINTE
        MOV   A,R6
        MOV   R3,A
        MOV   R4,#0
LTINTE:
        CLR   TR0
        MOV   TL0,#TMOUTL
        MOV   TH0,#TMOUTH
        SETB  TR0
        POP   PSW
        RET
        ;
;=======================================================================
;   Rutina de servicio a sobreflujo en el timer 0 (TMOUT)
;=======================================================================
LTF0:
        CLR   TR0
        CLR   RS0
        MOV   DPL,R4
        MOV   DPH,#1EH
        MOVX  A,@DPTR
        MOV   DPTR,#POLLNG              ; Envía siguiente POLLING
        MOVX  @DPTR,A
        CALL  MSEND
        CLR   TF0
        RETI
```

```
;=============================================================
;   Rutinas de envio y recepción de bytes.
;=============================================================
SEND:   MOV   SBUF,A
        MOV   R5,A
        ADD   A,7FH
        MOV   7FH,A
        MOV   A,R5
SEND0:  JNB   TI,SEND0
        CLR   TI
        RET
;----------------------------
RECV:   JNB   RI,RECV
        CLR   RI
        MOV   A,SBUF
        MOV   R5,A
        ADD   A,7FH
        MOV   7FH,A
        MOV   A,R5
        RET
;----------------------------
SHEAD:  MOV   A,#60H
SHEA0:  DJNZ  ACC,SHEA0
        SETB  T1
        CLR   REN
        SETB  TB8
        MOV   7FH,#00H
        MOV   A,#CTRADDR
        MOV   SBUF,A
        ADD   A,7FH
        MOV   7FH,A
SHEA1:  JNB   TI,SHEA1
        CLR   TI
        CLR   TB8
        MOV   A,#MYADDR
        MOV   SBUF,A
        ADD   A,7FH
        MOV   7FH,A
SHEA2:  JNB   TI,SHEA2
        CLR   TI
        RET
        ;
        ;
;=============================================================
;     PROGRAMA DE APLICACION        (MAESTRO)
;=============================================================
MAIN:
        MOV   PSW,#0
        CLR   EA
        MOV   A,#1                      ; Programación del timer 0
        ORL   A,TMOD
        MOV   TMOD,A
        CLR   TR0
        ;
        MOV   R3,#0                     ; Contador de POLLINGS
        MOV   R4,#0                     ; Apunt. bajo a dir. de POLLINC
        MOV   R5,#0                     ; Registro auxiliar
        MOV   R6,#0                     ; Número de POLLINGS
        ;
```

```
RWAIT:
        SETB   SM2
        SETB   REN
        CLR    T1
        MOV    7FH,#0
        CALL   RECV                          ; MyAddr
        CJNE   A,#MYADDR,RWAIT
        CLR    SM2
        CALL   RECV                          ; CntrlAddr
        CJNE   A,#CTRADDR,RWAIT
        CALL   RECV                          ; Código de función = ADDPOL
        CJNE   A,#ADDPOL,RWAIT
        CALL   RECV                          ; Longitud
        MOV    R6,A
        MOV    R3,A
        MOV    DPTR,#1E00H                   ; Area de direcciones de Pollin
RW1:
        JNB    RI,RW1
        CLR    RI
        MOV    A,SBUF
        MOVX   @DPTR,A
        ADD    A,7FH
        MOV    7FH,A
        INC    DPTR
        DJNZ   R3,RW1
        CALL   RECV
        MOV    A,#00H
        CJNE   A,7FH,RWAIT
        CALL   SHEAD
        MOV    A,#ACKN                       ; Envio de reconocimiento
        CALL   SEND
        MOV    A,#00H
        CALL   SEND
        MOV    A,7FH
        CPL    A
        INC    A
        CALL   SEND
        ;                                      Habilita recepción.
        CLR    T1
        SETB   SM2
        SETB   REN
        SETB   ES
        SETB   ET0
        SETB   EA
        ;                                  Preparación del primer POLLING
        MOV    A,R6
        MOV    R3,A
        MOV    R4,#0
        MOV    DPTR,#1E00H
        MOVX   A,@DPTR
        MOV    DPTR,#POLLNG
        MOVX   @DPTR,A
        CALL   MSEND                         ; Envía primer POLLING
        ;
LOOP:
        JMP    LOOP                          ; Ciclo infinito.
        ;
        END    START
```

```
;=============================================================
;
;                  SISTEMA DISTRIBUIDO 8051
;
;                (c) 1990   CINVESTAV del I.P.N.
;
;           Programa: Rutinas de libreria (esclavos).
;           Archivo:  LIBRA.ASM
;           Función:  Se encarga de realizar las comunicaciones
;                     mediante el sistema de POLLING (permiso)
;                     dentro de cada controlador 8051.
;           Autor:    Ing. Alejandro Tinoco Alvarado
;           Asesor:   Dr. Jan Janecek Hyan
;           Fecha:    Julio  1990
;=============================================================
MAESTRO EQU   0FEH
MYADDR  EQU   xxH
DSTADDR EQU   xxH
;
ACKN    EQU   0FH
POLL    EQU   10H
ADDPOL  EQU   11H
TREND   EQU   12H
DATOS   EQU   15H
TMOUT   EQU   07H
;
MRECV   EQU   0030H
MSEND   EQU   0033H
;
        ORG   1DF8H                     ; Señal de fin de mensaje
MEND    DB    MAESTRO,MYADDR,TREND,00H
        ;
        ORG   1DFCH                     ; Mensaje de reconocimiento
RECON   DB    MAESTRO,MYADDR,ACKN,00H
        ;
        ORG   1E40H                     ; Mensaje de información
MENSG   DB    DSTADDR,MYADDR,DATOS,01H
        ;
        ORG   2000H
START:  JMP   MAIN
        ;
        ORG   2023H                     ; Mensaje recibido
        JMP   LRINT
        ;
        ORG   2028H                     ; Mensaje transmitido
        JMP   LTINT
        ;
;-------------------------------------------------------------
;       Rutina de servicio a la llegada de un mensaje
;-------------------------------------------------------------
        ORG   2030H
LRINT:
        PUSH  PSW
        CLR   RS0
        MOV   DPTR,#1D7BH
        MOVX  A,@DPTR
        CJNE  A,#MAESTRO,LRINT1         ; Maestro ?
        INC   DPTR
```

```
        MOVX  A,@DPTR
        CJNE  A,#POLL,LRINT0          ; Polling ?
        CJNE  R0,#1,LRINT01           ; ACKFLAG = ON ?
        MOV   DPTR,#RECON
        CALL  MSEND                   ; Envia reconocimiento
        MOV   R7,#1
        MOV   R0,#0
LRINT0: JMP   LRINTE
LRINT01:
        CJNE  R6,#1,LRINT03           ; Buffer Tx = LLENO ?
        MOV   DPTR,#MENSG             ; Envia mensaje
        CALL  MSEND
        MOV   R7,#1
        JMP   LRINTE
LRINT03:                              ; Envio de fin de mensaje
        MOV   DPTR,#MEND
        CALL  MSEND
        MOV   R7,#0
        JMP   LRINTE
LRINT1:
        INC   DPTR
        MOVX  A,@DPTR
        CJNE  A,#ACKN,LRINT2          ; Reconocimiento ?
        CJNE  R6,#1,LRINT11           ; Buffer Tx = LLENO ?
        MOV   R6,#0
LRINT11:
        JMP   LRINTE
LRINT2:
        CJNE  R5,#0,LRINT11           ; Buffer Rx = VACIO ?
        INC   DPTR
        MOVX  A,@DPTR
        ADD   A,#4
        MOV   R1,A                    ; Longitud del mensaje
        MOV   R2,#7AH
        MOV   R3,#0
LRINT21:
        MOV   DPL,R2
        MOV   DPH,#1DH
        MOVX  A,@DPTR                 ; Guarda el mensaje
        MOV   DPL,R3
        MOV   DPH,#1EH
        MOVX  @DPTR,A
        INC   R2
        INC   R3
        DJNZ  R1,LRINT21
LRINT23:
        MOV   DPTR,#1D7BH             ; Prep. reconocimiento.
        MOVX  A,@DPTR
        MOV   DPTR,#RECON
        MOVX  @DPTR,A                 ; Destino
        MOV   R5,#1
        MOV   R0,#1
        JMP   LRINTE
        ;
LRINTE:
        POP   PSW
        RET
        ;
```

```
;-------------------------------------------------------------
;       Rutina de servicio de un mensaje transmitido
;-------------------------------------------------------------
LTINT:
        PUSH   PSW
        CLR    RS0
        CLR    T1
        CLR    TB8
        SETB   REN
        CJNE   R6,#1,LTINT01              ; Hay algo que transm.?
        INC    R4
        CJNE   R4,#TMOUT,LTINT01          ; TIME OUT ?
        MOV    R4,#0
        MOV    R6,#0
LTINT01:
        CJNE   R7,#1,LTINT02              ; Envio fin de mensaje ?
        MOV    DPTR,#MEND
        CALL   MSEND
        MOV    R7,#0
LTINT02:
        POP    PSW
        RET
        ;
;-------------------------------------------------------------
;     PROGRAMA DE APLICACION
;-------------------------------------------------------------
MAIN:
        MOV    PSW,#0
        ;
        CLR    T1
        SETB   ES
        SETB   EA
        SETB   SM2
        SETB   REN
        ;
        MOV    R0,#0                      ; Bandera Recon. apag.
        MOV    R1,#0                      ; Contador de datos
        MOV    R2,#0                      ; Apuntador Buff. llegada
        MOV    R3,#0                      ; Apuntador Buff. almacen.
        MOV    R4,#0                      ; Inicializa TMOUT
        MOV    R5,#0                      ; Buff-Rx   vacio
        MOV    R6,#0                      ; Buff-Tx   vacio
        MOV    R7,#0                      ; Bandera de fin de mensaje
        ;
        ;
        ;    PROGRAMA DE APLICACION DEL USUARIO
        ;
        ;
        END START
```

APENDICE E.

LISTADOS DE LOS PROGRAMAS DE APLICACION.

```
;=============================================================
;
;                    SISTEMA DISTRIBUIDO 8051
;                     Programas de aplicación
;               México (c) 1990   CINVESTAV del I.P.N.
;
;            Archivo:  APL6D.ASM
;            Programa: Módulo 1
;            Función:  Controla los sensores de nivel y
;                      temperatura del sistema de
;                      calentamiento de agua.
;            Autor:    Ing. Alejandro Tinoco Alvarado
;            Asesor:   Dr. Jan Janecek Hyan
;            Fecha:    Agosto  1990
;=============================================================
MAESTRO EQU   0FEH
MYADDR  EQU   6DH
DSTADDR EQU   05H
;
ACKN    EQU   0FH
POLL    EQU   10H
ADDPOL  EQU   11H
TREND   EQU   12H
DATOS   EQU   15H
;
MRECV   EQU   0030H
MSEND   EQU   0033H
;
TMOUT   EQU   10H
;
;   VARIABLES USADAS
CNT0    EQU   60H
CNT1    EQU   61H
CNT2    EQU   62H
TEM1    EQU   63H
TEM2    EQU   64H
TEM3    EQU   65H
DATO    EQU   66H
BCD     EQU   67H
RESI    EQU   68H
TMPE    EQU   69H
NIVEL   EQU   6AH
```

```
;  CONSTANTES USADAS
;
IN      EQU   02H                       ; Inicio
TM      EQU   04H                       ; Temperatura
NV      EQU   06H                       ; Nivel del agua
AL      EQU   08H                       ; Agua lista
TE      EQU   10H                       ; Terminé
DT      EQU   12H                       ; Dame temperatura
DN      EQU   14H                       ; Dame nivel
EP      EQU   16H                       ; Esperando
BM      EQU   18H                       ; Bombeando
CA      EQU   20H                       ; Calentando
LV      EQU   22H                       ; Llenando vaso
;
;   NODOS USADOS
;
M1      EQU   6DH
M2      EQU   12H
M3      EQU   25H
PC      EQU   05H
;
        ;                               ; dirección de datos en 7 segme
        ORG   1D00H
BCDSEG  DB    40H,79H,24H,30H,19H,12H,02H,78H,00H,10H,08H,03H,46H,21H,C
        ;
        ORG   1DF8H                     ; Señal de fin de mensaje
MEND    DB    MAESTRO,MYADDR,TREND,00H
        ;
        ORG   1DFCH                     ; Mensaje de reconocimiento
RECON   DB    MAESTRO,MYADDR,ACKN,00H
        ;
        ORG   1E40H                     ; Mensaje de información
MENSG   DB    DSTADDR,MYADDR,DATOS,01H
        ;
        ;
        ORG   2000H
START:  JMP   MAIN
        ;
        ORG   2023H                     ; Mensaje recibido
        JMP   LRINT
        ;
        ORG   2028H                     ; Mensaje transmitido
        JMP   LTINT
        ;
;---------------------------------------------------------------
;       Rutinas de librería
;---------------------------------------------------------------
        ORG   2030H
LRINT:
;
LTINT:
;
;
;             VER RUTINAS DE LIBRERIA (APENDICE D)
;
;
```

```
;-------------------------------------------------------------
;     PROGRAMA DE APLICACION       (MODULO 1)
;-------------------------------------------------------------
MAIN:
        MOV   PSW,#0
        CLR   T1
        SETB  ES
        SETB  EA
        SETB  SM2
        SETB  REN
        MOV   R0,#0                     ; Bandera Recon. apag.
        MOV   R1,#0                     ; Contador de datos
        MOV   R2,#0                     ; Apuntador Buff. llegada
        MOV   R3,#0                     ; Apuntador Buff. almacen.
        MOV   R4,#0                     ; Inicializa TMOUT
        MOV   R5,#0                     ; Buff-Rx   vacio
        MOV   R6,#0                     ; Buff-Tx   vacio
        MOV   R7,#0                     ; Bandera de fin de mensaje RQ.
        MOV   DPTR,#1E43H
        MOV   A,#02H                    ; Coloca long. de mgs. = 2
        MOVX  @DPTR,A
        CALL  CLEAR
LOOP:
        CALL  ESPDAT                    ; Recibe temp. deseada (A)
        MOV   DPTR,#1E45H
        MOVX  @DPTR,A
        MOV   B,#PC                     ; Envía señal de inicio
        MOV   A,#IN                     ; a todos los nodos del
        CALL  SEND                      ; sistema junto con la
        MOV   B,#M2                     ; temp. deseada.
        MOV   A,#IN
        CALL  SEND
        MOV   B,#M3
        MOV   A,#IN
        CALL  SEND
        CALL  SERV                      ; Entrega valores solicitados
        JMP   LOOP
        ;
;-----------------------------------------
;      Rutina recive datos del usuario
;-----------------------------------------
ESPDAT:
        MOV   DATO,#1CH
EP0:
        CALL  DISPLAY
        CALL  TECLA
        CJNE  A,#0FFH,EP1
        MOV   A,DATO
        MOV   DPTR,#1E45H
        MOVX  @DPTR,A
        MOV   B,#PC
        MOV   A,#EP
        CALL  SEND
        JMP   EP0
EP1:
        CJNE  A,#02H,EP2
        INC   DATO
```

```
        CALL    CONV
        JMP     EP0
EP2:
        CJNE    A,#04H,EP3
        DEC     DATO
        CALL    CONV
        JMP     EP0
EP3:
        CJNE    A,#06H,EP0
        MOV     A,DATO
        RET
        ;
;-----------------------------------------
;       Rutina lectora de teclado
;-----------------------------------------
TECLA:
        CALL    CLEAR
        CALL    Y3
        MOV     A,P1
        CALL    Y7
        ANL     A,#0FH
        CJNE    A,#0EH,TC1
        MOV     A,#02H
        JMP     TC4
TC1:
        CJNE    A,#0DH,TC2
        MOV     A,#04H
        JMP     TC4
TC2:
        CJNE    A,#0BH,TC3
        MOV     A,#06H
        JMP     TC4
TC3:
        MOV     A,#0FFH
TC4:
        RET
        ;
;-----------------------------------------
;       Rutina de servicio de lecturas
;-----------------------------------------
SERV:
        MOV     A,#0
        CALL    LECTURA                 ; Lee nivel del agua
        MOV     NIVEL,A
        MOV     A,#1
        CALL    LECTURA                 ; Lee temperatura
        MOV     TMPE,A
        MOV     BCD,#0FFH
        CALL    DISPLAY
        CALL    RECV
        CJNE    A,#0FFH,SV0
        JMP     SERV
SV0:
        CJNE    A,#DT,SV1
        MOV     DPTR,#1E01H
        MOVX    A,@DPTR
        MOV     B,A
```

```
        MOV   A,TMPE
        MOV   DPTR,#1E45H
        MOVX  @DPTR,A
        MOV   A,#TM
        CALL  SEND                      ; Envio temp. a solicitó
        MOV   B,#PC
        MOV   A,#TM
        CALL  SEND                      ; Envio temp. a PC
        JMP   SERV
SV1:
        CJNE  A,#DN,SV2
        MOV   DPTR,#1E01H
        MOVX  A,@DPTR
        MOV   B,A
        MOV   A,NIVEL
        MOV   DPTR,#1E45H
        MOVX  @DPTR,A
        MOV   A,#NV
        CALL  SEND                      ; Envio nivel a solicitó
        MOV   B,#PC
        MOV   A,#NV
        CALL  SEND                      ; Envio nivel a PC
        JMP   SERV
SV2:
        CJNE  A,#TE,SERV
        RET
        ;
;-----------------------------------------
;      Rutina para clarificar puerto
;-----------------------------------------
CLEAR:
        MOV   P1,#0FFH      ; Clarifica puerto
        RET
        ;
;-----------------------------------------
;            Rutina de retardo
;-----------------------------------------
DELAY:
        MOV   CNT0,#80H
LOOP1:  DJNZ  CNT0,LOOP1
        RET
        ;
;-----------------------------------------
;   Control de los dispositivos externos
;-----------------------------------------
Y0:
          ORL   P3,#00000000B   ; Selecc. e inicia conv. A/D
          ANL   P3,#11100011B
          RET
Y1:
          ORL   P3,#00000100B   ; Abre driver del conv. A/D
          ANL   P3,#11100111B
          RET
Y2:
          ORL   P3,#00001000B   ; Carga latch para DISPLAY
          ANL   P3,#11101011B
          RET
```

```
Y3:
          ORL   P3,#00001100B   ; Lectura actual del teclado
          ANL   P3,#11101111B
          RET
Y4:
          ORL   P3,#00010000B
          ANL   P3,#11110011B
          RET
Y5:
          ORL   P3,#00010100B
          ANL   P3,#11110111B
          RET
Y6:
          ORL   P3,#00011000B
          ANL   P3,#11111011B
          RET
Y7:
          ORL   P3,#00011100B   ; Estabiliza decoder
          ANL   P3,#11111111B
          RET
          ;
;-----------------------------------------
;    Rutina lectura convertida de ent X
;-----------------------------------------
LECTURA:
          MOV   P1,A            ; Selecc. entrada X
          CALL  Y0              ; Inicia conv. A/D
          CALL  Y7
          CALL  DELAY
          CALL  CLEAR           ; CLARIFICA PUERTO
          CALL  Y1              ; ABRE DRIVER
          MOV   A,P1            ; LEE SENAL DIGITILIZADA
          CALL  Y7
          RET
          ;
;-----------------------------------------
;    Rutina despliega información
;-----------------------------------------
DISPLAY:
          CAL   CONV
          MOV   CNT1,#0FFH
          MOV   DPH,#1DH
DSPL0:    MOV   A,BCD
          MOV   B,A
          ANL   A,#0FH
          MOV   DPL,A
          MOVX  A,@DPTR
          MOV   P1,#7FH
          CALL  Y2
          CALL  Y7
          MOV   P1,A
          CALL  Y2
          CALL  Y7
          MOV   A,B
          SWAP  A
          ANL   A,#0FH
          MOV   DPL,A
```

```
          MOVX  A,@DPTR
          ORL   A,#80H
          MOV   P1,#0FFH
          CALL  Y2
          CALL  Y7
          MOV   P1,A
          CALL  Y2
          CALL  Y7
          DJNZ  CNT1,DSPL0
          MOV   P1,#0FFH
          CALL  Y2
          CALL  Y7
          RET
          ;
;-----------------------------------------
;  Rutina convertidora HEX a BCD
;-----------------------------------------
CONV:
        MOV   A,DATO
        MOV   B,#10
        DIV   AB
        MOV   RESI,B
        MOV   B,#10H
        MUL   AB
        ADD   A,RESI
        MOV   BCD,A
        RET
        ;
;-----------------------------------------
;        Rutina de envio de mensaje
;-----------------------------------------
SEND:
        CJNE  R6,#0,SEND                ; Ya se vació ?
        MOV   DPTR,#1E44H
        MOVX  @DPTR,A
        MOV   DPTR,#1E40H
        MOV   A,B
        MOVX  @DPTR,A
        MOV   R4,#0
        MOV   R6,#1                     ; Lleno el buffer
        RET
        ;
;-----------------------------------------
;     Rutina de recepción de mensaje
;-----------------------------------------
RECV:
        MOV   A,#0FFH
        CJNE  R5,#01,REC0
        MOV   DPTR,#1E04H
        MOVX  A,@DPTR
        MOV   R5,#0
REC0:   RET
        ;
        ;
        END START
```

```
;=================================================================
;
;                    SISTEMA DISTRIBUIDO 8051
;                     Programas de aplicación
;              México (c) 1990   CINVESTAV del I.P.N.
;
;            Archivo:  APL12.ASM
;            Programa: Módulo 2
;            Función:  Controla el bombeo y calentamiento
;                      del agua del sistema.
;            Autor:    Ing. Alejandro Tinoco Alvarado
;            Asesor:   Dr. Jan Janecek Hyan
;            Fecha:    Agosto 1990
;=================================================================
MAESTRO EQU   0FEH
MYADDR  EQU   12H
DSTADDR EQU   05H
;
ACKN    EQU   0FH
POLL    EQU   10H
ADDPOL  EQU   11H
TREND   EQU   12H
DATOS   EQU   15H
;
MRECV   EQU   0030H
MSEND   EQU   0033H
;
TMOUT   EQU   07H
;
;   VARIABLES USADAS
;
CNT0    EQU   60H
CNT1    EQU   61H
CNT2    EQU   62H
TEMP0   EQU   63H
TEMP1   EQU   64H
TMPE    EQU   65H
TEDES   EQU   66H
NIVEL   EQU   67H
;
;  CONSTANTES USADAS
;
IN      EQU   02H                       ; Inicio
TM      EQU   04H                       ; Temperatura
NV      EQU   06H                       ; Nivel del agua
AL      EQU   08H                       ; Agua lista
TE      EQU   10H                       ; Terminé
DT      EQU   12H                       ; Dame temperatura
DN      EQU   14H                       ; Dame nivel
EP      EQU   16H                       ; Esperando
BM      EQU   18H                       ; Bombeando
CA      EQU   20H                       ; Calentando
LV      EQU   22H                       ; Llenando vaso
;
ALT     EQU   20H
;
```

```
;    NODOS USADOS
M1       EQU    6DH
M2       EQU    12H
M3       EQU    25H
PC       EQU    05H
;
         ORG    1DF8H                          ; Señal de fin de mensaje
MEND     DB     MAESTRO,MYADDR,TREND,00H
         ;
         ORG    1DFCH                          ; Mensaje de reconocimiento
RECON    DB     MAESTRO,MYADDR,ACKN,00H
         ;
         ORG    1E40H                          ; Mensaje de información
MENSG    DB     DSTADDR,MYADDR,DATOS,01H
         ;
         ;
         ORG    2000H
START:   JMP    MAIN
         ;
         ORG    2023H                          ; Mensaje recibido
         JMP    LRINT
         ;
         ORG    2028H                          ; Mensaje transmitido
         JMP    LTINT
         ;
;-----------------------------------------------------------------
;        Rutina de servicio a la llegada de un mensaje
;-----------------------------------------------------------------
         ORG    2030H
LRINT:
;
LTINT:
;
;       VER RUTINAS DE LIBRERIA (APENDICE D)
;
;
;-----------------------------------------------------------------
;      PROGRAMA DE APLICACION        ( MODULO 2)
;-----------------------------------------------------------------
MAIN:
         MOV    PSW,#0
         ;
         CLR    T1
         SETB   ES
         SETB   EA
         SETB   SM2
         SETB   REN
         ;
         MOV    R0,#0                          ; Bandera Recon. apag.
         MOV    R1,#0                          ; Contador de datos
         MOV    R2,#0                          ; Apuntador Buff. llegada
         MOV    R3,#0                          ; Apuntador Buff. almacen.
         MOV    R4,#0                          ; Inicializa TMOUT
         MOV    R5,#0                          ; Buff-Rx   vacio
         MOV    R6,#0                          ; Buff-Tx   vacio
         MOV    R7,#0                          ; Bandera de fin de mensaje
```

```
        ;
        MOV   P1,#0FFH                  ; Dispositivos apagados
        MOV   NIVEL,#1
        MOV   TMPE,#1
LOOP:
        CALL  RECV
        CJNE  A,#0FFH,LP0               ; Espera señal de inicio
        JMP   LOOP
LP0:
        CJNE  A,#IN,LOOP                ; Recibí señal de inicio ?
        INC   DPTR
        MOVX  A,@DPTR
        MOV   TEDES,A                    ; Obtiene temperatura deseada
        CALL  BOMBEA
        CALL  CALIENT
        MOV   B,#M3
        MOV   A,#AL
        CALL  SEND                      ; Envía señal de agua lista a M
        MOV   B,#PC
        MOV   A,#AL
        CALL  SEND                      ; Lo envía también a PC
        JMP   LOOP
        ;
        ;
;-----------------------------------------
;      Rutina de bombeo de agua
;-----------------------------------------
BOMBEA:
        MOV   B,#PC
        MOV   A,#BM
        CALL  SEND
BB0:
        MOV   A,#ALT
        MOV   B,NIVEL
        DIV   AB
        CJNE  A,#0,BB1
        JMP   BB4
BB1:
        CALL  BOMBON
        MOV   B,#M1
        MOV   A,#DN
        CALL  SEND
BB2:
        CALL  RECV
        CJNE  A,#0FFH,BB3
        JMP   BB2
BB3:
        CJNE  A,#NV,BB1
        INC   DPTR
        MOVX  A,@DPTR
        MOV   NIVEL,A
        JMP   BB0
BB4:
        CALL  BOMBOFF
        RET
        ;
```

```
;-----------------------------------------
;     Rutina de calentamiento de agua
;-----------------------------------------
CALIENT:
        MOV   B,#PC
        MOV   A,#CA
        CALL  SEND
CT0:
        MOV   A,TEDES
        MOV   B,TMPE
        DIV   AB
        CJNE  A,#0,CT1
        JMP   CT4
CT1:
        CALL  CALON
        MOV   B,#M1
        MOV   A,#DT
        CALL  SEND
CT2:
        CALL  RECV
        CJNE  A,#0FFH,CT3
        JMP   CT2
CT3:
        CJNE  A,#TM,CT1
        INC   DPTR
        MOVX  A,@DPTR
        MOV   TMPE,A
        JMP   CT0
CT4:
        CALL  CALOFF
        RET
        ;
;-----------------------------------------
;  Rutinas de enc/apag. de bomba y calent.
;-----------------------------------------
BOMBON:
        CLR   P1.0                      ; Enciende bomba
        RET
BOMBOFF:
        SETB  P1.0                      ; Apaga bomba
        RET
CALON:
        CLR   P1.2                      ; Enciende calentador
        RET
CALOFF:
        SETB  P1.2                      ; Apaga calentador
        RET
;-----------------------------------------
; Rutinas de envio y recepción de mensajes
;-----------------------------------------
SEND:
;
RECV:
;           VER PROGRAMA MODULO 1
;
        ;
        END START
```

```
;=================================================================
;
;                      SISTEMA DISTRIBUIDO 8051
;                       Programas de aplicación
;                 México (c) 1990   CINVESTAV del I.P.N.
;
;              Archivo:  APL25.ASM
;              Programa: Módulo 3
;              Función:  Controla el brazo mecánico para
;                        colocar un vaso en el lugar adecuado
;                        y abre-cierra una válvula de paso.
;              Autor:    Ing. Alejandro Tinoco Alvarado
;              Asesor:   Dr. Jan Janecek Hyan
;              Fecha:    Agosto  1990
;=================================================================
MAESTRO EQU   0FEH
MYADDR  EQU   25H
DSTADDR EQU   05H
;
ACKN    EQU   0FH
POLL    EQU   10H
ADDPOL  EQU   11H
TREND   EQU   12H
DATOS   EQU   15H
;
MRECV   EQU   0030H
MSEND   EQU   0033H
;
TMOUT   EQU   10H
;
;   VARIABLES USADAS
;
CNT0    EQU   60H
CNT1    EQU   61H
CNT2    EQU   62H
TEMP0   EQU   63H
TEMP1   EQU   64H
POSA    EQU   65H
POS     EQU   66H
PACT    EQU   67H
NIVEL   EQU   68H
;
;   CONSTANTES USADAS
;
HOME    EQU   00H                       ; Motor en CASA
ENTR    EQU   10H                       ;          ENTREGA
WAIT    EQU   50H                       ;          ESPERA
VASO    EQU   85H                       ;          VASO
;
IN      EQU   02H                       ; Inicio
TM      EQU   04H                       ; Temperatura
NV      EQU   06H                       ; Nivel del agua
AL      EQU   08H                       ; Agua lista
TE      EQU   10H                       ; Terminé
DT      EQU   12H                       ; Dame temperatura
DN      EQU   14H                       ; Dame nivel
```

```
EP      EQU   16H                       ; Esperando
BM      EQU   18H                       ; Bombeando
CA      EQU   20H                       ; Calentando
LV      EQU   22H                       ; Llenando vaso
;
;   NODOS USADOS
;
M1      EQU   6DH
M2      EQU   12H
M3      EQU   25H
PC      EQU   05H
;
        ORG   1C00H                     ; Sec. de activ. del motor
SEC     DB    00H,0FEH,0FCH,0FDH,0F9H,0FBH,0F3H,0F7H,0F6H,00H
        ;
        ORG   1DF8H                     ; Señal de fin de mensaje
MEND    DB    MAESTRO,MYADDR,TREND,00H
        ;
        ORG   1DFCH                     ; Mensaje de reconocimiento
RECON   DB    MAESTRO,MYADDR,ACKN,00H
        ;
        ORG   1E40H                     ; Mensaje de información
MENSG   DB    DSTADDR,MYADDR,DATOS,01H
        ;
        ;
        ORG   2000H
START:  JMP   MAIN
        ;
        ORG   2023H                     ; Mensaje recibido
        JMP   LRINT
        ;
        ORG   2028H                     ; Mensaje transmitido
        JMP   LTINT
        ;
;-----------------------------------------------------------------
;       Rutina de servicio a la llegada de un mensaje
;-----------------------------------------------------------------
        ORG   2030H
LRINT:
;
LTINT:
;             VER RUTINAS DE LIBRERIA (APENDICE D)
;
;
;
;-----------------------------------------------------------------
;     PROGRAMA DE APLICACION       (MODULO 3)
;-----------------------------------------------------------------
MAIN:
        MOV   PSW,#0
        ;
        CLR   TI
        SETB  ES
        SETB  EA
        SETB  SM2
        SETB  REN
        ;
```

```
        MOV   R0,#0                     ; Bandera Recon. apag.
        MOV   R1,#0                     ; Contador de datos
        MOV   R2,#0                     ; Apuntador Buff. llegada
        MOV   R3,#0                     ; Apuntador Buff. almacen.
        MOV   R4,#0                     ; Inicializa TMOUT
        MOV   R5,#0                     ; Buff-Rx   vacio
        MOV   R6,#0                     ; Buff-Tx   vacio
        MOV   R7,#0                     ; Bandera de fin de mensaje RQ.
        ;
        MOV   PACT,#1
        CLR   P3.2                      ; prepara dispositivos
        SETB  P3.3
        CLR   P3.4
        CALL  GHOME                     ; Motor en posición HOME
LOOP:
        CALL  RECV
        CJNE  A,#0FFH,LP0               ; Espera por señal de inicio
        JMP   LOOP
LP0:
        CJNE  A,#IN,LP1                 ; Recibí señal de inicio?
        CALL  GWAIT
        CALL  ALARMA
        JMP   LOOP
LP1:
        CJNE  A,#AL,LOOP                ; Recibi señal agua lista ?
        CALL  GVASO                     ; Motor en posición VASO
        CALL  LLENADO                   ; Llenando vaso
        CALL  GENTR                     ; Motor en posición ENTR
        MOV   B,#M1
        MOV   A,#TE
        CALL  SEND                      ; Avisa a M1 que terminó
        MOV   B,#PC
        MOV   A,#TE
        CALL  SEND                      ; Avisa a PC que terminó
        CALL  ALARMA
        JMP   LOOP
        ;
;-------------------------------
; Rutina hace sonar la alarma
;-------------------------------
ALARMA:
        MOV   CNT2,#5
ALP1:
        CLR   P3.3                      ; Suena la alarma
        CALL  DELAY
        SETB  P3.3                      ; Deja de sonar
        CALL  DELAY
        DJNZ  CNT2,ALP1
        RET
        ;
;-------------------------------
; Rutina  llenado del vaso
;-------------------------------
LLENADO:
        MOV   B,#PC
        MOV   A,#LV                     ; Avisa a PC de su estado
        CALL  SEND
```

```
LL0:
        MOV   B,#M1
        MOV   A,#DN                     ; Solicita nivel
        CALL  SEND
LL1:
        CALL  RECV
        CJNE  A,#0FFH,LL2               ; Espera respuesta
        JMP   LL1
LL2:
        CJNE  A,#NV,LL0
        INC   DPTR
        MOVX  A,@DPTR                   ; Obtiene nivel inicial
        DEC   A
        MOV   NIVEL,A
        CALL  VAON                      ; Abre la válvula
LL3:
        MOV   B,#M1
        MOV   A,#DN                     ; Solicita nivel
        CALL  SEND
LL4:
        CALL  RECV
        CJNE  A,#0FFH,LL5               ; Espera respuesta
        JMP   LL4
LL5:
        CJNE  A,#NV,LL3
        INC   DPTR
        MOVX  A,@DPTR                   ; Obtiene nivel actual
        MOV   B,NIVEL
        DIV   AB
        CJNE  A,#0,LL3
        CALL  VAOFF                     ; Cierra la válvula
        RET
        ;
;-------------------------------
; Rutina perdida de tiempo Slow
;-------------------------------
DELAY:
        MOV   CNT0,#0FFH
DLP1:   MOV   CNT1,#0FFH
DLP2:   DJNZ  CNT1,DLP2
        DJNZ  CNT0,DLP1
        RET
        ;
;-------------------------------
; Rutina perdida de tiempo Fast
;-------------------------------
DELAYF:
        MOV   CNT0,#20H
DFLP1:  MOV   CNT1,#0FFH
DFLP2:  DJNZ  CNT1,DFLP2
        DJNZ  CNT0,DFLP1
        RET
        ;
```

```
;-------------------------------
; Rutina  enc/apag válvula
;-------------------------------
VAON:
        SETB   P3.2                     ; Abre la válvula
        RET
VAOFF:
        CLR    P3.2                     ; Cierra la válvula
        RET
        ;
;-------------------------------
;  Rutina: motor Go-home
;-------------------------------
GHOME:
        MOV    A,POSA
        CJNE   A,#HOME,GH1
        JMP    GHEND
GH1:
        DEC    PACT
        MOV    DPH,#1CH
        MOV    DPL,PACT
        MOVX   A,@DPTR
        CJNE   A,#0,GH2
        MOV    PACT,#08H
        MOV    DPH,#1CH
        MOV    DPL,PACT
        MOVX   A,@DPTR
GH2:    MOV    P1,A
        CALL   DELAYF
        JNB    P3.4,GH1
GHEND:
        MOV    POSA,#0
        RET
        ;
;-------------------------------
;  Rutina: motor Go-enter
;-------------------------------
GENTR:
        MOV    A,POSA
        CJNE   A,#HOME,GE1
        MOV    A,#10H
        CALL   GMOTR
        JMP    GEEND
GE1:    CJNE   A,#WAIT,GE2
        MOV    A,#40H
        CALL   GMOTL
        JMP    GEEND
GE2:    CJNE   A,#VASO,GEEND
        MOV    A,#75H
        CALL   GMOTL
GEEND:
        MOV    POSA,#ENTR
        RET
        ;
```

```
;------------------------------
;  Rutina: motor Go-wait
;------------------------------
GWAIT:
        MOV   A,POSA
        CJNE  A,#HOME,GW1
        MOV   A,#50H
        CALL  GMOTR
        JMP   GWEND
GW1:    CJNE  A,#ENTR,GW2
        MOV   A,#40H
        CALL  GMOTR
        JMP   GWEND
GW2:    CJNE  A,#VASO,GWEND
        MOV   A,#35H
        CALL  GMOTL
GWEND:
        MOV   POSA,#WAIT
        RET
        ;
;------------------------------
;  Rutina: motor Go-vaso
;------------------------------
GVASO:
        MOV   A,POSA
        CJNE  A,#HOME,GV1
        MOV   A,#85H
        CALL  GMOTR
        JMP   GVEND
GV1:    CJNE  A,#ENTR,GV2
        MOV   A,#75H
        CALL  GMOTR
        JMP   GVEND
GV2:    CJNE  A,#WAIT,GVEND
        MOV   A,#35H
        CALL  GMOTR
GVEND:
        MOV   POSA,#VASO
        RET
        ;
;------------------------------
; Rutina: Gira motor Left/Right
;------------------------------
GMOTL:
        MOV   POS,A
GML1:
        DEC   PACT
        MOV   DPH,#1CH
        MOV   DPL,PACT
        MOVX  A,@DPTR
        CJNE  A,#0,GML2
        MOV   PACT,#08H
        MOV   DPH,#1CH
        MOV   DPL,PACT
        MOVX  A,@DPTR
GML2:   MOV   P1,A
```

```
        CALL   DELAYF
        DJNZ   POS,GML1
        JMP    GEND
GMOTR:
        MOV    POS,A
GMR1:
        INC    PACT
        MOV    DPH,#1CH
        MOV    DPL,PACT
        MOVX   A,@DPTR
        CJNE   A,#0,GMR2
        MOV    PACT,#01H
        MOV    DPH,#1CH
        MOV    DPL,PACT
        MOVX   A,@DPTR
GMR2:   MOV    P1,A
        CALL   DELAYF
        DJNZ   POS,GMR1
GEND:
        RET
        ;
;-----------------------------------------
;         Rutina de envio de mensaje
;-----------------------------------------
SEND:
        CJNE   R6,#0,SEND                  ; Ya se vació ?
        MOV    DPTR,#1E44H
        MOVX   @DPTR,A
        MOV    DPTR,#1E40H
        MOV    A,B
        MOVX   @DPTR,A
        MOV    R4,#0
        MOV    R6,#1                       ; Lleno el buffer
        RET
        ;
;-----------------------------------------
;      Rutina de recepción de mensaje
;-----------------------------------------
RECV:
        MOV    A,#0FFH
        CJNE   R5,#01,REC0
        MOV    DPTR,#1E04H
        MOVX   A,@DPTR
        MOV    R5,#0
REC0:   RET
        ;
        ;
        END START
```

```
/*-------------------------------------------------------------------*
/*                   SISTEMA  DISTRIBUIDO  8051                     *
/*                                                                   *
/*                 C I N V E S T A V    del  I P N                   *
/*           Depto. Ingenieria Eléctrica  Secc. Computación          *
/*                                                                   *
/*      Archivo:  APLPC.C                                            *
/*      Programa: Módulo PC                                          *
/*      Función:  Monitorea el funcionamiento del sistema            *
/*                de calentamiento de agua para llenar un            *
/*                vaso con la cantidad de agua adecuada.             *
/*      Autor:    Ing. Alejandro Tinoco Alvarado                     *
/*      Asesor:   Dr. Jan Janecek Hyan                               *
/*      Fecha:    Agosto-1990                                        *
/*-------------------------------------------------------------------*
#include <stdio.h>
#include <dos.h>
#include "sd.h"

byte            IntMask, CIntMask;
byte            DesAddr, MyAddr, AckAddr, bcc;
byte            Edo, mg, len, dato, cde;
byte            MsgR[10], MsgT[10], BuffR[10];
byte            c, n, nl, ln, Hm, pch, pcl;
int             mx, my;
bool            Finish, Status, QEdit, MsgRRdy, MsgTRdy, MsgARdy;
bool            Termine;

#define     IN      0x02
#define     TM      0x04
#define     NV      0x06
#define     AL      0x08
#define     TE      0x10
#define     DT      0x12
#define     DN      0x14
#define     EP      0x16
#define     BM      0x18
#define     CA      0x20
#define     LV      0x22

#define     M1      0x6D
#define     M2      0x12
#define     M3      0x25
#define     PC      0x05

void SisOpc(void)
{
   SetFrame(1,1,80,3,bw);
   wselect1;
   wsprintf(15,1,by,"MONITOR DEL SISTEMA DE CALENTAMIENTO    Módulo PC"
   SetFrame(1,21,80,24,bw);
   wselect3;
   wclrscr(bw);
   wsprintf(30,1,by,"  C - Continuar");
   wsprintf(30,2,by,"ESC - Salir");
   wselect2;
}
```

```
void SisApl(void)
{
    byte c;
    bool Exit=FALSE;

    SetFrame(1,4,80,20,bw);
    wselect2;
    wclrscr(bw);
    wsprintf(30,3,by,"M1: ");
    wsprintf(30,4,by,"M2: ");
    wsprintf(30,5,by,"M3: ");
    wsprintf(40,7,by,"Nivel: ");
    wsprintf(40,8,by,"Tempe: ");
    wsprintf(58,13,by,"Bomba: APAGADA");
    wsprintf(40,10,by,"Calen: APAGADO");
    wsprintf(20,10,by,"Valvula: CERRADA");
    wsprintf(10,13,by,"Temp. deseada: ");
    HabInt;
    SisOpc();
    Termine = FALSE;

    while (!Exit)
     {
     if (MsgRRdy)
      {
       switch (MsgR[3])
        {
         case IN:
                  wsprintf(34,3,by,"Inicio...        ");
                  wsprintf(34,4,by,"Inicio...        ");
                  wsprintf(34,5,by,"Inicio...        ");
                  Termine = FALSE;
                  break;
         case EP:
                  wsprintf(34,3,by,"Seleccionando...");
                  wprintf(25,13,by,"%d   ",MsgR[4]);
                  break;
         case NV:
                  wsprintf(34,3,by,"Lectura...       ");
                  wprintf(47,7,by,"%d   ",MsgR[4]);
                  break;
         case TM:
                  wprintf(47,8,by,"%d   ",MsgR[4]);
                  break;
         case AL:
                  wsprintf(47,10,by,"APAGADO  ");
                  wsprintf(34,4,bk,"Agua Caliente ! ");
                  break;
         case BM:
                  wsprintf(34,4,by,"Bombeando...     ");
                  wsprintf(65,13,bk,"ENCENDIDA");
                  break;
         case CA:
                  wsprintf(65,13,by,"APAGADA  ");
                  wsprintf(34,4,by,"Calentando...    ");
                  wsprintf(47,10,bk,"ENCENDIDO");
                  break;
```

```
        case LV:
                wsprintf(34,5,by,"Llenando vaso...");
                wsprintf(29,10,bk,"ABIERTA");
                break;
        case TE:
                wsprintf(29,10,by,"CERRADA");
                wsprintf(47,7,by,"    ");
                wsprintf(47,8,by,"    ");
                wsprintf(34,3,by,"                    ");
                wsprintf(34,4,by,"                    ");
                wsprintf(34,5,bk,"                    ");
                wsprintf(10,11,bk,"Vaso listo !");
                Termine = TRUE;
                bell();
                wsprintf(34,3,bk,"Esperando...     ");
                break;
      }
     MsgRRdy = FALSE;
    }
   if (kbhit())
    {
      c = getch();
      switch(c)
         {
           case 'C':
           case 'c':
                   if (!MsgTRdy && Termine)
                     {
                      MsgT[0] = MI;
                      MsgT[1] = MyAddr;
                      MsgT[2] = DATOS;
                      MsgT[3] = 1;
                      MsgT[4] = IN;
                      MsgTRdy = TRUE;
                      Termine = FALSE;
                      wsprintf(10,11,by,"              ");
                     }
                   else
                     {
                      beep();
                     }
                   break;
           case  27:
                   Exit = TRUE;
                   DesInt;
                   break;
           default:
                   beep();
                   continue;
         }
     }
   }
}
```

```
main()
{
   wclrscr(bw);
   SetFrame(16,8,65,15,gw);
   wsprintf(28,10,gn,"Programa de aplicación 8051");
   wsprintf(28,12,gn,"(c) CINVESTAV I.P.N.   1990");
   wsprintf(21,14,gn,"por: Alejandro Tinoco A.      MEXICO");
   delay(2000);

   ComInit();
   MyAddr = 0x05;
   DesAddr = 0xFE;
   Finish = FALSE;
   MsgRRdy = MsgTRdy = MsgARdy = FALSE;
   Edo = cde = mg = 0;

   SisApl();

   ComRest();
   wselect0; wclrscr(bw);
   SetFrame(16,10,65,16,gw);
   wsprintf(36,13,gn,"  A d i o s  ");
   delay(1000);
   wclrscr(nw);
}
```

El jurado designado por la Sección de Computación del Departamento de Ingeniería Eléctrica del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, aprobó esta tesis el 27 de Septiembre de 1990.

Dr. Juan Manuel Ibarra Zannatha

Dr. Armando Maldonado Talamantes

M. en C. José Oscar Olmedo Aguirre