UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERIA Y ARQUITECTURA
ESCUELA DE INGENIERIA ELECTRICA

"DISEÑO Y CONSTRUCCION DE UNA INTERFASE IEEE 488 CON LA COMPUTADORA AT LEMMON 286S"

TRABAJO DE GRADUACION PRESENTADO POR:

ALBERTO ERNESTO HERNANDEZ PAZ

PARA OPTAR AL TITULO DE

INGENIERO ELECTRICISTA

SEPTIEMBRE 1993

SAN SALVADOR, EL SALVADOR, CENTRO AMERICA

UNIVERSIDAD DE EL SALVADOR

RECTOR:
DR. FABIO CASTILLO FIGUEROA

SECRETARIO GENERAL:
LIC. MIRNA ANTONIETA PERLA DE ANAYA

FACULTAD DE INGENIERIA Y ARQUITECTURA

DECANO:
ING. JUAN JESUS SANCHEZ SALAZAR

SECRETARIO:
ING. JOSE RIGOBERTO MURILLO CAMPOS

ESCUELA DE INGENIERIA ELECTRICA

DIRECTOR:
ING. RICARDO ERNESTO CORTEZ

Coordinador del trabajo:

ING. RICARDO ERNESTO CORTEZ

Asesor:

ING. OSBALDO ADOLFO CAMPOS

ACTA DE CONSTANCIA DE NOTA Y DEFENSA FINAL

En esta fecha, 3 de Septiembre de 1993,
en el local de Sala de Lectura de la Escuela de Ingeniería Eléctrica
a las 16:00 horas, con la presencia de las siguientes autoridades de la Escuela de Ingeniería Eléctrica de la Universidad de El Salvador:

1- Ing. Ricardo E. Cortez
Director de la E.I.E.

2- Ing. Jorge A. Galdámez
Secretario de la E.I.E.

3- _____

ESCUELA DE INGENIERIA ELECTRICA
FACULTAD DE INGENIERIA
Y ARQUITECTURA
Universidad de El Salvador

Firma

con el Honorable Jurado de evaluación integrado por las personas siguientes:

1- Ing. Jairo Edgaro Guzmán Flores

2- Ing. Oscar Alonso Rodríguez Linares

3- Ing. Miguel Angel Ramírez

4- _____

5- _____

6- _____

Se efectuó la defensa final reglamentaria del Trabajo de Graduación: "DISEÑO Y CONSTRUCCION DE UNA INTERFASE IEEE 488 CON LA COMPUTADORA AT LEMMON 286S"

a cargo del (los) Br(es): Alberto Ernesto Hernández Paz

Habiendo obtenido el presente trabajo una nota final, global de 8.5

(ocho punto cinco)

D E D I C A T O R I A:

- Especialmente a **Alejandro,** a **Raquel** y a mis **Padres** que son lo más importante en mi vida.

- A **Jeanette** e **Irvin** mi mayor gratitud.

- Y a todos aquellos que me ayudaron y colaboraron desinteresadamente.

PREFACIO

Desde hace muchos años el desarrollo de la comunicación a distancia con instrumentos de control e instrumentación eléctronicos ha llevado un ritmo acelerado, dentro de esta área el bus estandar IEEE-488 también conocido como GPIB, ha tenido un papel importante.

Este bus, introducido desde hace mas de veinte años aún conserva su vigencia como un elemento mas que permite la comunicación entre instrumentos eléctronicos por medio del computador, siendo esto un factor importante para el desarrollo de la investigación en áreas tales como: la quimica, la física, la eléctronica, la ingeniería civil, etc.

A pesar de esto en la Escuela de Ingeniería Eléctrica, el estudio y la investigación aplicando este tipo de bus es poco conocida, incluso a pesar de que en la escuela existen dispositivos que pueden ser operados con este bus.

El objetivo de este trabajo es diseñar y construir una interface sencilla para operar el bus estandar IEEE-488 por medio de una computadora PC-501-AT Lemmon; así como elaborar su software de operación. Senteando las bases teóricas sobre el funcionamiento de este bus.

Los alcances logrados cumplen con los objetivos planteados, finalizando el trabajo con la construcción de la interfaz y su programa de operación. Dejando planteada también la suficiente base teórica para cualquier desarrollo posterior tanto en el circuito como en el software.

RESUMEN DEL TRABAJO

El- presente trabajo se desarrolla a partir de una explicación de la concepción del bus IEEE-488 hasta la construcción y operación de la interfaz que operará con los requemientos de este bus.

En el capitulo I se describe de manera general la concepción del bus, su forma física, los requerimientos eléctricos, mecánicos y protocolos de comunicación, pero sin profundizar sobre cada tema especifico.

En el segundo capitulo se describe el circuito de interfaz, su funcionamiento y comunicación dentro de la computadora; además de una breve explicación de la construcción del mismo.

Los capitulos III y IV son complementarios y se relacionan con la programación de la interfaz y la implementación de los protocolos de comunicación del bus. En el capitulo III se describen las funciones de interfaz como están definidas en el estandar, así como otros protocolos y requerimientos del mismo estandar.

En el capitulo IV se describe el programa principal que operará la tarjeta de interfaz, dicho programa es elaborado en lenguaje Turbo Pascal versión 4.0, aunque en una versión previa se utilizaron unas subrutinas en lenguaje ensamblador.

TABLA DE CONTENIDOS

Capitulo Pag.

I. BUS DE INTERFACE IEEE488 1

1.1 Generalidades.....1
1.2 Historia del Bus de Interface IEEE488 2
1.3 Conceptos Generales del Bus IEEE488 o GPIB.....3
1.4 Lineas de Señal de la GPIB.....5
1.4.1 Líneas de Datos 5
1.4.2 Líneas de Comunicación o Handshake.....6
1.4.3 Líneas de Manejo de la Interface.....7
1.5 Operación del bus IEEE488 o GPIB.....10
1.5.1 Funciones de Interface 10
1.5.2 Comunicación entre dispositivos (Secuencia Handshake).....11
1.5.3 Tipos de Mensajes de la GPIB.....14
1.5.4 Codificación de los Mensajes de la Interface.....16
1.6 Especificaciones Eléctricas y Mecánicas del bus IEEE488.....19
1.6.1 Especificaciones Eléctricas 19
1.6.2 Especificaciones Mecánicas.....22
CONCLUSIONES.....26
REFERENCIAS BIBLIOGRAFICAS.....27

II DISEÑO DEL CIRCUITO DE INTERFACE IEEE488.....28

Introducción.....28
2.1 Interconexión de la Interface con la computadora.....28
2.2 Descripción de los Slots o Canales I/O de Expansión de las computadoras IBM PC o compatibles.....29
2.3 Conexión con la Interface 31
2.4 Propuesta para el Circuito de Interface 32
2.4.1 Descripción General del 8255A (Modo 0).....32
2.4.2 Lógica de acceso al 8255A.....34
2.4.3 Circuito de Interface para el bus IEEE488.....35
2.4.4 Construcción de la Tarjeta de Interface IEEE488 o GPIB.....40
CONCLUSIONES.....44
REFERENCIAS BIBLIOGRAFICAS.....45

III. REQUERIMIENTOS DE PROGRAMACION DEL ESTANDAR IEEE488.....46

Introducción 46

3.1 Funciones de Interfaz.....46
3.1.1 Función Source Handshake (SH).....50
3.1.2 Función Acceptor Handshake (AH).....54
3.1.3 Función Talker (T o TE).....56
3.1.4 Función Listen (L o LE).....59
3.1.5 Función Controller C (Controlador).....61
3.2 Protocolo de las Subrutinas básicas 63
3.3 Protocolo para las funciones Talker y Listen.....67
CONCLUSIONES.....72
REFERENCIAS BIBLIOGRAFICAS 73

IV. DISEÑO DEL PROGRAMA DE MANEJO DE LA INTERFAZ.....74

Introducción 74
4.1 Descripción del Problema 74
4.2 Analisis del Problema.....75
4.2.1 Requerimientos del Estandar IEEE488.2.....78
4.3 Programa principal (estructura).....79
4.3.1 Variables 80
4.3.2 Inicialización 80
4.3.3 Portada.....81
4.3.4 Mapa de Dispositivos 81
4.3.5 Teclado.....81
4.3.6 Mover Cursor 81
4.3.7 Configurar.....81
4.4 Guía de Usuario.....84
CONCLUSIONES.....86
REFERENCIAS BIBLIOGRAFICAS.....87

CONCLUSIONES GENERALES Y RECOMENDACIONES.....88

ANEXO A
Subsets de las funciones de Interfaz.....90

ANEXO B
Especificaciones del 8255A y el 74LS640.....96

ANEXO C
Listado del Programa Principal.....100

LISTA DE TABLAS

Tabla Pag.

1.1 Comandos de Unilínea.....15
1.2 Carta de Codigo ASCII/ISO e IEEE.....17
1.3 Codificación de los Mensajes de Interface del bus IEEE-488.....18
1.4 Relación entre nivel lógico y voltaje 19
1.5 Especificaciones de voltaje y corriente en emisores y receptores 20
2.1 Configuración de Puertos en el Modo 0.....34
2.2 Dirección de los 3 puertos y el registro de control.....34
2.3 Dirección de las líneas Handshake 37
2.4 Control del flujo de datos 39
2.5 Asignación de las líneas de la GPIB al 8255A.....40
3.1 Valor de tiempos limite para las funciones de Interfaz.....49
3.2 Nemónicos para la función SH.....53
3.3 Nemónicos para la función AH.....56
3.4 Nemónicos de las funciones T o TE.....59
3.5 Nemónicos de las funciones L o LE.....61
3.6 Codificación de las direcciones Talk y Listen.....70

LISTA DE FIGURAS

Figura Pag.

1.1 Disposición del Bus IEEE488....4
1.2 Estructura de las líneas del bus IEEE-488....5
1.3 Formato de la trasmisión de dato en la GPIB....6
1.4 Secuencia de transferencia de información (Handshake) de la GPIB....13
1.5 Línea de carga en DC para los manejadores colector abierto de la GPIB....21
1.6 Circuito de entrada-salida típico de una línea de señal de GPIB....22
1.7 Conector del Cable GPIB....23
1.8 Configuraciones del Cableado de la GPIB....23
1.9 Conector para IEEE y ANSI....24
1.10 Conector para IEC 625-1....24
2.1 Diagrama de bloques de la interconexión de la computadora y la interface 29
2.2 Descripción del Pin-Out de los canales de expansión....29
2.3 Diagrama de bloques del 8255A....32
2.4 Byte de Control....33
2.5 Lógica de acceso al 8255A....35
2.6 Circuito de Interface del bus IEEE-488 con el 8255A....36
2.7 Adaptación de las líneas handshake....38
2.8 Esquema de la tarjeta impresa....41
2.9 Esquema de los componentes de la Interface sobre la tarjeta impresa....42
2.10 Conexión recomendada de los cables a la interfaz....43
3.1 Representación de un estado para una función 47
3.2 Representación de tres estados y sus transiciones....48
3.3 Representación del inicio de una función de interfaz (a)representación completa (b)notacion abreviada 50
3.4 Diagrama de estado de la función SH....51
3.5 Diagrama de estado de la función AH....54
3.6 Diagrama de estado de la función T....57
3.7 Diagrama de estado para la función (a)L y (b)LE 60
3.8 Diagrama de estado de la función C 62
3.9 Diagrama de flujo de la Secuencia Handshake....65
3.10 Aspecto de los interruptores para colocar la dirección del dispositivo 69
4.1 Diagrama de flujo del programa principal 80
4.2 Flujograma del procedimiento Configurar 82
4.3 Flujograma del procedimiento Operaciones 83

CAPITULO I
BUS DE INTERFACE IEEE 488.

Introducción

En este capitulo se pretende describir la mayor parte de los conceptos generales relacionados con el bus estandar IEEE-488, los cuales se encuentran en el documento del estandar, se hace una breve reseña historica, una descripción también breve sobre su funcionamiento, sus protocolos, sus funciones y requerimientos de operación.

1.1 Generalidades

Un bus es un circuito o un grupo de lineas sobre la cual datos, direcciones, señales de control o potencia es diseminada. Estos, frecuentemente, tienen una naturaleza bidireccional, en las que pueden existir diferentes conexiones de fuente y destino. En un sistema de microcomputador se encuentran, generalmente muchos buses, por eso el bus es un componente fundamental del sistema, llamandosele al set principal de buses como Bus del Sistema.

Así, muchas diferentes configuraciones de bus existen aún para sistemas basados en la misma familia de componentes de microprocesador. Aunque algunos de estos, están tan pobremente trazados, que excluyen el uso de cualquier otro tipo de microprocesador para el sistema CPU principal.

Por otro lado, el concepto de un Bus del Sistema concierne realmente al problema de la comunicación entre dispositivos. A continuación se presentan algunos buses estandar conocidos:

A. Buses Internos

1. MULTIBUS (Intel)
2. LSI-11 (DEC)
3. UNIBUS (DEC)
4. Q-BUS (DEC)
5. EXORCISER (Motorola)
6. Zilog MCB
7. PRO-LOG
8. microNOVA
9. TI990
10. CAMAC
11. NuBus (Apple)
12. S-100

B. Buses Externos

1. IEEE 488
2. RS-232C
3. RS-449

Del primer grupo destacan el MULTIBUS de Intel y el CAMAC como los más rigurosamente definidos y bien diseñados, por otro lado se encuentra el S-100 como uno de los menos especificados y no está bien diseñado debido a que fué impuesto por los usuarios del 8080 como una necesidad para suplir un bus común de 100 lineas para sistemas basados en el 8080 y el Z80.

El RS-232C es el clásico bus de interface serie binario, es el bus estandar de comunicación serie más ampliamente usado, sin embargo su uso está limitado a longitudes de cable relativamente cortas (50 pies o 15 metros) y un promedio de datos de hasta 20 Kbits/segundo.

El IEEE 488 es en cambio, el bus estandar de comunicación paralelo más ampliamente usado.

1.2 Historia del Bus de Interface IEEE 488

El bus de interface IEEE 488 nace como una especificación técnica de la Hewlett-Packard en 1965, estos necesitaban de medios de interconexión en control automático y transferencia de datos, dentro de sistemas con equipos electrónicos de prueba por lo que crearon el HP-IB (Hewlett-Packard Interface Bus).

En 1972 se adopta en los Estados Unidos el concepto de interface utilizado por Hewlett-Packard, como punto de partida y fué aprobado por la IEC (International Electrotechnical Committe) en ese mismo año.

En 1976 fué aprobada por los paises miembros de IEC, la recomendación 625-1, esta definición, utilizando un conector diferente es totalmente compatible con la definición del HP-IB. La que es publicada en 1980 con el nombre de "Un sistema de interface para Aparatos de Medición Programable (Byte Serie Bit Paralelo)".

Por otro lado, la IEEE Standards Board aprobó en 1975 el Estandar IEEE 488, con el nombre de "Interface Digital para Instrumentación Programable", siendo completamente compatible con la definición HP-IB.

En 1976, ANSI (American Standard Interfacing System) adoptó la Estandar IEEE 488 y lo publicó como la ANSI Standard MC1.1.

Una revisión del IEEE 488 ocurrió en noviembre de 1978, principalmente para aclaración de conceptos; una nueva revisión se llevó a cabo en 1982 para recomendaciones prácticas en conveciones de código y formato, y la última publicación en 1987 con menores cambios editoriales, conociendose como las publicaciones Standard 488.1 y 488.2, siendo la segunda un complemento de la primera, en la que están contenidos los codigos, formatos, protocolos y comandos comunes.

Este bus es el mas usado internacionalmente y se le conoce en varias versiones:

a. HP-IB (Hewlett-Packard Interface Bus)
b. GPIB (General Purpouse Interface Bus)
c. IEEE BUS
d. ASCII BUS
e. PLUS BUS

Todos ellos describen lo mismo: un ordenado y predecible modo para intercambiar datos y mantener control entre los dispositivos de instrumentación.

Además está definido por los cuatro mayores estandares mundiales:

1. IEEE 488
2. ANSI MC 1.1 (Idéntico al IEEE 488)
3. IEC 625-1 (Idéntico al IEEE 488 excepto el conector)
4. B.S. 6146 (British Standard-Identico al IEC 625-1)

Debido a sus altos rangos de transferencia de datos, de 250 KBytes a 1 MegaByte por segundo, este bus rapidamente ganó popularidad en otras aplicaciones tales como: comunicación entre computadoras y control de perifericos.

1.3 Conceptos Generales del Bus IEEE 488 o GPIB

El IEEE 488 o GPIB (Bus de Interfase de Proposito General) es un de 24 lineas de las cuales 16 lineas son de señal y 8 lineas son de conexión a tierra

Las 16 lineas de señal están agrupadas en 8 lineas de datos, 3 lineas de comunicación y 5 lineas de manejo de la interface.

Estas lineas son usadas para interconectar dispositivos en un arreglo paralelo y mantener ordenadamente un flujo de información relacionado al dispositivo y la interface.

Son tres dispositivos los que están definidos por el IEEE 488, para organizar y manejar el flujo de información en el bus:

a. TALKERS (Conversador o Emisor): Son dispositivos que envían información, tal como datos, instrucciones de programas y estados a los LISTENERS a través de la interface, cuando es direccionado. Ejemplo de estos tipos de dispositivos son: voltimetros, contadores, discos opticos,etc.

b. LISTENERS (Escuchas o Receptores): Son dispositivos que se pueden direccionar para recibir datos y mensajes a través de la interface. Ejemplo de estos tipos de dispositivos son: impresores, graficadores, dispositivos display, fuentes

de potencia programables, generadores de señal programables, etc.

c. CONTROLLERS (Controladores): Son dispositivos que manejan la GPIB, direccionando los dispositivos complementarios para que sean TALKERS o LISTENERS, y comandándolos para desarrollar otras funciones internas. Una computadora con una tarjeta apropiada I/O es un ejemplo de un controlador.

La GPIB emplea un bus de 24 lineas para conectar hasta 15 dispositivos usualmente en lugares cercanos a la computadora (ver figura 1.2).

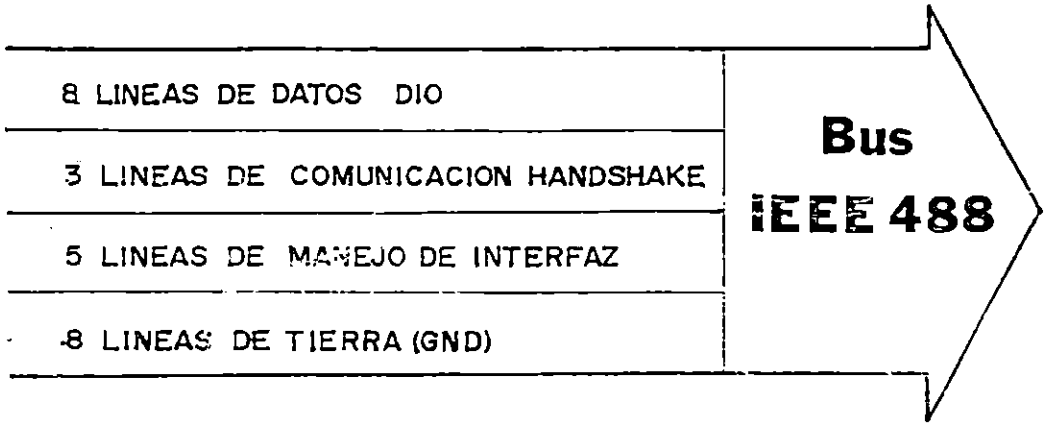


Figura 1.1 Disposición del Bus IEEE 488.

Como se mencionó anteriormente los dispositivos pueden ser emisores, receptores y/o controladores. Un voltimetro digital por ejemplo, es un Emisor y puede ser también Receptor.

Un Emisor ó Conversador envía datos a uno o más Receptores o Escuchas, el cual recibe datos. El Controlador maneja el flujo de información en la GPIB enviando comandos a todos los dispositivos.

Con la interface GPIB y los dispositivos de programación apropiados, el usuario de computadoras puede desempeñar las tres funciones de EMISOR, RECEPTOR y CONTROLADOR.

La GPIB utiliza un bus ordinario de computadora, excepto por que la computadora tiene sus circuitos interconectados por medio de un bus plano, mientras que la GPIB tiene dispositivos interconectados por medio de un bus cable, pero que básicamente son equivalentes.

El papel del controlador GPIB se puede comparar con el centro de interrupciones de un sistema telefónico de una ciudad. En este caso, el centro de interrupciones (Controlador) maneja la red de comunicaciones (GPIB). Cuando

el centro (Controlador) notifica que una parte (Dispositivos) quiere hacer una llamada (enviar un dato), conecta a la persona que llama (Conversador ó Emisor) con el receptor (Escucha ó Receptor).

El Controlador, usualmente direcciona un Emisor (Talker) y un Receptor (Listener) antes que el Emisor pueda enviar su mensaje al Receptor. Después que el mensaje se transmite, usualmente el controlador anula la dirección de ambos dispositivos. Algunas configuraciones no requerirán de un Controlador. Por ejemplo, un dispositivo que siempre debe ser un Emisor (dispositivo denominado ONLY TALK) conectado en un bus con uno o más dispositivos solamente Receptor (ONLY LISTEN).

1.4 Lineas de Señal de la GPIB

Como se mencionó anteriormente este bus consta de 16 lineas, las que se dividen en tres grupos de lineas:

- Lineas de Datos
- Lineas de Comunicación o Handshake
- Lineas de Manejo de la Interface

Así como en otros buses estandar, se utiliza lógica negativa.

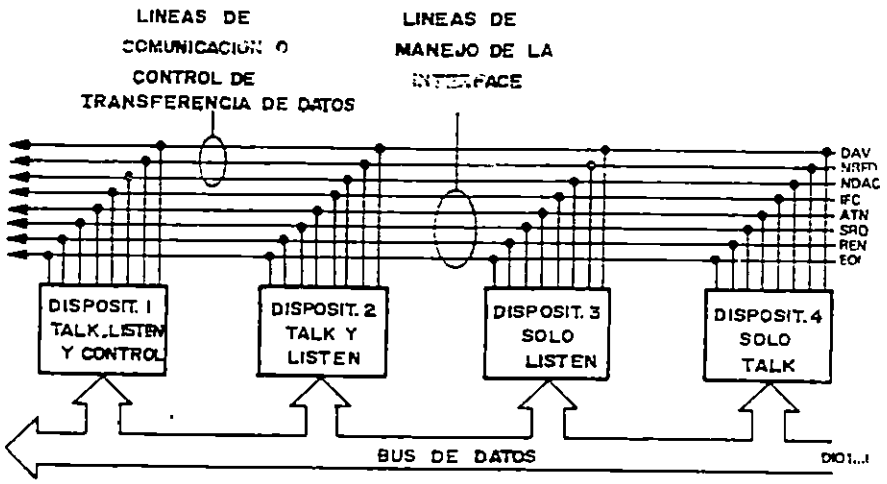


Figura 1.2 Estructura de las lineas del bus IEEE 488.

1.4.1 Lineas de Datos.

Las 8 lineas de datos, son identificadas como DIO1 hasta DIO8, las cuales llevan mensajes de datos o de comandos.

Todos los comandos y la mayoría de los datos usan el bit 7 del código ASCII (American Standard Code for Information Interchange). El equivalente internacional es el bit 7 del código ISO (International Standard Organization). No obstante, otras técnicas de codificación pueden ser empleadas para comprimir información en estas 8 lineas.

Por ejemplo, para transferir la secuencia "BUS" de 3 bytes, como se observa en la figura 1.3 , presenta un formato de BIT PARALELO - BYTE SERIE.

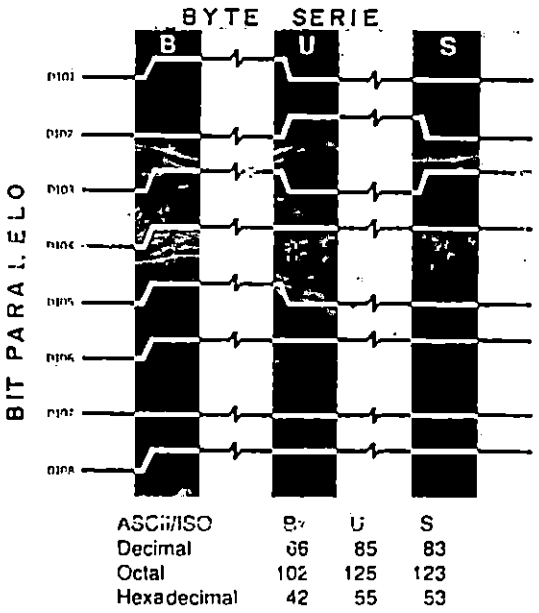


Figura 1.3 Formato de la transmisión de dato en la GPIB

Se transmite un byte a la vez codificado como se muestra,pero cada byte está formado por 8 bits en paralelo.

1.4.2 Lineas de Comunicación o Handshake.

Son tres lineas las que controlan la transferencia de los mensajes entre los dispositivos, el proceso es llamado "Comunicación entrelazada de 3 alambres", y garantiza que los mensajes en las lineas de datos sean llevados y recibidos sin transmisión de error.

Estas lineas de comunicación son:

a) DAV (Data Valid):
Indica cuando las señales en la linea de datos son válidas y pueden ser aceptadas con seguridad por los dispositivos.
El CONTROLADOR maneja las lineas DAV cuando envía comandos y el EMISOR maneja las lineas DAV cuando envía datos. Es activado por la fuente cuando el dato es colocado y validado , y si NRFD (condición alta) ha sido sensado.

b) NRFD (Not Ready For Data):
Usado para indicar cuando un dispositivo está o no listo para recibir un byte de mensaje. Esta línea es manejada por todos los dispositivos cuando reciben comandos y por los RECEPTORES cuando reciben datos.

Un RECEPTOR activa su NRFD (condición baja) para indicar que no está listo para aceptar datos, y libera esta línea cuando está listo para aceptar datos.

c) NDAC (Not Data Accepted):
Indica cuando un dispositivo ha aceptado o no, un mensaje. Esta línea es manejada por todos los dispositivos cuando reciben comandos y por RECEPTORES cuando recibe datos. El receptor activa su NRFD (condición baja) para indicar que no ha aceptado el dato y cuando se acepta un dato desde las líneas DI/O, este liberará su línea NDAC.

1.4.3 Líneas de Manejo de la Interface.

Son cinco líneas las que se utilizan para manejar el flujo de información a través de la interface. Estas líneas son:

a) ATN (Attention):
El CONTROLADOR maneja líneas ATN; "CIERTO" cuando usa líneas de datos para envíar comandos y "FALSO" cuando permite que un EMISOR envíe mensajes como datos.

Todos los dispositivos deben monitorear la línea ATN, todo el tiempo y responder a esta dentro de 200 nseg.

Cuando está en "CIERTO", ATN coloca a la interface en el "MODO COMANDO", donde todos los dispositivos aceptan (secuencia handshake) datos en las Líneas de Datos y los interpreta como COMANDOS o DIRECCIONES.

Cuando está en "FALSO", ATN coloca a la interface en el "MODO DATO" donde el emisor activo en ese momento, envía datos a todos los receptores.

MODO COMANDO (ATN Cierto)

Los comandos sirven para diversos propositos, como seleccionar las direcciones de EMISOR o RECEPTOR de los instrumentos que serán fuente o receptores de datos. Siendo todos ellos mensajes de multilínea (mensajes enviados sobre el bus de datos). Las direcciones son enviadas a todos los dispositivos.

Se definen además otros tres tipos de comandos: Comandos Universales, Comandos Direccionados y Comandos Secundarios. Los que más adelante se describen con detalle.

MODO DATO (ATN Falso)
En el Modo Dato, el dato de los dispositivos dependientes (datos de programa, datos de medida o datos de estado) es envíado desde el emisor activo al receptor activo sobre la inteface.

b) IFC (Interface Clear):
El sistema CONTROLADOR maneja la línea IFC para inicializar el bus y convertirse en CIC (Controller In Charge ó Controlador en Mandato). Esta línea es usada para operaciones de interrupción corriente en el bus.
Todos los dispositivos deben monitorear IFC todo el tiempo y responder dentro de 100 µseg. (ancho mínimo del pulso para IFC).

c) REN (Remote Enable):
El sistema CONTROLADOR maneja la línea REN para colocar los dispositivos en modo de programación local ó remoto. Cuando la línea es "Cierta", todos los receptores capacitados de operación remoto son colocados en dicho estado.
Cuando la línea es "Falso" todos los dispositivos retornan a operación local. Todos los dispositivos habilitados de ambas operaciones, remoto y local, deben monitorear la línea REN todo el tiempo y responder a esta dentro de los 100 µseg. siguientes.

d) SRQ (Service Request):
Cualquier dispositivo puede manejar la línea SRQ para solicitar servicio del controlador y puede actuar como una interrupción de la secuencia actual de eventos.
Tipicamente SRQ indica que un dato está listo para transmitirse y/o una condición de error existe (error de sintaxis, sobrecarga, disparo demasiado rapido, etc).
El controlador puede enmascarar la interrupción SRQ y debe realizar un Sondeo (POLL) de los dispositivos para determinar quíen requiere el servicio y porque. SRQ es limpiado o inicializado por un Sondeo Serie (Serial Poll) solamente.

Existen dos procedimientos de sondeo:

1. Sondeo Serie (SERIAL POLL)
2. Sondeo Paralelo (PARALLEL POLL)

SONDEO SERIE: Es una secuencia que habilita al controlador para conocer si un dispositivo o grupo de dispositivos requiere servicio y/o determinar el estado multi-bit de dispositivos en la interface.

Los dispositivos que pueden ser sondeados en serie, retornarán un BYTE de ESTADO al CONTROLADOR para indicar su estatus dentro del control del programa.

El Controlador secuencialmente sondea cada dispositivo individual en la interface (enviando un comando SPE- Serial Poll Enable- si IFC es falso y secuencialmente direccionando los dispositivos para transmitir) y evalúa cada byte de estatus por vez.

Este procedimiento puede ser prolongado en sistemas muy grandes, pero provee la naturaleza del requerimiento al mismo tiempo que la identidad del dispositivo.

SONDEO PARALELO: Es una operación iniciada por el Controlador para obtener información de los dispositivos.

Cuando el Controlador inicia un Sondeo Paralelo, cada dispositivo regresa un BIT de ESTADO vía una de las líneas DIO.
Las asignaciones de dispositivo DIO, son hechas por interruptores o uniones, o por el Controlador durante la secuencia PPC (Parallel Poll Configure - Configurar Sondeo Paralelo)

Los dispositivos responden individualmente, cada uno en una línea DIO separada, o colectivamente en una sola línea DIO o alguna combinación de ambos. Cuando responden colectivamente, el resultado es una operación lógica AND u OR de los grupos de bits de estatus.

Los dispositivos configurados deben responder a un Sondeo Paralelo (la aceptación simúltanea de ATN y EOI) dentro de 200 nanosegundos. El Controlador puede, entonces, leer los resultados del sondeo 2 µseg. más tarde.

El Sondeo Paralelo a menudo es usado en el mundo de las computadoras, para chequear el estado de las acciones siguientes: qué periféricos están listos para datos o recibiendo datos.

El conocimiento de esta información contribuye a que los tiempos muertos sean reducidos y el ancho de banda del sistema es usado mas eficientemente.
e) EOI (End or Identify):

La línea EOI tiene dos propositos:
1. El EMISOR usa la línea EOI para marcar el final de un mensaje enlazado.

2. El CONTROLADOR usa la línea EOI para explicar a los dispositivos a identificar, sus respuestas en una consulta paralela.

Cuando ATN es "cierto" la línea EOI es usada por un controlador para ejecutar un Sondeo Paralelo. Cuando ATN es "falso", la línea EOI es usada por un emisor activo para indicar el último byte de un mensaje de datos.

1.5 Operación del bus IEEE 488 ó GPIB

1.5.1 Funciones de Interface

El Estandar IEEE 488 especifica que hasta once funciones de interface pueden ser incorporadas dentro de la interface GPIB. Estas funciones de interface no necesitan estar todas implementadas dentro de un dispositivo particular. Y una de esas funciones de interface, la de controlador es raramente implementada por más de un dispositivo dentro del sistema.
El total de funciones disponibles son:

SOURCE HANDSHAKE (SH)
Provee al dispositivo la capacidad de transferir correctamente mensajes de multilínea. Para transmitir asíncronamente estos mensajes, se realiza una secuencia entrelazada de comunicación (handshake) entre un dispositivo que contiene una función SH y uno ó más dispositivos que contienen la función Acceptor Handshake.

ACCEPTOR HANDSHAKE (AH)
Provee al dispositivo la capacidad de garantizar la recepción apropiada de mensajes remotos de múltiple línea.

TALKER ó EXTENDER TALKER (T ó TE)
Es la capacidad de un dispositivo para ser un Conversador ó Emisor. Esta puede ser un byte de dirección o en el caso de dirección extendida (Extended Adress) una dirección de dos bytes.

LISTEN ó EXTENDED LISTEN (L ó LE)
Es la capacidad de un dispositivo para ser un Escucha ó Receptor. Su dirección puede ser de un byte ó de dos bytes para dirección extendida.

SERVICE REQUEST (SR)
Esta capacidad permite al dispositivo solicitar asíncronamente servicio desde el controlador. También, se sincroniza el bit de requerimiento de servicio (SR) del byte de estado durante un sondeo serie.

REMOTE/LOCAL (RL)
Provee la capacidad para seleccionar entre dos fuentes de entrada de información. La información de local corresponde a los controles de la parte delantera del dispositivo y la información de remoto corresponde a la entrada de información desde el bus.

PARALLEL POLL (PP)
Provee la capacidad al dispositivo únicamente para identificar por sí mismo, si requiere servicio, cuando el Controlador está preguntando. Esta capacidad difiere de SR, en que necesita una comisión de parte del controlador para conducir periódicamente una consulta paralela.

DEVICE CLEAR (DC)
Esta función permite al dispositivo inicializarse para un estado predefinido. Un dispositivo con esta capacidad tendrá el efecto de este comando, descrito en su manual de operación.

DEVICE TRIGGER (DT)
Esta función permite al dispositivo tener inicializada su operación básica, por el Emisor en el bus.

CONTROLLER (C)
Esta función permite al dispositivo enviar direcciones, comandos universales y comandos direccionados a otros dispositivos.

DRIVERS (D)
Este puede definirse mas como un código que describe el tipo de manejadores eléctricos usados en el dispositivo.

1.5.2 Comunicación entre dispositivos (Secuencia Handshake)

La Secuencia Handsake se realiza através de las 3 líneas de comunicación del bus (DAV, NRFD y NDAC).

Estas 3 líneas se utilizan para coordinar la transferencia de datos sobre el bus de datos desde una fuente (que puede ser un emisor ó conversador direccionado ó un controlador) hacia un aceptor ó receptor (un escucha ó receptor direccionado ó todos los dispositivos que están recibiendo comandos de interface) para asegurar la integridad en la transferencia de datos.

Esta técnica tiene las siguientes características:

1. La tranferencia de datos es asíncrona y el promedio de transferencia automáticamente se ajusta a la velocidad del emisor y receptor(es) y corre en el promedio de velocidad del dispositivo mas lento.

2. Mas de un dispositivo puede aceptar el dato al mismo tiempo.

3. Cada byte transferido atraviesa por el handshake. (Escepto para respuesta a un Sondeo Paralelo)

4. Cuando los comandos universales son enviados sobre el bus de datos, el dispositivo mas lento en el bus determinará el retardo de transferencia de ese comando.

5. El actual retardo de transferencia del dato es afectado también, por el tiempo que se toma el instrumento para hacer la lectura y el tiempo necesario para que el controlador introduzca la información.

Las líneas de señal, utilizan convención de lógica negativa (Cierto: Estado Bajo), lo que permite reducir la susceptibilidad al ruido en el estado cierto (1 lógico = Cero Volts).

La Secuencia Handshake está formada por cuatro estados fuente y cuatro estados aceptores ó receptores.

Los estados fuente son:

1. SGNS (Source Generate State): Estado de generación de fuente.

2. SDYS (Source Delay State): Estado de retardo de la fuente.

3. STRS (Source Transfer State): Estado de transferencia de la fuente.

4. SWNS (Source Wait for New Cycle State): Estado de espera de la fuente para nuevo ciclo.

Los estados aceptores son:

1. ANRS (Acceptor None Ready State): Estado de ningún aceptor o receptor listo.

2. ACRS (Acceptor Ready State): Estado de aceptor ó receptor listo.

3. ACDS (Accept Data State): Estado de aceptación de dato.

4. AWNS (Acceptor Wait for New Cycle State): Estado de espera del receptor para nuevo ciclo.

La secuencia de transferencia (handshake) de un byte, desde un emisor hasta un receptor, se muestra en la siguiente figura.

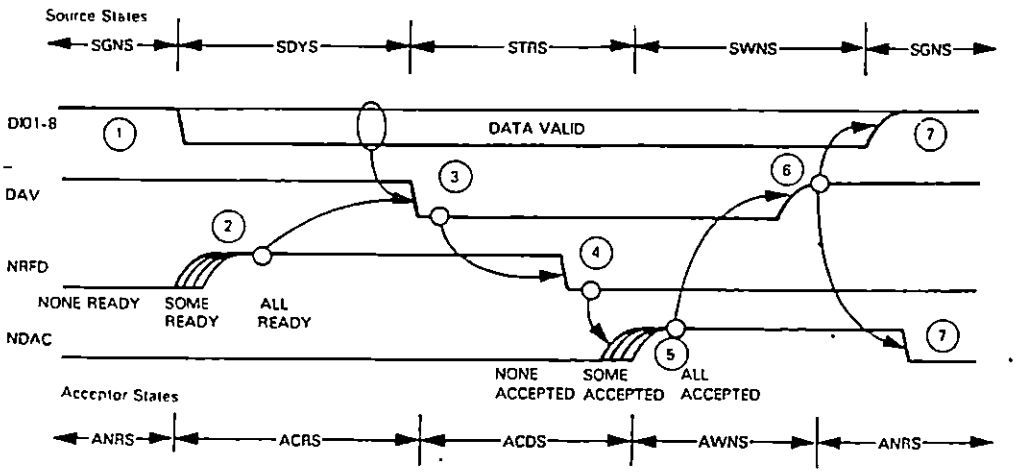


Figura 1.4 Secuencia de transferencia de información (Handshake) de la GPIB.

Esta secuencia comienza con la fuente en el estado SGNS. Aquí, la fuente no está colocando algún dato dentro de las líneas de datos, así que esta situación causa que todas las lineas vayan a un nivel alto. Los receptores están colocando en condición cierta (nivel bajo) las líneas NRFD y NDAC, como se indica en (1).

En el punto (2), vemos que todos los receptores están listos para aceptar datos, también la fuente ha colocado un byte de dato en el bus de datos y ha entrado al estado SDYS, permitiendo que el dato sea fijado. La fuente podría colocar en la línea "EOI" la condición "cierto" si este fuese el último byte en el mensaje.

Despues de el retardo de fuente, esta entra al estado STRS, indicado en (3). La línea DAV (Data Valid) es colocada en condición "cierta" (nivel bajo), para indicar que el dato ha sido enviado. Los receptores están actualmente en el estado ACRS (Estado del receptor listo).

Los receptores, entonces, se mueven dentro del estado ACDS para colocar de nuevo en condición "cierta" la línea NRFD, como se indica en (4).

La fuente se mueve al estado SWNS, de espera para un nuevo ciclo, y los receptores niegan la línea NDAC (condición "falsa") y se desplazan al estado AWNS de espera para un nuevo ciclo, como se indica en (5).

Con todos los receptores habiendo acptado el dato, la fuente libera la línea DAV (condición "falso")como se observa en (6).
Con la línea DAV inactiva (condición "falsa"), la fuente retorna a su estado inicial (SGNS) indicado en (7). Mientras

tanto los receptores colocan NDAC en condición "cierta" y retornan a su estado inicial (ANRS).
De esta manera, el ciclo puede realizarse de nuevo.

1.5.3 Tipos de Mensajes de la GPIB

La comunicación entre los dispositivos interconectados se realiza por el paso de mensajes a traves del sistema de interface. La GPIB lleva mensajes de dispositivos dependientes y mensajes de la interface.

Las definiciones de mensajes de dispositivos dependientes y mensajes de interface son:

a. Mensajes de dispositivos dependientes: Muchas veces llamados datos o mensajes de datos, contienen información de dispositivos especificos tales como; instrucciones de programa, resultados de medidas, estados de máquina y datos de archivo.

b. Mensajes de interface: Manejan el bus y estos son usualmente llamados COMANDOS o mensajes de COMANDOS.
Los mensajes de interface desarrollan funciones tales como: inicializar el bus, direccionar dispositivos y colocar los dispositivos en modos de programación remoto ó local.

De estos los más importantes son los mensajes de interface ó comandos, los que se dividen en 3 grupos:

1. Comandos Universales
2. Comandos Direccionados
3. Comandos Secundarios

Comandos Universales:

Están divididos en 5 comandos de multilínea y 4 comandos de unilínea ó una línea.

Existen cinco tipos de comandos de multilínea universales:

a. Device Clear (DCL)
Causa que todos los dispositivos reconocidos retornen a un estado predefinido de dispositivo dependiente.

b. Local Lockout (LLO)
Deshabilita una función particular de reset local de panel frontal ó panel trasero, retornando a control local a los dispositivos que reconocen este comando.

c. Serial Poll Enable (SPE)

Establece el modo de sondeo serie, para que todos los dispositivos conversadores (talkers) respondan en el bus. Cuando son direccionados para emisor, cada dispositivo retornará un simple byte de estatus de 8 bits.

d. Serial Poll Disable (SPD)
Deshabilita el modo de Sondeo Serie para todos los dispositivos que pueden responder a este comando.

e. Parallel Poll Unconfigure (PPU)
Restablece todos los dispositivos con Sondeo Paralelo al estado ocioso ó inactivo (deshabilitado para responder a un Sondeo Paralelo).

Aunque también se incluyen los comandos Untalk (UNT) y Unlisten (UNL) dentro de los comandos de multilínea, estos son clasificados como direcciones, porque su fin es actuar sobre las direcciones de determinados dispositivos.

Untalk: deshabilita el actual emisor direccionado. Si se colocase una dirección de emisor inusual ,se cumpliría el mismo objetivo.

Unlisten: deshabilita los actuales receptores direccionados en el bus. Un solo receptor no puede ser deshabilitado, en este caso, sin dehabilitar todos los demás, garantizandose que solamente los receptores deseados serán direccionados.

En cuanto a los cuatro comandos de unilínea,estos se muestran en la tabla siguiente:

Tabla 1.1 Comandos de Unilínea

Comando de Unilínea	Línea de Manejo de la Interface
Interface Clear	IFC
Remote Enable	REN
Attention	ATN
Identify (IDY)	EOI + ATN

Comandos de Direccionamiento:

Un dispositivo puede estar o no diseñado para responder a algún comandos de estos en particular. Este grupo de comandos consta de cinco tipos:

a. Group Execute Trigger (GET)
Este comando provoca que todos los dispositivos con la capacidad de operar con este comando y que están actualmente direccionados para escuchar, inicien una

acción pre-programada (disparo, procesar un barrido, etc). Algunos dispositivos, también pueden reconocer un caracter de dato de dispositivo dependiente o string, para esta función. El comando GET provee un medio de gatillar dispositivos simultaneamente.

b. Selected Device Clear (SDC)
Este comando resetea el dispositivo actualmente direccionado para escucha a un estado de dispositivo dependiente (estado de encendido, abrir todos los relés, etc.)

c. Go to Local (GTL)
Provoca que el dispositivo actualmente direccionado para escuchar, retorne al control local del panel (salida del estado remoto), el dispositivo retornará al estado remoto cuando sea direccionado de nuevo para escucha ó receptor.

d. Parallel Poll Configure (PPC)
Provoca que el receptor direccionado sea configurado de acuerdo al comando secundario PPE (Habilitar Sondeo Paralelo) que vá a continuación.

e. Take Control (TCT)
Permite que el controlador , tome el control del bus como parte de una acción pre-programada.

Comandos Secundarios:

Este grupo está formado por las casillas mas bajas, correspondientes a la tabla de carácteres ASCII (ver Tabla 1.2) y son usados para direcciones extendidas de Emisor (Talk) y Receptor (Listen) y comandos de Sondeo Paralelo Secundario.

Estos comandos se utilizan en conjunto con una dirección, un comando universal o un comando de direccionamiento (también llamados comandos primarios) para extender el espacio de código cuando es necesario.

1.5.4 Codificación de los Mensajes de Interface

En la tabla 1.3 se resumen todos los mensajes de la interface utilizados por la GPIB. Es de notar que mas de un mensaje puede ser puesto simultaneamente por diferentes funciones de interface. Para cada mensaje en la tabla se lista solamente la información especifica a ese mensaje.

Las siguientes abreviaciones son usadas en la Tabla 1.3:

AC : Comando de Direccionamiento
AD : Dirección

Tabla 1.2
Carta de Código ASCII/ISO e IEEE

B7 B6 B5 / BITS / B4 B3 B2 B1	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
	CONTROL		NUMBERS SYMBOLS		UPPER CASE		LOWER CASE	
0 0 0 0	0 NUL 0 0 0	20 DLE 10 16	40 SP 20 32	60 0 30 48	100 @ 40 64	120 P 50 80	140 ` 60 96	160 p 70 112
0 0 0 1	1 GTL SOH 1 1	21 LLO DC1 11 17	41 ! 21 33	61 1 31 49	101 A 41 65	121 Q 51 81	141 a 61 97	161 q 71 113
0 0 1 0	2 STX 2 2	22 DC2 12 18	42 " 22 34	62 2 32 50	102 B 42 66	122 R 52 82	142 b 62 98	162 r 72 114
0 0 1 1	3 ETX 3 3	23 DC3 13 19	43 # 23 35	63 3 33 51	103 C 43 67	123 S 53 83	143 c 63 99	163 s 73 115
0 1 0 0	4 SDC EOT 4 4	24 DCL DC4 14 20	44 $ 24 35	64 4 34 52	104 D 44 68	124 T 54 84	144 d 64 100	164 t 74 116
0 1 0 1	5 PPC ENQ 5 5	25 PPU NAK 15 21	45 % 25 37	65 5 35 53	105 E 45 69	125 U 55 85	145 e 65 101	165 u 75 117
0 1 1 0	6 ACK 6 6	26 SYN 16 22	46 & 26 38	66 6 36 54	106 F 46 70	126 V 56 86	146 f 66 102	166 v 76 118
0 1 1 1	7 BEL 7 7	27 ETB 17 23	47 ' 27 39	67 7 37 55	107 G 47 71	127 W 57 87	147 g 67 103	167 w 77 119
1 0 0 0	10 GET BS 8 8	30 SPE CAN 18 24	50 (28 40	70 8 38 56	110 H 48 72	130 X 58 88	150 h 68 104	170 x 78 120
1 0 0 1	11 TCT HT 9 9	31 SPD EM 19 25	51) 29 41	71 9 39 57	111 I 49 73	131 Y 59 89	151 i 69 105	171 y 79 121
1 0 1 0	12 LF A 10	32 SUB 1A 26	52 * 2A 42	72 : 3A 58	112 J 4A 74	132 Z 5A 90	152 j 6A 106	172 z 7A 122
1 0 1 1	13 VT B 11	33 ESC 1B 27	53 + 2B 43	73 ; 3B 59	113 K 4B 75	133 [5B 91	153 k 6B 107	173 { 7B 123
1 1 0 0	14 FF C 12	34 FS 1C 28	54 , 2C 44	74 < 3C 60	114 L 4C 76	134 \ 5C 92	154 l 6C 108	174 | 7C 124
1 1 0 1	15 CR D 13	35 GS 1D 29	55 - 2D 45	75 = 3D 61	115 M 4D 77	135] 5D 93	155 m 6D 109	175 } 7D 125
1 1 1 0	16 SO E 14	36 RS 1E 30	56 . 2E 46	76 > 3E 62	116 N 4E 78	136 ^ 5E 94	156 n 6E 110	176 ~ 7E 126
1 1 1 1	17 SI F 15	37 US 1F 31	57 / 2F 47	77 UNL ? 3F 63	117 O 4F 79	137 UNT _ 5F 95	157 o 6F 111	177 RUBOUT (DEL) 7F 127
	ADDRESSED COMMANDS	UNIVERSAL COMMANDS	LISTEN ADDRESSES		TALK ADDRESSES		SECONDARY ADDRESSES OR COMMANDS	

octal 25 PPU — Message Mnemonic
NAK — ASCII/ISO character
hex 15 21 — decimal

Tabla 1.3
Codificación de los Mensajes de Interface del Bus IEEE 488

Nemónico	Nombre del Mensaje	Tipo	Clase	DI08	DI07	DI06	DI05	DI04	DI03	DI02	DI01	DAV	NRFD	NDAC	ATN	EOI	SRQ	IFC	REN
ACG	addressed command group	M	AC	X	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X
ATN	attention	U	UC	X	X	X	X	X	X	X	X	X	X	X	1	X	X	X	X
DAB	data byte	M	DD	D8	D7	D6	D5	D4	D3	D2	D1	X	X	X	X	X	X	X	X
DAC	data accepted	U	HS	X	X	X	X	X	X	X	X	X	X	0	X	X	X	X	X
DAV	data valid	U	HS	X	X	X	X	X	X	X	X	1	X	X	X	X	X	X	X
DCL	device clear	M	UC	X	0	0	1	0	1	0	0	X	X	X	X	X	X	X	X
END	end	U	ST	X	X	X	X	X	X	X	X	X	X	X	X	1	X	X	X
EOS	end of string	M	DD	E8	E7	E6	E5	E4	E3	E2	E1	X	X	X	X	X	X	X	X
GET	group execute trigger	M	AC	X	0	0	0	1	0	0	0	X	X	X	X	X	X	X	X
GTL	go to local	M	AC	X	0	0	0	0	0	0	1	X	X	X	X	X	X	X	X
IDY	identify	U	UC	X	X	X	X	X	X	X	X	X	X	X	X	1	X	X	X
IFC	interface clear	U	UC	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1	X
LAG	listen address group	M	AD	X	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X
LLO	local lock out	M	UC	X	0	0	1	0	0	0	1	X	X	X	X	X	X	X	X
MLA	my listen address	M	AD	X	0	1	L5	L4	L3	L2	L1	X	X	X	X	X	X	X	X
MTA	my talk address	M	AD	X	1	0	T5	T4	T3	T2	T1	X	X	X	X	X	X	X	X
MSA	my secondary address	M	SE	X	1	1	S5	S4	S3	S2	S1	X	X	X	X	X	X	X	X
NUL	null byte	M	DD	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X
OSA	other secondary address	M	SE	(OSA = SCG ∧ $\overline{MSA}$)															
OTA	other talk address	M	AD	(OTA = TAG ∧ $\overline{MTA}$)															
PCG	primary command group	M	—	(PCG = ACG ∨ UCG ∧ LAG ∧ TAG)															
PPC	parallel poll configure	M	AC	X	0	0	0	0	1	0	1	X	X	X	X	X	X	X	X
PPE	parallel poll enable	M	SE	X	1	1	0	S	P3	P2	P1	X	X	X	X	X	X	X	X
PPD	parallel poll disable	M	SE	X	1	1	1	D4	D3	D2	D1	X	X	X	X	X	X	X	X
PPR1	parallel poll response 1	U	ST	X	X	X	X	X	X	X	1	X	X	X	X	X	X	X	X
PPR2	parallel poll response 2	U	ST	X	X	X	X	X	X	1	X	X	X	X	X	X	X	X	X
PPR3	parallel poll response 3	U	ST	X	X	X	X	X	1	X	X	X	X	X	X	X	X	X	X
PPR4	parallel poll response 4	U	ST	X	X	X	X	1	X	X	X	X	X	X	X	X	X	X	X
PPR5	parallel poll response 5	U	ST	X	X	X	1	X	X	X	X	X	X	X	X	X	X	X	X
PPR6	parallel poll response 6	U	ST	X	X	1	X	X	X	X	X	X	X	X	X	X	X	X	X
PPR7	parallel poll response 7	U	ST	X	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X
PPR8	parallel poll response 8	U	ST	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
PPU	parallel poll unconfigure	M	UC	X	0	0	1	0	1	0	1	X	X	X	X	X	X	X	X
REN	remote enable	U	UC	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1
RFD	ready for data	U	HS	X	X	X	X	X	X	X	X	0	X	X	X	X	X	X	X
ROS	request service	U	ST	X	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X
SCG	secondary command group	M	SE	X	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X
SDC	selected device clear	M	AC	X	0	0	0	0	1	0	0	X	X	X	X	X	X	X	X
SPD	serial poll disable	M	UC	X	0	0	1	1	0	0	1	X	X	X	X	X	X	X	X
SPE	serial poll enable	M	UC	X	0	0	1	1	0	0	0	X	X	X	X	X	X	X	X
SRQ	service request	U	ST	X	X	X	X	X	X	X	X	X	X	X	X	1	X	X	X
STB	status byte	M	ST	S8	X	S6	S5	S4	S3	S2	S1	X	X	X	X	X	X	X	X
TCT	take control	M	AC	X	0	0	0	1	0	0	1	X	X	X	X	X	X	X	X
TAG	talk address group	M	AD	X	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X
UCG	universal command group	M	UC	X	0	0	1	X	X	X	X	X	X	X	X	X	X	X	X
UNL	unlisten	M	AD	X	0	1	1	1	1	1	1	X	X	X	X	X	X	X	X
UNT	untalk	M	AD	X	1	0	1	1	1	1	1	X	X	X	X	X	X	X	X

DD : Dispositivo Dependiente
HS : Handshake
US : Comando Universal
SE : Comando Secundario
ST : Estatus
M : Mensaje de Multilínea
U : Mensaje de Unilínea

Se marca con una (X) a las líneas cuyo estado sea irrelevante y pueden ser ó no manejadas por alguna función en el bus.

1.6 Especificaciones Eléctricas y Mecánicas del bus IEEE488

1.6.1 Especificaciones Eléctricas

Las especificaciones para los manejadores de línea de señal y receptores están basadas en el uso de tecnología TTL. Y dispositivos de colector abierto ó de tres estados pueden ser usados en muchos casos.
Al utilizar la tecnología de tres estados se pueden alcanzar las mas altas tasas de transferencia de datos.

La relación entre los niveles lógicos y de voltaje se listan en la tabla 1.4.

Tabla 1.4
Relación entre nivel lógico y voltaje

Nivel Lógico	Nivel de Voltaje
0 (falso)	> +2.0 V (alto)
1 (cierto)	< +0.8 V (bajo)

Para enviar y recibir datos, se definen dos tipos de dispositivos:
- Manejador ó Driver
- Receptor ó Receiver

Los manejadores (drivers) pueden ser de colector abierto ó de tres estados.

Existen limitaciones al utilizar cualquiera de los dos tipos de drivers. Así tenemos que las lineas que pueden operar con tecnología de colector abierto son: SRQ, NRFD, NDAC y DIO1-8.

Las líneas de datos, en este caso, son usadas en los dispositivos con capacidad de sondeo en paralelo.

Entre las líneas que pueden operar con manejadores ya sea de colector abierto o de tres estados están: ATN, IFC, REN, EOI, DAV, DIO1-8.

Las líneas de datos, para tipo tres estados, solo pueden operar en dispositivos que no poseen capacidad de sondeo paralelo.

La mayor ventaja que ofrecen los manejadores de tres estados frente a los de colector abierto, es la mayor velocidad de transferencia de datos.

Las especificaciones de voltaje y corriente para los emisores de línea y receptores de línea, se listan en la tabla 1.4.

Tabla 1.4 Especificaciones de voltaje y corriente en emisores y receptores

Simbolo	Paramétrc	Min.	Max.	Tipo de Disposit.
V_{OH}	salida alta,0	+2.4V	+5.0V	Triestado
V_{OL}	salida baja,1	0 V	+0.5V	Triest. ó Col.Ab.
V_{IH}	entrada alta,0	+2.0V	+5.5V	Tipo Schmitt pref.
V_{IL}	entrada baja,1	-0.6V	+0.8V	Tipo Schmitt pref.
I_{OH}	salida alta,0		-5.2mA	
I_{OL}	salida baja,1	+48mA		
I_{IH}	entrada alta,0		+50µA	
I_{IL}	entrada baja,1		-1.6mA	

Las especificaciones de los emisores no requieren de lógica de colector abierto ni de tres estados, pero uno de esos dos tipos se dá por asumido.

Para añadir inmunidad al ruido, se sugiere que los receptores sean del tipo Schmitt, con un mínimo de hísteresis de 0.4 V.

$$V_{IL} = V_{tneg} \leq +0.8 \text{ V}$$

$$V_{IH} = V_{tpos} \geq +2.0 \text{ V}$$

Hísteresis: $V_{tpos} - V_{tneg} \geq +0.4$ V

Otro aspecto importante de estas especificaciones es el grafico de línea de carga en DC para colector abierto, como se muestra en la figura 1.5.

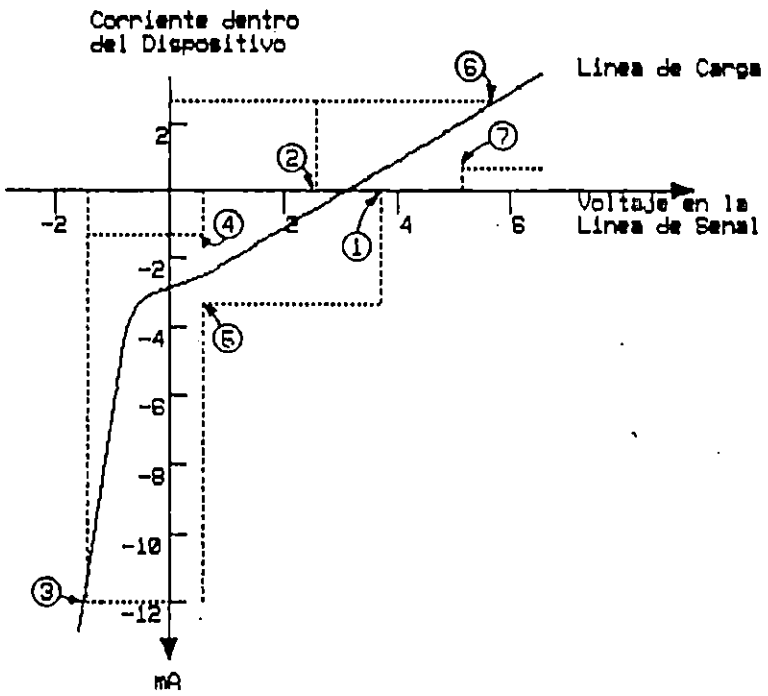


Figura 1.5 Línea de carga en DC para los manejadores de colector abierto de la GPIB.

De la gráfica se definen los valores de voltaje de línea de señal y de corriente dentro del dispositivo en los siguientes puntos:

1. Si I < 0 mA, V será < 3.7 V
2. Si I > 0 mA, V será > 2.5 V
3. Si I >-12.0 mA, V será >-1.5 V

   (solamente si el receptor existe)
4. Si V < 0.4 V, I será <-1.3 mA
5. Si V > 0.4 V, I será >-3.2 mA
6. Si V < 5.5 V, I será > 2.5 mA
7. Si V > 5.0 V, I será > 0.7 mA ó la impedancia de pequeña señal debe ser < 2 KΩ a una frecuencia de 1 MHz

En general la pendiente de la línea de carga en DC corresponde a una resistencia que no excede de 3 KΩ.

Las carácteristicas para pequeña señal AC correspondientes, están resumidas por el circuito de la figura 1.6.

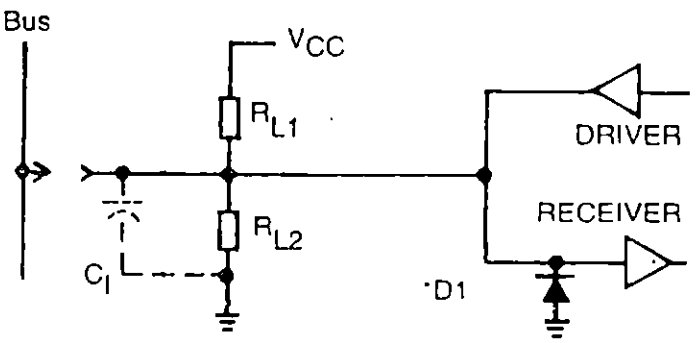


Figura 1.6 Circuito de entrada-salida tipico de una línea de señal de GPIB.

De donde:

Vcc = +5.0 V ± 5%

RL1 = 3.0 KΩ ± 5% (a Vcc)

RL2 = 6.2 KΩ ± 5% (a tierra)

Cl = Ccable + Ccomponentes ≤ 150 pF/metro a una frec. de 1 KHz y 0-5 V @

Dl = Diodo Clamping contenido tipicamente dentro de los componentes del receptor.

Driver ó Emisor:
cuya corriente de fuga para colector abierto es de +0.25 mA max. a un voltaje Vo = 5.25 V, para dispositivos de tres estados es de ±40 µA max. a un voltaje de Vo = +2.4 V.

Receiver ó Receptor:
cuya corriente de entrada es de -1.6 mA max. a Vo = +0.4 V y la corriente de fuga de entrada es de +40 µA max. a Vo = +2.4 V y +1.0 mA max. a Vo = 5.25 V

1.6.2 Especificaciones Mecánicas

Las especificaciones para el conector, montaje y cableado están definidas en el estandar. Los dispositivos están usualmente conectados con un cable ensamblado , consistente de 24 coductores con ambos tipos de conectores, plug y receptáculo, disponibles en cada extremo terminal (ver figura 1.7).

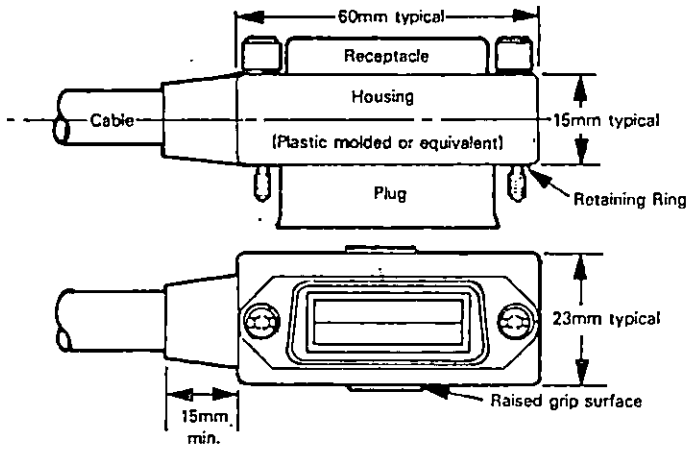
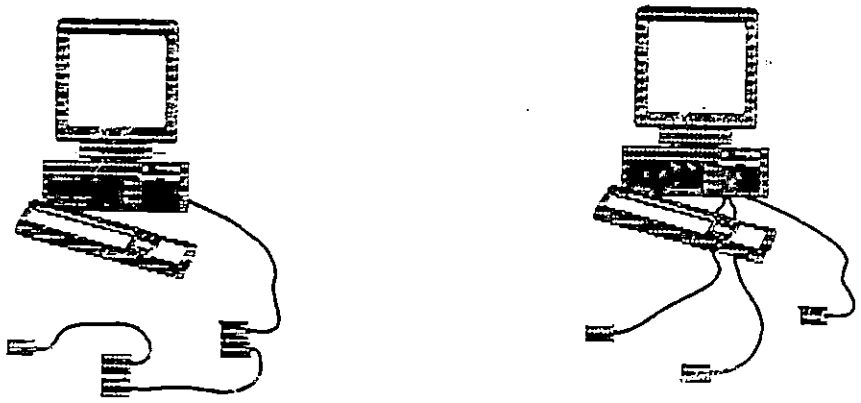


Figura 1.7 Conector del cable GPIB

Este diseño permite que los dispositivos se enlacen en cualquier configuración LINEAL ó ESTRELLA ó una combinación de ambas (ver figura 1.8).



(a) (b)

Figura 1.8 Configuraciones del cableado de la GPIB
(a) LINEAL (b) ESTRELLA

Para llevar a cabo el alto porcentaje de transferencia de datos, para lo cual, la GPIB fué diseñada, la distancia física entre los dispositivos y el número de dispositivos están limitados. Son típicas las siguientes restricciones:

a) Una separación máxima de 4 metros entre cualquier par de dispositivos y una separación promedio de 2 metros atraves del bus.
b) Una longitud total del cable de 20 metros.
c) No conectar mas de 15 dispositivos en cada bus, con no mas de dos tercios de los dispositivos encendidos.

Buses de extensores se pueden obtener de algunos fabricantes para usarlos cuando los límites anteriores deben ser excedidos.

El Conector: Este se define de acuerdo a los estandares.

Para los estandar IEEE 488 y ANSI MC1.1 el conector es del tipo ribbon de 24 pines, cuyos contactos son asignados como muestra la figura 1.9.

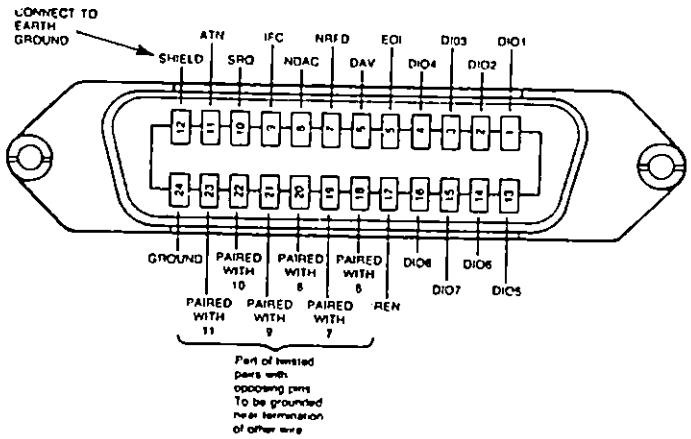
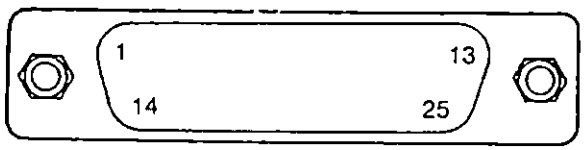


Figura 1.9 Conector para IEEE y ANSI

Cuyas especificaciones eléctricas y mecánicas son las siguientes:

Rango de voltaje: 200 Vdc
Promedio de corriente: 5 A
Duración: ≥ 1000 inserciones
Conectores sugeridos: Microribbon (Amphenol ó Cinch Serie 57) ó Champ (AMP)

Para el estandar IEC 625-1 el conector es uno tipo DB de 25 pines (MIL-C-24308) cuyos contactos son asignados como muestra la figura 1.10.



(HORIZONTAL)

Contact	Signal line	Contact	Signal line
1	DIO1	14	DIO 5
2	DIO 2	15	DIO 6
3	DIO 3	16	DIO 7
4	DIO 4	17	DIO 8
5	REN	18	Gnd (5)
6	EOI	19	Gnd (6)
7	DAV	20	Gnd (7)
8	NRFD	21	Gnd (8)
9	NDAC	22	Gnd (9)
10	IFC	23	Gnd (10)
11	SRQ	24	Gnd (11)
12	ATN	25	Gnd (12)
13	shield		

Figura 1.10 Conector para IEC 625-1

Y cuyas especificaciones eléctricas y mecánicas son:

Rango de Voltaje: 60 Vdc
Promedio de corriente: 5 A
Duración: > 1000 inserciones
Conector sugerido: Serie "D" de 25 pines

Por otra parte, el conector tipo D de 25 pines es usado por el Estandar de EIA (Electronic Industry Association) para el RS-232C, operando con voltajes superiores a los utilizados por la GPIB(arriba de ±25V y 0.5 A), por lo que deben manejarse con cuidado para no dañar dispositivos que trabajen con la GPIB.

CONCLUSIONES

- A pesar de que el bus IEEE-488 tiene muchos años de haber sido creado, y de que este tiene muchas aplicaciones en el area de la instrumentación electrónica y el control de muchos otros aparatos, el estudio de este bus no existe dentro de los programas de las asignaturas de la carrera de Ingenieria Eléctrica

- Debido a que no se ha hecho mucho enfasis en el estudio de este bus, la información corcerniente al estandar IEEE 488 es extremadamente poca, y para el diseño requerido en el presente trabajo fue necesario estudiar los documentos del estandar IEEE-488, en este capitulo se abordan los puntos mas generales abordados por el estandar, el que consta de dos libros, denominados IEEE-488.1 e IEEE-488.2.

- Hasta el momento este tipo de bus ha sido muy explotado comercialmente, particularmente en el area de programación, actualmente existen paquetes de software comerciales muy potentes, que le dan al usuario del bus funcioenes de alto nivel, permitiendo de esta manera comunicarse con los dispositivos mas rapidamente y obtener datos con mejor presentación. Por otro lado en el area del hardware para los circuitos de interfaz también se ha evolucionado mucho, se han creado microprocesadores que ejecutan las funciones del estandar o administran la operación de la interfaz, proporcionando mucha rapidez en las operaciones.

REFERENCIAS BIBLIOGRAFICAS

-IEEE Standard Digital Interface for Programmable Instrumentation. ANSI/IEEE Std 488.1-1987. Published by the Institute of Electrical and Electronics Engineers. June 16,1988.

-Tutorial Description of the Hewlett-Packard Interface Bus. Published by Hewlett-Packard Company. Noviembre 1984.

CAPITULO II

DISEÑO DEL CIRCUITO DE INTERFACE IEEE488

Introducción

En este cápitulo se aborda el diseño del circuito de interface, así como el ambiente de operación (dentro del computador), finalmente se describe brevemente la construcción de la tarjeta de interface y la conexión del cableado a utilizar con esta.

El diseño de la interface debe cumplir con las especificaciones del estandar IEEE 488 y con todos los criterios de construcción necesarios para que opere dentro de una computadora PC 501 AT LEMMON.

La propuesta está basada en el chip 8255A PPI (Interfase Programable de Periféricos).

Una interface elaborada con un circuito como este, hace necesaria una arquitectura de funcionamiento basada mas en el software.

Donde todas las funciones implementadas deberán hacerse a traves de programación.

Como se mencionó anteriormente, el bus IEEE 488 está ampliamente generalizado, teniendo hasta este momento una enorme cantidad de aplicaciones en la industria y la investigación científica.

Existen por lo tanto, un gran número de tarjetas ó circuitos de interface que van desde lo mas simple hasta lo mas complejo.

Esto se logra a traves de circuitos integrados diseñados para efectuar las funciones especificas del bus, liberando a los usuarios del complicado trabajo de dominar las funciones y códigos del bus, creandose así, funciones de alto y bajo nivel; siendo las de bajo nivel, las que requieren que el usuario posea un conocimiento considerable sobre el manejo de la GPIB.

2.1 Interconexión de la Interface con la computadora.

Las computadoras PC ó compatibles poseen canales o ranuras de expansión ó también llamadas Slots, los que están dedicados particularmente para establecer la interconexión con una interface cuando es necesario.

En la figura 2.1 se muestra por medio de buses las partes más importantes en la interconexión con la interface.

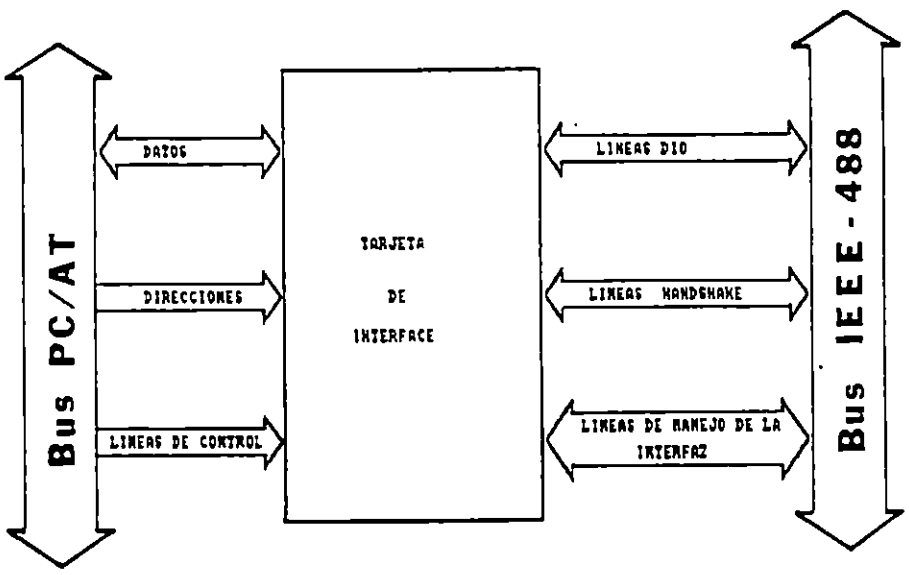


Figura 2.1 Diagrama de bloques de la interconexión de la computadora y la interfase.

2.2 Descripción de los Slots o Canales I/O de Expansión de las computadoras IBM PC ó Compatibles.

La descripción fisica de los canales I/O de expansión se muestran en la figura 2.2.

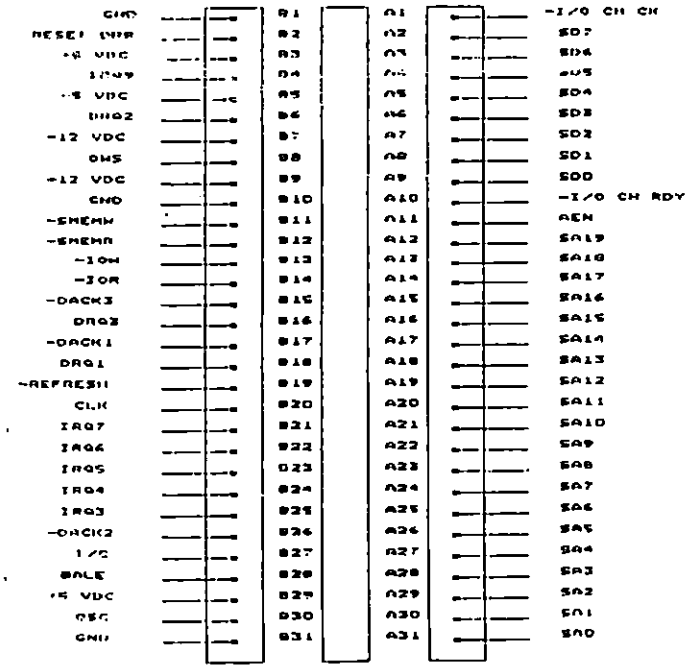


Figura 2.2 Descripción del Pin-Out de los canales de expansión

El funcionamiento de cada uno de los pines se describe a continuación:

SD0-SD7 (E/S)
Son los 8 bits menos significativos del bus de datos del sistema. En transferencia de 8 bits son los unicos utilizados.

SA0-SA19 (E/S)
Representa el bus de direcciones del computador y son usados para direccionar memoria y dispositivos de Entrada/Salida dentro del sistema. Son 20 líneas de direccionamiento que permiten accesar hasta un megabyte de memoria.

CLK (S)
Es la señal de reloj de 8 ó 10 MHz del sistema. Debe usarse solo para efectos de sincronización.

RESET DRV (S)
Inicializa la lógica del sistema al encender la máquina o durante un bajo voltaje.

BALE (S)
Abreviatura de "Buffered Address Latch Enable". Esta señal proviene del controlador del bus y es utilizada para enclavar direcciones válidas del microprocesador. Se utiliza como indicador de direcciones válidas del DMA ó del CPU.

I/O CHCK (E)
Esta señal proporciona al sistema, información de error de paridad en memoria ó de dispositivos en los canales de entrada/salida. Es activa en bajo.

IRQ3-IRQ7 (E)
Son señales de solicitud de interrupción, IRQ7 es la de mayor prioridad e IRQ3 la de mayor.

IOR (E/S)
Esta señal le indica a un periférico que maneje sus datos a traves del bus de datos. La señal es controlada por el microprocesador ó el contolador del DMA O similares, presentes en el canal entrada/salida. Es activa en bajo.

IOW (E/S)
Esta señal le indica a un periférico que lea los datos presentes en el bus de datos. Es controlada en igual forma que IOR. Es activa en bajo.

SMEMR (S)
Esta señal indica a dispositivos de memoria, manejar datos a traves del bus de datos. Es controlada por el CPU ó el controlador del DMA. Es activa solo para rangos menores de 1 Megabyte de memoria.

SMEMW (S)

Esta señal indica a los dispositivos de memoria almacenar los datos presentes en el bus de datos. Controlada por el CPU ó por el controlador del DMA. Es activa solo para rangos menores de un megabyte.

DRQ1, DRQ2, DRQ3 (E)
Son señales de solicitud de transferencia por canales DMA. DRQ1 es la de mayor prioridad y DRQ3 la de menor prioridad. Se mantiene en alto hasta que el correspondiente DACK se activa.

DACK1, DACK2, DACK3 (S)
Se utiliza para accesar solicitudes de DMA. Son activas en bajo.

AEN (S)
Cuando esta señal se activa (alto), el CPU cede el control sobre el bus de direcciones y bus de datos al controlador del DMA, permitiendo la transferencia de DMA.

REFRESH (E/S)
Esta señal indica un ciclo de refrescamiento. Es activa en bajo y puede controlar el microprocesador en el canal de entrada/salida.

T/C (S)
Proporciona un pulso alto cuando es alcanzada la cuenta final por cualquier canal DMA.

OSC (S)
Señal de reloj de 14.31818 MHz, no sincronizada.

2.3 Conexión con la Interface

En general, para interconectar la interface con la computadora, utilizan las siguientes ranuras:

El bus de datos entrada/salida de 8 bits (de A2 a A9) con el bit menos significativo en A2 (SD7) y el mas significativo en A9 (SD0).
La dirección del flujo de datos es controlada por los pines B13 y B14 que corresponden a IOW e IOR respectivamente.

Del bus de direcciones se utilizarán los pines SA0 hasta SA9, esto se determina por el rango que le corresponde a los puertos de entrada/salida (I/O) dentro del mapa de direcciones de la computadora.

Otros pines que también se utilizarán son: la línea IRQ4 de interrupción y para la fuente de alimentación +5 VDC (B3 y B29) y GND (B1 y B31).

Aunque los pines utilizados varían dependiendo del diseño,

en general, los pines descritos anteriormente, serán los más usados.

2.4 Propuesta para el Circuito de Interface

El circuito propuesto está basado en el chip 8255A (PPI), que es una interface paralela de periféricos. Permitiendo la comunicación en paralelo con tres puertos (A, B y C), los cuales se pueden direccionar a través de la lógica de control del chip.

2.4.1 Descripción General del 8255A (Modo 0)

El 8255A es un periférico programable de entrada/salida paralelo para propositos generales y para el presente diseño viene a ser la parte central del circuito de interface.

El diagrama de bloques se muestra en la figura 2.3. Es un circuito de soporte para microprocesadores de la familia Intel, y proporciona 24 pines programables de entrada /salida, divididos en tres puertos (A, B y C).

La configuración de cada puerto se hace por medio de software.

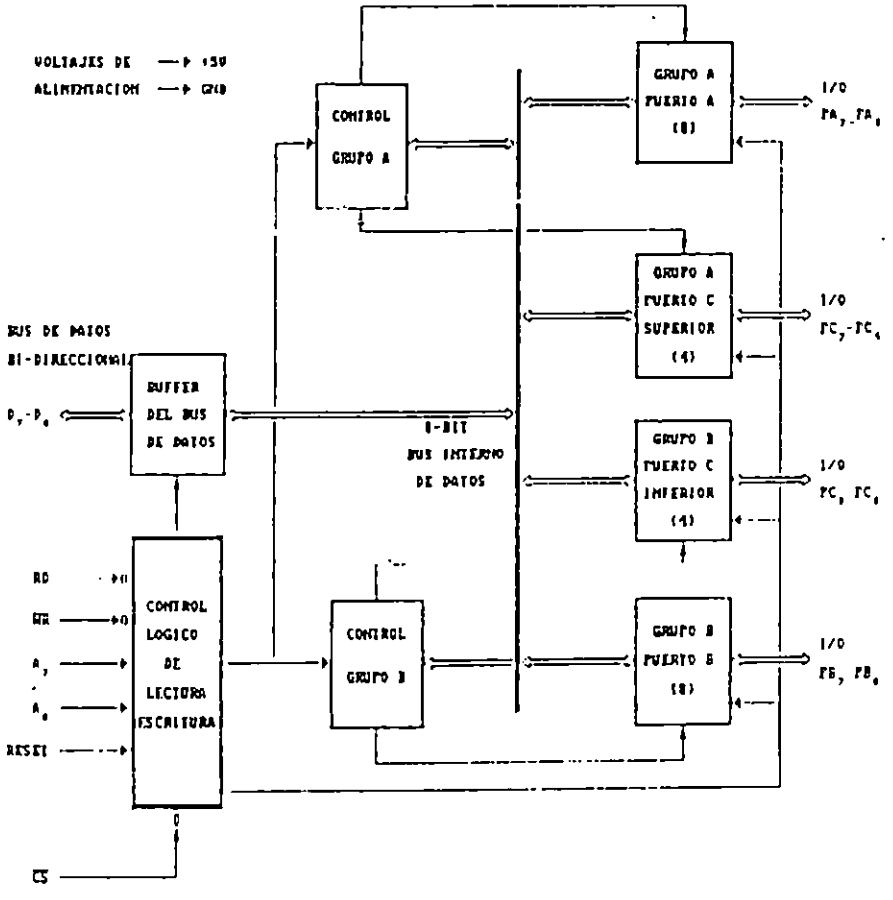


Figura 2.3 Diagrama de bloques del 8255A

El 8255A posee 3 modos de operación: modo 0, modo 1 y modo 2. Como su funcionamiento es muy conocido y para esta aplicación en particular sólo se operará en modo 0 de entrada/salida elemental, a continuación se describe unicámente este modo de operación.

Al encender la máquina el 8255A, se inicializa y sus tres puertos quedan configurados como puertos de entrada. Para configurar los puertos se utiliza la palabra de control, que consta de ocho bits, y que son colocados dentro del registro de control, accesado previamente.

La interface solo responde a 4 direcciones, correspondientes a los puertos A, B y C y el registro de control de solo escritura. El formato del byte de control se muestra en la figura 2.4.

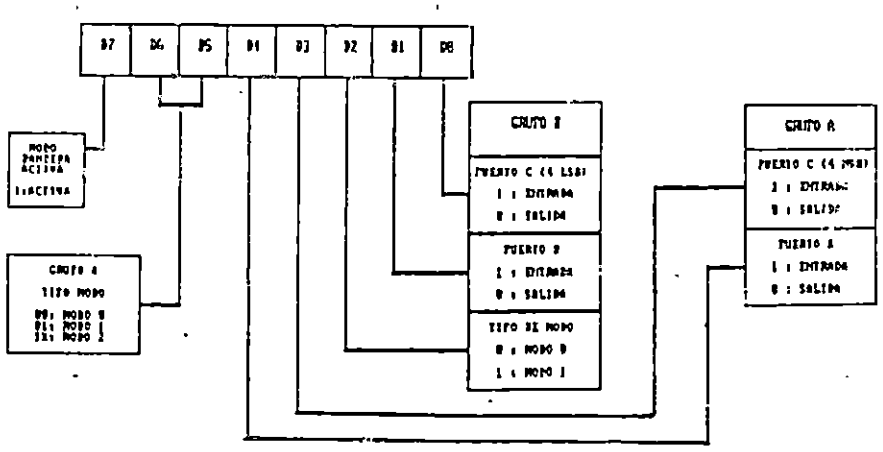


Figura 2.4 Byte de Control

Operación en Modo 0: Entrada/Salida Elemental

Este modo configura a los tres puertos solamente como entrada/salida, presentando las siguientes carácteristicas:

- Los puertos A y B de 8 bits cada uno y el puerto C dividido en dos grupos de 4 bits cada uno de los cuales pueden ser entrada ó salida.

- Cualquier puerto puede ser entrada o salida.

- Las salidas son enclavadas.

- Las entradas no son enclavadas.

- Existen 16 formas de configurar este modo.

En la tabla 2.1 se muestran las 16 posibilidades de configuración en el modo 0.

Tabla 2.1 Configuración de Puertos en el Modo 0

Byte de Control	PUERTO A	4MSB PUERTO C	PUERTO B	4LSB PUERTO C
80	salida	salida	salida	salida
81	salida	salida	salida	entrada
82	salida	salida	entrada	salida
83	salida	salida	entrada	entrada
88	salida	entrada	salida	salida
89	salida	entrada	salida	entrada
8A	salida	entrada	entrada	salida
8B	salida	entrada	entrada	entrada
90	entrada	salida	salida	salida
91	entrada	salida	salida	entrada
92	entrada	salida	entrada	salida
93	entrada	salida	entrada	entrada
98	entrada	entrada	salida	salida
99	entrada	entrada	salida	entrada
9A	entrada	entrada	entrada	salida
9B	entrada	entrada	entrada	entrada

2.4.2 Lógica de Acceso al 8255A

La dirección del flujo de datos en la PPI, es controlada por los pines RD y WR, los que se conectan a los pines IOR e IOW de las ranuras de expansión respectivamente.

En la tabla 2.2 se muestra el direccionamiento de los puertos y el registro de control.

Tabla 2.2
Dirección de los 3 puertos y el registro de control

	SA9	SA8	SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0	Hex
PUERTO A	1	1	1	1	1	0	1	1	0	0	3E8
PUERTO B	1	1	1	1	1	0	1	1	0	1	3E9
PUERTO C	1	1	1	1	1	0	1	1	1	0	3EA
Palabra de Control	1	1	1	1	1	0	1	1	1	1	3EB

La PPI posee, además, dos pines de dirección para los puertos (A0 y A1) y el pin CS para habilitar el acceso al Chip. Para estos pines se hace necesario un decodificador de direcciones que habilite el chip con las direcciones correspondientes dentro del rango I/O del mapa de memoria.

Tales direcciones serán:

- 3E8 Hex : Puerto A
- 3E9 Hex : Puerto B
- 3EA Hex : Puerto C
- 3EB Hex : Dirección del byte de control

El decodificador de direcciones para esta interface se muestra en la figura 2.5.

Se utiliza la línea AEN (All) para evitar que la interface sea habilitada durante la transferencia DMA. En general através de las compuertas se habilita el pin CS (Chip Select) para las cuatro direcciones anteriores.

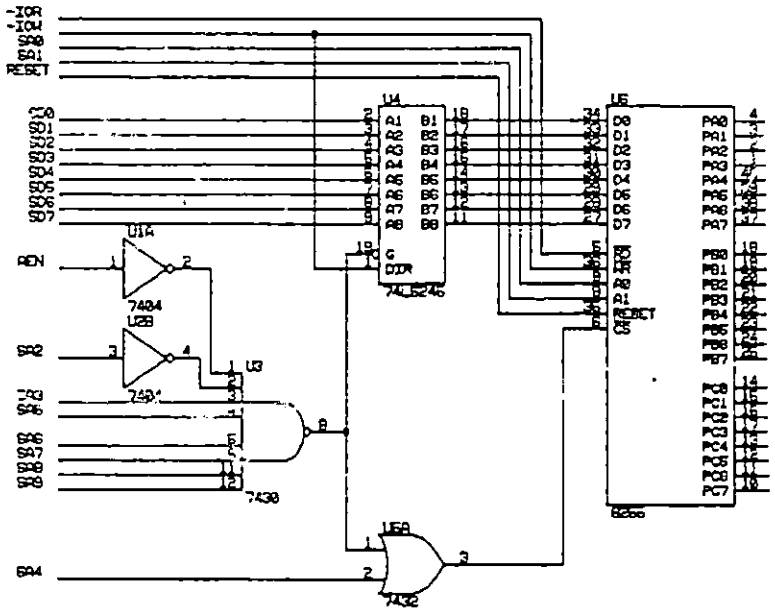


Figura 2.5 Lógica de acceso al 8255A

2.4.3 Circuito de la Interface para el bus IEEE 488

El esquema del circuito para la interface se muestra en la figura 2.5,en el se muestra el circuito completo, incluyendo el decodificador de direcciones.

El circuito cumple exclusivamente la misión de adaptación de los niveles propios del bus IEEE488 con los de la PPI.

Para lo cual se hace necesario, diferenciar las líneas según su aplicación:

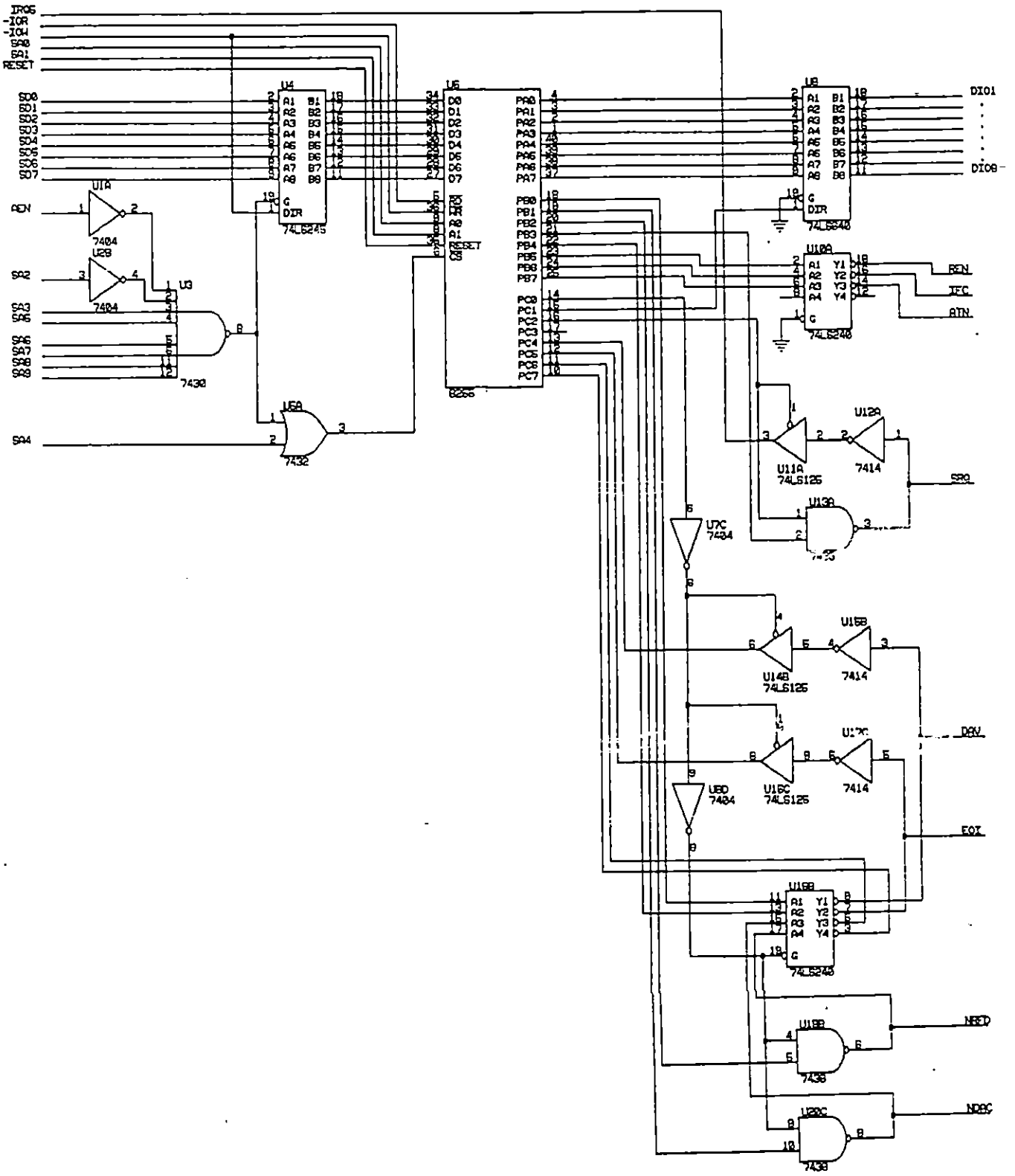


Figura 2.6 Circuito de Interface del bus IEEE 488 con el 8255A

a. Las líneas de datos DIO1-DIO8, deben ser bidireccionales:
   - De salida cuando actúa como controlador o como transmisor.
   - De entrada cuando trabaja como receptor o cuando realizan un sondeo paralelo.

b. Las líneas de intercambio de señales (handshake): DAV, NDAC, NRFD y EOI; deben ser bidireccionales , pero sus direcciones no son concurrentes, si no que actúan según la tabla 2.3.

Tabla 2.3 Dirección de las Líneas Handshake

	DAV	EOI	NRFD	NDAC
Controlador-Emisor	salida	salida	entrada	entrada
Controlad.-Receptor	entrada	entrada	salida	salida

c. Las líneas de control: ATN, IFC, REN y SRQ, son propias del funcionamiento como controlador, de manera que siempre son unidireccionales. Las direcciones respectivas son:
   - ATN : Salida
   - IFC : Salida
   - REN : Salida
   - SRQ : Entrada

   La línea de petición de servicio (SRQ) se toma siempre como entrada, porque se parte del hecho de que el circuito funciona solo como controlador único dentro del sistema.

Para las líneas de datos (DIO1-8), se utilizará el puerto A del 8255A, por lo que para la adaptación bidireccional, de acuerdo a las normas del estándar IEEE488, se utiliza el IC 74LS640, que es un tranceiver (tranceptor) TTL, con inversión, aunque opera con 24mA (ICL) es posible utilizarlo para este proposito, la dirección de los datos es manejada a traves de la pin PC1 de la PPI.

Para las líneas de intercambio de señal (handshake), la situación es algo diferente. De la tabla 2.3, se observa que las direcciones de DAV y EOI son contrarias a las NRFD y NDAC. Además las líneas DAV y EOI trabajan con TTL de tres estados (ver sección 1.6.1 del capitulo I) y las líneas NDAC y NRFD operan con TTL de colector abierto.

Para lograr adaptar estas líneas se utiliza el circuito de la figura 2.7.

El driver 74LS240 trabaja como entrada para NRFD y NDAC y como salida para DAV y EOI.

Las salidas para NRFD y NDAC son manejadas por las compuertas NAND 7438 de colector abierto.

Las entradas para DAV y EOI son manejadas por los inversores tipo Schmitt 7414, que cuentan con un voltaje de histeresis de 0.4 V. El buffer 74LS125 en serie con el inversor 7414 habilita las líneas como entradas.

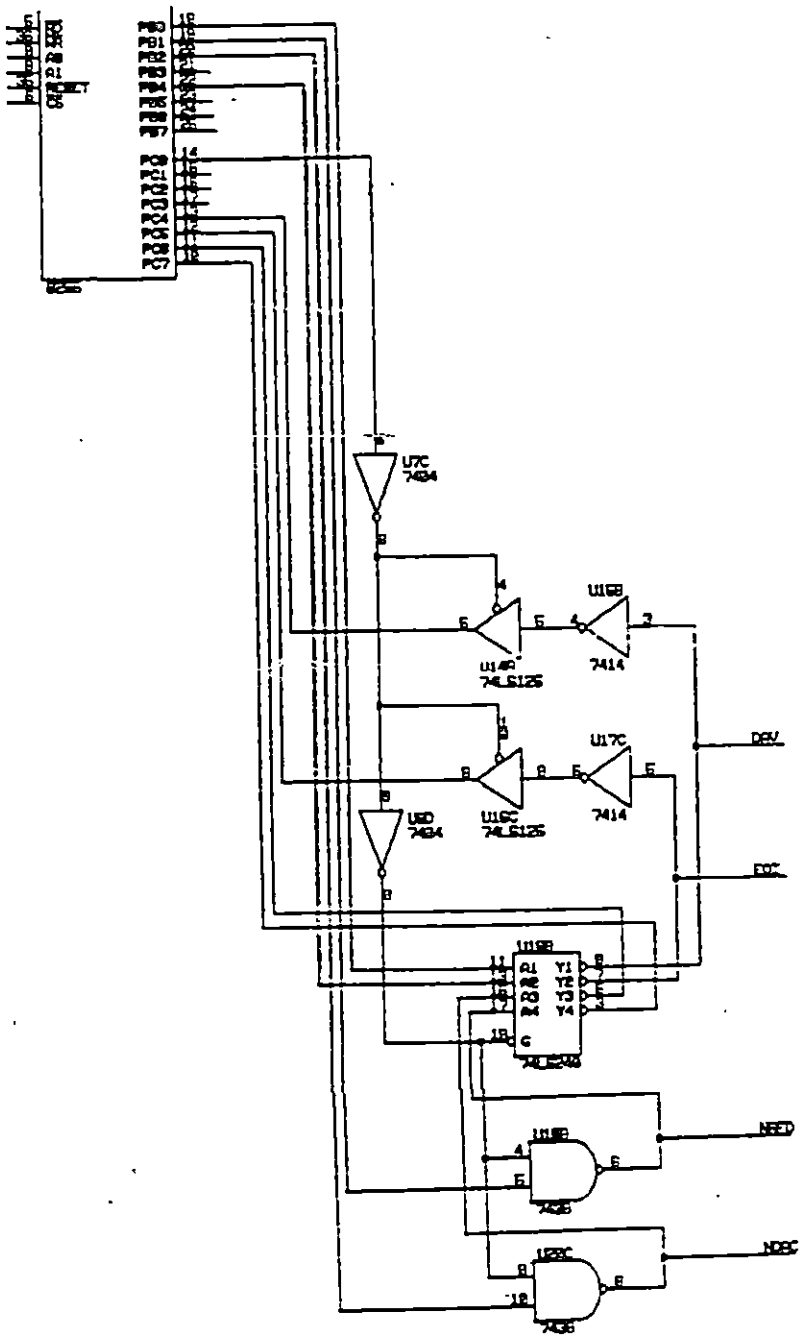


Figura 2.7 Adaptación de las Líneas Handshake

Como dichas líneas son bidireccionales y están agrupadas en pares, se recurre a la disposición anterior, utilizando el puerto B como salida y el puerto C como entrada. De esta manera a través de programación se controla la dirección de dichas líneas.

Las líneas de control usadas en los circuitos de adaptación de las líneas de datos DIO, así como en las líneas de intercambio de señal, se obtienen del puerto C (PC0 y PC1), conectados a inversores (7404).

El establecimiento del flujo de datos se realiza con las líneas PC0, PC1 y PC2 de acuerdo con la tabla 2.4.

Tabla 2.4 Control del flujo de datos

PC0	PC1	PC2	DIO 1-8	DAV-EOI	NDAC-NRFD	SRQ
0	x	x	-	salida	entrada	-
1	x	x	-	entrada	salida	-
x	0	x	entrada	-	-	-
x	1	x	salida	-	-	-
x	x	0	-	-	-	entrada
x	x	1	-	-	-	salida

Finalmente , para las líneas de control del bus, y debido a que estas son unidireccionales, basta con colocar un driver unidireccional 74LS240 de tres estados, enclavado fijamente como salida

. Dado que estas actúan siempre como salida (exceptuando la línea SRQ), se conectan al puerto B, de tal manera que este puerto operará siempre en dicha dirección.

La línea SRQ, no influye en el funcionamiento del circuito, actuando directamente sobre las líneas de interrupción de la computadora, debido a que una petición de servicio, desencadenará una respuesta definida por el programa y que actúa sobre el computador.

La asiganción de las líneas del bus a los puertos del 8255A se resume en la tabla 2.5.

Tabla 2.5 Asignación de las líneas de la GPIB al 8255A

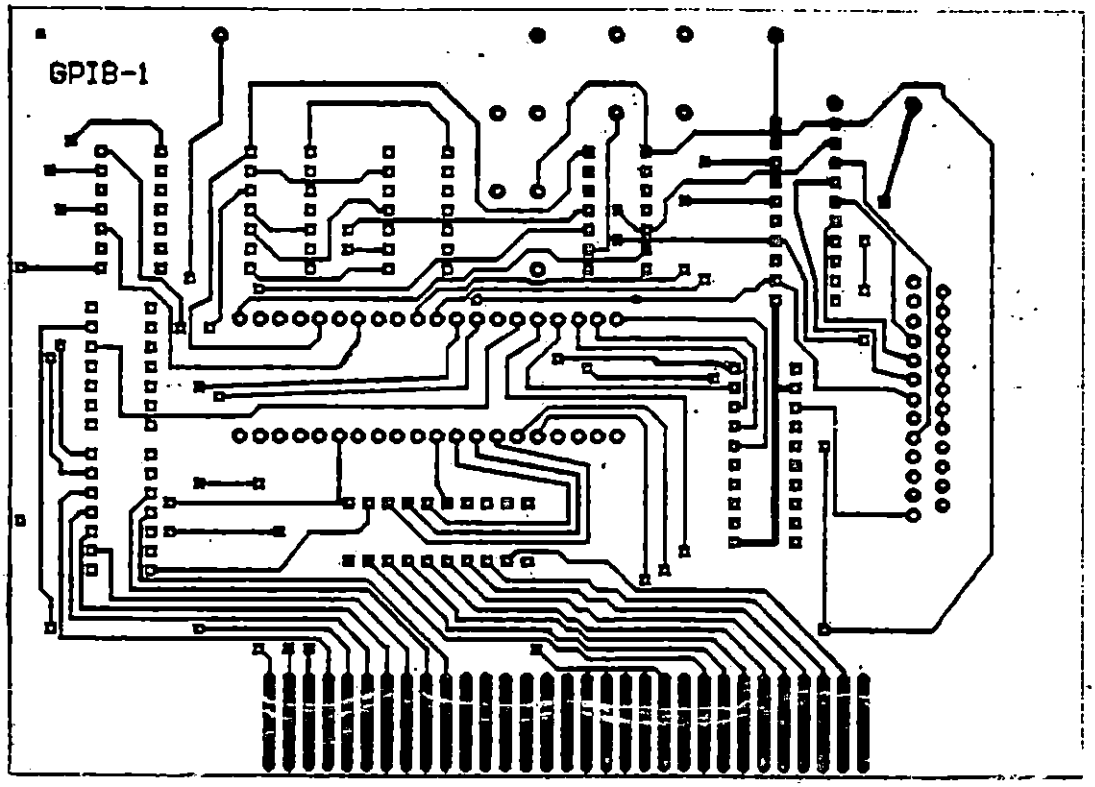
PUERTO A	PA0	DIO 1	BIDIRECCIONAL
	PA1	DIO 2	"
	PA2	DIO 3	"
	PA3	DIO 4	"
	PA4	DIO 5	"
	PA5	DIO 6	"
	PA6	DIO 7	"
	PA7	DIO 8	"
PUERTO B	PB0	DAV	SALIDA
	PB1	NRFD	"
	PB2	NDAC	"
	PB3	SRQ	"
	PB4	EOI	"
	PB5	REN	"
	PB6	IFC	"
	PB7	ATN	"
PUERTO C	PC0 Control		SALIDA (Interna)
	PC1 Control		" "
	PC2 Control		" "
	PC3 Sin usar		
	PC4	DAV	ENTRADA
	PC5	EOI	"
	PC6	NDAC	"
	PC7	NRFD	"

2.4.4 Construcción de la Tarjeta de Interface IEEE488 o GPIB

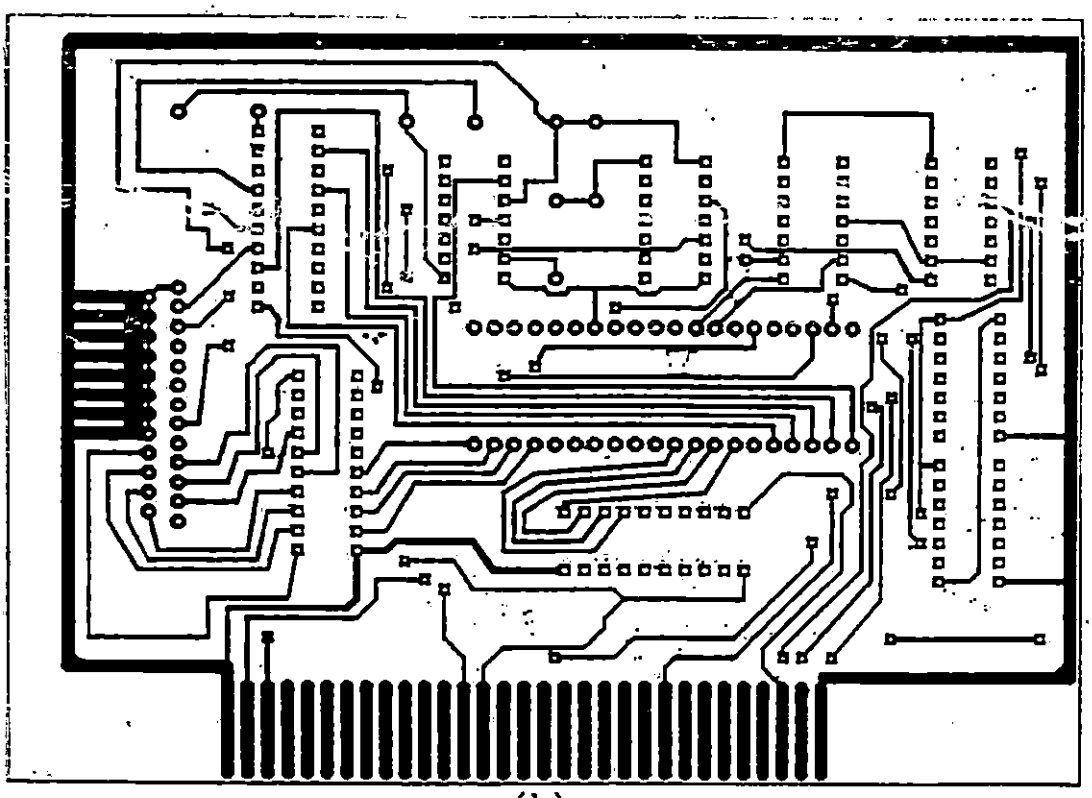
El impreso para la interface fue elaborado con el programa EAGLE, que es un programa CAD (Diseño Auxiliado por Computadora) para ingeniería. Este programa es capaz de producir el esquema de la tableta impresa a partir del diagrama esquemático. El impreso se construyó de esta manera, pero debido a la alta densidad de pistas fue necesario hacer muchas modificaciones antes de imprimirlo sobre la tableta, por medio del plotter.

Para elaborar las conexiones que van al slot de la computadora fue necesario crear un macro y de la misma forma ocurrió con la ubicación de los pads de conexión al bus.

El diagrama del impreso se muestra en la figura 2.8, en ella se muestran ambos lados: de componentes y de soldadura.



(a)



(b)

Figura 2.8 Esquema de la tarjeta impresa (a) Lado de componentes (b) Lado de soldadura

También con este programa se obtiene el diagrama de componentes sobre la tableta, este se muestra en la figura 2.9, en ella se pueden observar la ubicación de todos los componentes sobre la tarjelta, además del nombre que recibe: GPIB-1, con el que se identificará la tarjeta de interfaz en cuestión.

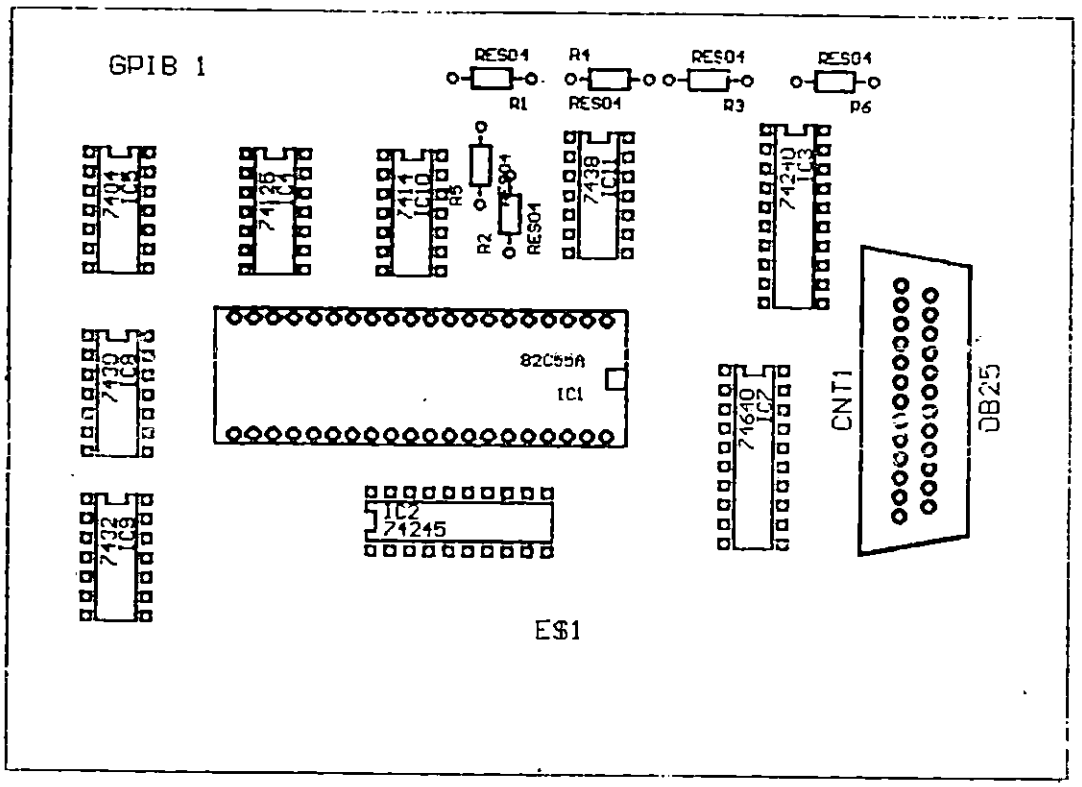


Figura 2.9 Esquema de los componentes de la Interface sobre la tarjeta impresa.

Las dimensiones de la tarjeta son de 10 x 14 cm , estas medidas se tomaron en base a las dimensiones de otras tarjetas. Puede, en todo caso, variarse estas medidas haciendo un poco más grande la tarjeta, pero esto dependerá del diseñador, solo debe tomarse en cuenta que la tarjeta debe caber dentro de la computadora, las medidas físicas máximas permisibles para la tarjeta son de 11.5 X 24 cm.

En cuanto al conector, se escogió el tipo DB25 (de 25 pines), utilizado por el estandar IEC-625. La asignación de líneas corresponde a las de la figura 1.10, el conector de la interfaz está soldado a 90° del plano de la tarjeta.

Este tipo de conector se escogió porque es más fácil de

adquirir, que el conector tipo ribbon del estandar IEEE488 (de 24 pines).

Debido a que es imposible adquirir conectores como los mostrados en la figura 1.7, es necesario acoplar los cables que dan a los dispositivos, uniendo las líneas de un cable con las líneas correspondientes de otro cable en los terminales de cada conector, como se muestra en la siguiente figura.

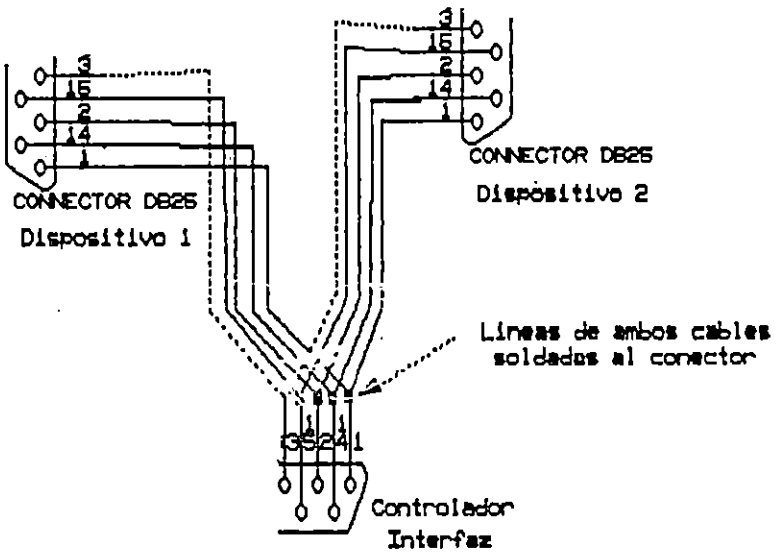


Figura 2.10 Conexión recomendada de los cables a la interfaz

Por el consumo de potencia, de la interfaz al conectarse con cierto número de dispositivos, es preferible no instalar otras tarjetas dentro de la computadora, porque puede afectar la fuente potencia del computador.

Además la cantidad máxima de dispositivos que se pueden conectar es de 10, y preferiblemente apagando al menos 2 de los dispositivos instalados al bus. Pero esto debe tenerse en cuenta al operar arriba de los 700kbytes/segundo. A esta velocidad de comunicación el estandar recomienda encender todos los dispositivos conectados al bus.

CONCLUSIONES

- Para construir el circuito de la interface, debieron cumplirse los requerimientos señalados por el estandar. Y como el objetivo del trabajo era diseñar una interface sencilla, se recurrió al IC 8255A que proporciona buenos resultados, al direccionar los datos por tres puertos diferentes. Pero debe tenerse cuidado pues este consume más potencia que su contraparte CMOS el 82C55A, siendo este último la mejor opción.

- El circuito puede ser mejorado particularmente en la sección de manejo del Puerto B, separando las salidas a las líneas por medio de multiplexores y flip-flops tipo D o bien por medio de un Latch, pero esto ocasiona que el consumo de potencia aumente, así como el tamaño de la tarjeta y la densidad de pistas de la misma.

- La tarjeta construida presenta ciertas ventajas sobre algunas otras, particularmente porque utiliza la velocidad de operación del microprocesador y porque accesa directamente todos los datos, sin embargo esto presenta la desventaja que en procesos largos de comunicación con el bus, el tiempo que habrá que esperar para que el computador retorne el control al usuario puede ser muy extenso; por otro lado esto solo puede ocurrir si se crea un software de alto nivel que ejecute miles de operaciones automáticamente con poca participación del usuario, de cualquier modo el circuito fué diseñado de tal manera que se pueda alcanzar la máxima velocidad permisible de comunicación de 1 Megabyte por segundo.

REFERENCIAS BIBLIOGRAFICAS

-IEEE Standard Digital Interface for Programmable Instrumentation. ANSI/IEEE Std 488.1-1987. Published by the Institute of Electrical and Electronics Engineers. June 16,1988.

-Lopez F.J., Gutierrez P.M. y Castells F.J., Interface microprocesador bus IEEE488. Serie: Mundo Electronico. Microprocesadores y microcomputadoras. Publicaciones Marcombo,S.A.,Mexico-Barcelona.1984.

-Suria Montes, José Heriberto y Meléndez Valle, Werner David. "Diseño y construcción de interfase y módulo de prueba para componentes integrados digitales en la computadora PC-501-AT Lemmon." Tesis para optar al grado de Ingeniero Electricista.Biblioteca de la Escuela de Ingenieria Electrica. Universidad de El Salvador. Abril 1992.

CAPITULO III

REQUERIMIENTOS DE PROGRAMACION DEL ESTANDAR IEEE488

Introducción.

Para elaborar el programa de manejo de la interfaz es necesario conocer como se realizan los protocolos de las funciones, así como el set de intrucciones o comandos a los que una interfaz debe interpretar.

En este capitulo se aborda con cierto detalle los protocolos de las funciones, ademas de otros topicos relacionados con la operción de cada función y el acceso a las interfaces dentro de los dispositivos.

El estandar aborda el area del software para este bus desde dos puntos de vista uno desde las funciones de interfaz y otro desde las funciones programadas por el usuario, es por esto que la estandarización ha alcanzado dos areas el hardware y el software.

Son dos documentos para el mismo bus, pero en uno solo se abordan las funciones relacionadas mas directamente con el hardware , en el segundo se tocan aspectos relacionados con la programación del sistema, protocolos de alto nivel que deben ser comunes a todos los equipos que operan con este tipo de bus, aunque existen comandos e incluso protocolos dentro de este estandar que no son comunes entre ciertos dispositivos .

Sin embargo la información que se encuentra en este documento es mucho mas avanzada que la del primer estandar.

En este capitulo se abordan de manera general los protocolos de las funciones, los que serviran de base teórica para el diseño del programa de manejo de la interfaz, estos intorducen tiempos en el desarrollo de la comunicación, sino que también codigos y señales que determinan las respuestas correctas de los dispositivos.

3.1 Funciones de Interfaz

Una función de interfaz es el elemento del sistema que provee la facilidad operacional básica por medio de la cual un dispositivo puede recibir, procesar o enviar mensajes. Un número de funciones de interfaz, cada una de las cuales actúa en concordancia con el protocolo específico, están definidos en el estandar IEEE488. Cada función de interfaz específica puede solamente enviar o recibir un limitado set de mensajes dentro de clases particulares de mensajes.

Cada una de las funciones de interfaz está definida en términos de uno o más grupos de estados interconectados, mutualmente exclusivos.

Uno y solamente un estado estará activo en cualquier tiempo dentro de un grupo de estados interconectados.

Las definiciones para cada estado de una función de interfaz están dadas de la manera siguiente.

a. Mensajes que pueden o deben ser enviados sobre la interfaz mientras que ese estado esté activo.

b. Condiciones dentro de las cuales, la función debe dejar ese estado y entrar a uno de los otros estados en su grupo.

Estos mensajes y condiciones definen la capacidad de procesamiento del estado.

Cada estado, que una función de interfaz puede asumir, está representado gráficamente como un círculo. Un mnemónico de cuatro letras mayúsculas siempre terminadas en una s, es usado dentro del círculo para identificar su estado (ver figura 3.1)

Figura 3.1 Representacion de un estado para una función.

Todas las transiciones permisibles entre estados de una función de interfaz son representadas gráficamente por flechas entre ellos.

Cada transición es calificada por una expresión cuyo valor puede ser cierto o falso (ver figura 3.2). La función de interfaz permanecerá en su estado actual, si todas las expresiones que califican las transiciones a otros estados son falsas.

La función de interfaz entrará en el estado apuntado, si y solo si, una de esas expresiones es verdadera.

El nuevo estado puede ser alcanzado en cualquier tiempo despues que la expresión es cierta, pero al menos un valor de tiempo es especificado.

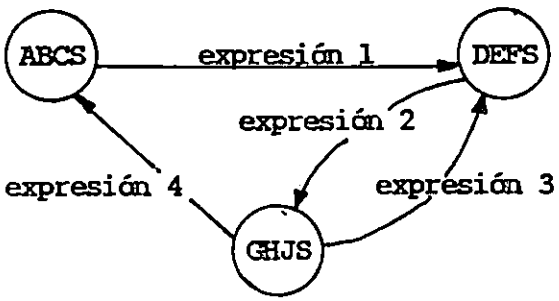


Figura 3.2 Representación de tres estados y sus transiciones.

Una expresión consiste de uno o mas: mensajes locales, mensajes remotos, enlaces de estados, o tiempos límite mínimos o máximos en conjunción con los operadores AND, OR, o NOT.

Un mensaje local a una función de interfaz es representada por un mnemónico de tres letras en minúsculas, como por ejemplo: rdy.

Un mensaje remoto (recibido vía la interfaz) es representado por un mnemónico de tres letras escritas en mayúsculas, como por ejemplo: ATN.

Un enlace desde cualquier otro diagrama de estado es representado por un mnemónico de cuatro letras encerradas en un óvalo; por ejemplo: LACS.

Un enlace de estado es cierto si el estado encerrado está actualmente activo, de otra manera éste es falso.

Un tiempo límite mínimo es representado por Tn. Este símbolo activa un valor cierto solamente después que la interfaz ha permanecido en el estado original la correspondiente transición para el valor de tiempo especificado. Este permanecerá cierto hasta que el estado esté fuera

. Si una transición es calificada por un tiempo límite máximo (dentro de tn), entonces el estado apuntado será alcanzado dentro del rango especificado. El valor para estos tiempos límites están definidos en la tabla 3.1.

El operador AND está representado por el símbolo $\wedge$.

El operador OR es representado por el símbolo $\vee$.

El operador AND toma precedencia sobre el operador OR dentro de una expresión a menos que sea especificado por paréntesis.

Tabla 3.1
Valor de tiempos límite para las funciones de Interfaz

Identificador * Valor de Tiempo	Función (Aplicada a)	Descripción	Valor
T1	SH	tiempo colocado para	≥ 2µs (1)
t2	SH,AH,T,L,LE,TE	respuesta a ATN	≤ 200ns
T3	AH	tiempo de aceptación de mensajes de interfaz (2)	> 0 (3)
t4	T,TE,L,LE,C,RL	respuesta a IFC o REN falso	<100µs
t5	PP	respuesta a ATN EOI	≤ 200ns
T6	C	tiempo de ejecución de sondeo paralelo	≥ 2µs
T7	C	retardo de controlador para permitir al actual talker ver un mensaje ATN	≥ 500ns
T8	C	longitud de IFC o o REN falsos	> 100µs
T9	C	retardo para EOI(4)	≥ 1.5µs (5)
T10	C	retardo para DAV	≥1.5µs

Notas:
*Valores de tiempos especificados por una t (minúscula)indican el tiempo máximo permitido para hacer una transición de estado.
Valores de tiempos especificados por T (mayúscula)indican el tiempo mínimo en que una función permanecerá en un estado antes de salir de èste.
(1)Si drivers de tres-estados son usados en las líneas DIO,
DAV y EOI
T1 puede ser:
a) >1100ns
b) o >700ns si se conoce que dentro del controlador ATN es manejado por un driver de tres-estados, sin embargo este valor no es recomendado
c) 0 >500ns para todos los subsecuentes bytes que siguen al primero enviado despues de cada transición falsa de ATN (el primer byte será enviado en concordancia con a) y b)
d) o >350ns para todos los subsecuentes bytes que siguen al primero enviado despues de cada transición falsa de ATN dentro de condiciones especificadas dentro del estandar IEEE 488.1 para la transmisión en alta velocidad.
(2)Tiempo requerido para las funciones de interfaz para aceptar, y no necesariamente responder a mensajes de interfaz.
(3) Depende de la implementación
(4)Retardo requerido para las líneas de señal EOI, NDAC y NRED para indicar estados válidos.
(5)>600ns para drivers de tres-estados

El operador NOT es representado por una barra horizontal colocada sobre la porción de la expresión para ser negada. La expresión negada resultante tiene un valor cierto si y solamentel si el valor de la expresión dentro de la barra es falso.

Si una porción de una expresión es opcional y si su valor cierto no es requerido por la expresión completa para ser verdadera (esto es escogido por el diseñador de la interfaz), entonces ésta es encerrada entre corchetes [...].

Si una expresión específica causa una transición a un estado, desde todos los otros estados del diagrama, una notación más corta es usada, en lugar de dibujar todas las transiciones individuales, se dibuja una flecha sin un estado como su origen para representar esta condición, y se asume que se origina en todos los estados (por ejemplo IFC o REN).

También power-off (POFS) es un estado válido de muchas funciones de interfaz y podría ser mostrado en todos los diagramas con una transición dirigiéndose al estado al ser alcanzado en el tiempo power-on, una forma más corta es usada mostrando el pseudomensaje pon, originando una transición al primer estado para entrar en él cuando es activada la función, como se muestra en la siguiente figura.

(a) (b)

Figura 3.3 Representacion del inicio de una función de interfaz (a)representación completa (b)notación abreviada.

Como se mencionó en el capítulo I son diez las funciones de interfaz desde el punto de vista del software y la descripción de cada una de ellas se encuentra en el estandar IEEE 488.1.

Como las descripciones de cada función es extensa solo se tomaron en cuenta cinco de ellas: SH, AH, T, L y C. Siendo estas las funciones principales con las que debe contar la interfaz.

3.1.1 Función Source Handshake (SH)

La función de interfaz SH le provee a un dispositivo la

capacidad para garantizar la apropiada transferencia de mensajes de multilínea.

Una secuencia handshake entrelazada entre la función SH y una o más funciones de acceptor handshake AH (Cada una contenida dentro de dispositivos separados) garantiza la transferencia asíncrona de cada mensaje de multilínea. La función de interfaz SH controla la iniciación y la terminación de la transferencia de un byte de mensaje de multilínea.

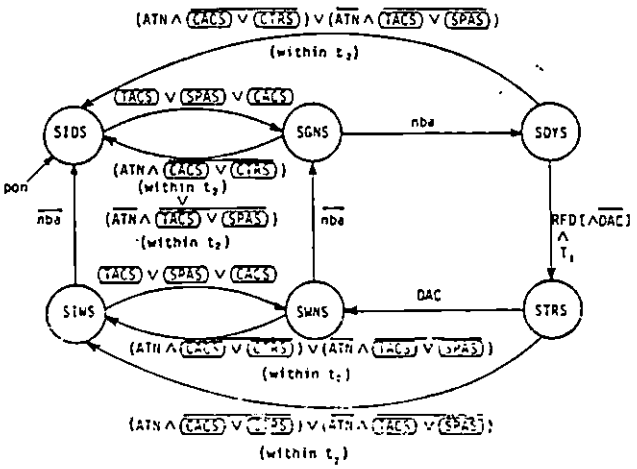


Figura 3.4 Diagrama de estado de la función SH

SIDS Source Idle State (Estado fuente ociosa)

En SIDS la función de interfaz SH está inactiva en el ciclo handshake y no hay aviso de un byte de mensaje disponible. La función SH inicia en SIDS. En este estado el mensaje DAV es puesto en falso pasivo.

La función SH saldrá de SIDS y entrará al estado de generar fuente (SGNS) si:

1. El estado talker activo (TACS) está activo.
2. o el estado de sondeo serie activo (SPAS) está activo
3. o el estado de controlador activo (CACS) está activo.

SGNS Source Generate State (Estado de generar fuente)

En SGNS el dispositivo está generando un nuevo byte de mensaje y la función esta esperando por el nuevo byte para hacerlo disponible.

En SGNS la función SH coloca el mensaje DAV en falso. En este estado el dispositivo puede cambiar el mensaje de multilínea, siendo colocado vía la función de interfaz Talker o Controller si una de ellas está en TACS, SPAS o CACS.

La función SH saldrá de SGNS y entra en:

1. El estado de fuente retardada (SDYS) si el mensaje de nuevo byte disponible (nba) es cierto.
2. El estado SIDS dentro de t2<200ns si ya sea:
   (a) El mensaje ATN es cierto y si ni CACS ni CTRS están activos.
   (b) o el mensaje ATN es falso y si ni TACS ni SPAS están activos.

SDYS Source Delay State (Estado fuente retardada)

En SDYS la función SH está esperando por un byte de mensaje para colocarlo en las líneas de señal de la interfaz, despues del cambio durante SGNS y por todas las funciones aceptoras al indicar su lectura para aceptar el byte de mensaje. En SDYS la función SH coloca el mensaje DAV falso. En este estado el dispositivo no cambiará el mensaje que esta siendo colocado. La función SH saldrá de SDYS y entra en:

1. El estado de transferencia de fuente (STRS) solamente despues de un tiempo T1>2us, swi el mensaje RED es cierto y si opcionalmente el mensaje DAC es falso.
2. El estado SIDS dentro de t2<200ns si ya sea:
   (a) El mensaje ATN es cierto y si ninguno de los estados CACS o CTRS estan activos.
   (b) o el mensaje ATN es falso y si ninguno de los estados TACS o SPAs estan activos.

STRS Source Transfer State (Estado de transferencia de fuente)

En STRS la función SH indica a la función AH que se está colocando continuamente un byte de mensaje válido.

En STRS la función SH coloca el mensaje DAV cierto. En este estado el dispositivo no cambiará tanto el mensaje de multilínea como el mensaje END que está siendo colocado (si es usado).

La función SH saldrá de STRS y entrará en:

1. El estado de espera de fuente ociosa (SIWS) dentro de t2<200ns si ya sea:
   (a) El mensaje ATN es cierto y ninguno dke los estados CACS ni CTRS están activos
   (b) El mensaje ATN es falso y ninguno de los estados TACS ni SPAS están activos.
2. El estado de espera de fuente para nuevo ciclo (SWNS) si el mensaje DAC es cierto.

SWNS Source Wait for New Cycle State_ (Estado de espera de fuente para nuevo ciclo)

En SWNS la función SH está esperando para que el dispositivo arranque un nuevo ciclo en la generación de mensaje. En la función puede poner el mensaje DAV cierto o falso. En este estado el dispositivo puede cambiar el mensaje de multilínea que está siendo colocado.

La función SH saldrá de SWNS y entrará en:

1. El estado SGNS si el mensaje nba es falso
2. El estado SIWS dentro de t2<200ns si ya sea:
   (a) El mensaje ATN es cierto y ninguno CACS ni CTRS están activos
   (b) o el mensaje ATN es falso y ninguno de los estados CACS ni SPAS están activos.

SIWS Source Idle Wait State (Estado de espera de fuente ociosa)

En SIWS la función SH no esta activa en el proceso de transferencia de bytes de mensaje externo, pero está activa en el proceso interno de espera para que el dispositivo arranque un nuevo ciclo de generación de mensaje. Este estado SIWS permite una secuencia de transferencia de byte de mensajes, para ser interrumpidos sin la pérdida de datos sobre la interfaz mientras que al mismo tiempo el dispositivo puede continuar preparándose para el nuevo (o próximo) ciclo de generación de byte de mensaje.

En SIWS el mensaje DAV será puesto en falso pasivo.

La función SH saldra de SIWS y entrará:
1. El estado SIDS si el mensaje nba es falso
2. El estado SWNS si ya sea:
   (a) El estado TACS está activo
   (b) o el estado SPAS esta activo
   (c) o el estado CACS está activo.

En la tabla 3.2 se especifica el set de mensajes y estados requeridos para efectuar la transición de un estado activo a otro.

Tabla 3.2
Mnemónicos para la función SH

Mensajes	Estados de Interfaz
pon=power on	SIDS=source idle state
nba=new byte available	SGNS=source generate state
ATN=attention	SDYS=source delay state
RFD=ready for data	STRS=source transfer state
DAC=data accepted	SWNS=source wait for new cycle state
	SYWS=source idle wait state
	TACS=talker active state (función T)
	SPAS=serial poll active state (función T)
	CACS=controller active state (función C)
	CTRS=controller transfer state (función C)

3.1.2 Función Acceptor Handshake (AH)

La función AH le provee al dispositivo la capacidad para garantizar la apropiada recepción de mensajes remotos de multilínea. Una entrelazada secuencia handshake entre una función SH y una o más funciones AH (cada una contenida dentro de dispositivos separados) garantiza la transferencia asíncrona de cada byte de mensaje. Una función AH puede retardar ya sea la iniciación o la terminación de la transferencia de un mensaje de multilínea hasta estar preparada para continuar con el proceso de transferencia.

La función AH utiliza los mensajes DAV, RED y DAC para efectuar cada transferencia de bytes de mensaje.

En la siguiente figura se muestra el diagrama de estado para la función Acceptor Handshake

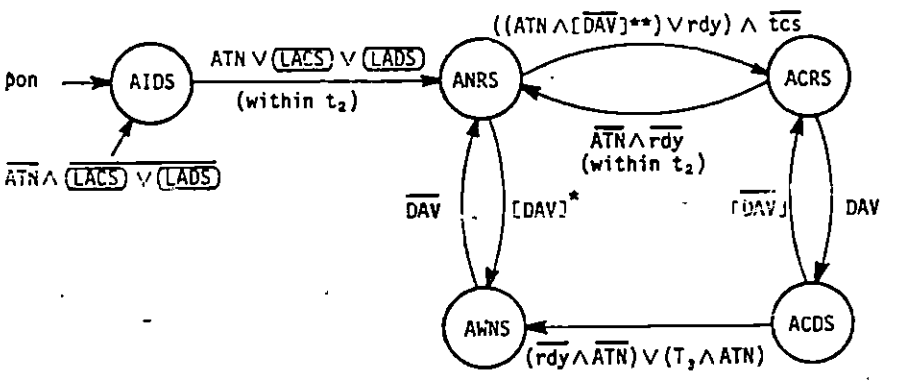


Figura 3.5 Diagrama de estado de la función AH

AIDS Acceptor Idle State (Estado aceptor inactivo)

En AIDS la función AH está inactiva y no ciclo handshake. La función AH inicia en AIDS. En este estado los menajes RED y DAC serán puestos en cierto pasivo.

La función AH saldrá de AIDS y entrará al estado dke aceptor no preparado (ANRS) dentro de t2<200ns si ya sea:

1. El mensaje ATN es cierto
2. o el estado LACS está activo
3. o el estado LADS está activo.

ANRS Acceptor Not Ready State (Estado de aceptor no preparado)

En ANRS la función AH indica a la interfaz que todavía no esta preparada para continuar con el ciclo handshake.
En ANRS los mensajes RED y DAC serán puestos en falso.
La función AH saldrá de ANRS y entrará a ya sea:

1. El estado ACRS si el mensaje tcs (take control synchronously) es falso y si
   (a) El mensaje ATN es cierto y el mensaje DAV (opcionalmente) es falso
   (b) o el mensaje rdy (ready for next message) es cierto.
2. El estado AIDS si el mensaje ATN es falso y si ninguno de los siguientes estados ocurre
   (a) LADS activo
   (b) LACS activo
3. El estado AWNS si opcionalmente, el mensaje DAV es verdadero (notar que esta transición nunca ocurrirá dentro de la operación normal de la interfaz)

ACRS Acceptor Ready State_(Estado de aceptor listo)

En ACRS la función AH indica a la interfaz que se está preparada para recibir mensajes de multilínea. En ACRS el mensaje DAC será enviado falso y el mensaje RED será enviado en cierto pasivo.

La función AH saldrá de ACRS y entrará a:

1. El estado de aceptar dato (ACDS) lsi el mensaje DAV es cierto
2. El estado AIDS si el mensaje ATN es falso y si:
   (a) LADS no está activo
   (b) o LACS no esta activo
3. El estado ANRS dentro de t2 si ambos mensajes ATN y rdy son falsos

ACDS Accept Data State (Estado de aceptar dato)

En ACDS la función AH indica a la función SH que se mantendrá un byte de mensaje válido. Este es el único estado en que los mensajes de multilínea en las líneas de señal DIO son válidos. El estado ACDS indica a las lfunciones de interfaz que un mensaje de interfaz está presente y es válido si el mensaje ATN es cierto. El estado ACDS indica a las funciones de dispositivo que un mensaje de dispositivo dependiente esta presente y validado si LACS está activo.

En ACDS los mensajes DAC y RED serán enviados como falso.
La función AH saldrá de ACDS y los mensajes entrarán a:

1. El estado de espera de aceptor para nuevo ciclo (AWNS) si:

(a) El mensaje ATN es cierto y un período de T3 ha culminado
(b) o los mensajes ATN y rdy son ambos falsos.

2. El estado AIDS si el mensaje ATN es falso y si:
   (a) LADS no está activo
   (b) LACS no está activo
3. El estado ACRS si opcionalmente el mensaje DAV es falso (notar que esta transición puede ocurrir solamente cuando el controlador toma el control asíncronamente)

AWNS Acceptor Wait for New Cycle State (Estado aceptor esperando para nuevo ciclo)

En AWNS la función AH indica que se ha recibido un byte de mensaje de multilínea.

En AWNS el mensaje RFD será enviado como falso y el mensaje DAC será puesto en cierto pasivo.

La función AH saldrá de AWNS y entrara a:

1. ANRS si DAV es falso
2. AIDS si el mensaje ATN es falso y si:
   (a) LADS no está activo
   (b) o LACS no está activo

En la tabla 3.3 se especifica el set de mensajes y estados requeridos para efectuar la transición de un estado activo a otro.

Tabla 3.3
Mnemónicos para la función AH

Mensajes	Estados de Interfaz
pon=power on	AIDS=acceptor idle state
rdy=ready for next message	ANRS=acceptor not ready state
tcs=take control synchonously	ACRS=acceptor ready state
ATN=attention	ACDS=accept data state
DAV=data valid	AWNS=acceptor wait for new cycle state
	LADS=listener addressed state (función L)
	LACS=listener active state (función L)

3.1.3 Función Talker (T o TE)

La función de interfaz T le provee a un dispositivo la capacidad para enviar datos dependientes de dispositivo (incluyendo datos de estatus durante una secuencia de sondeo serie) sobre la interfaz a otros dispositivos. Esta capacidad existe solamente cuando la función de interfaz es direccionada

para transmitir o conversar.

Existen dos versiones alternativas de la función: una con y una sin extensión de dirección. La función T normal usa una dirección de 1 byte, la dirección primaria talk. La función de interfaz T con extensión de dirección (llamada TE, extended talker) usa una dirección de 2 bytes, las direcciones talk primaria y secundaria. En todo lo demás, las capacidades de ambas versiones son las mismas y solamente una necesita ser implementada en un dispositivo específico.

En la figura 3.6 se muestra el diagrama de estado para implementar la función T.

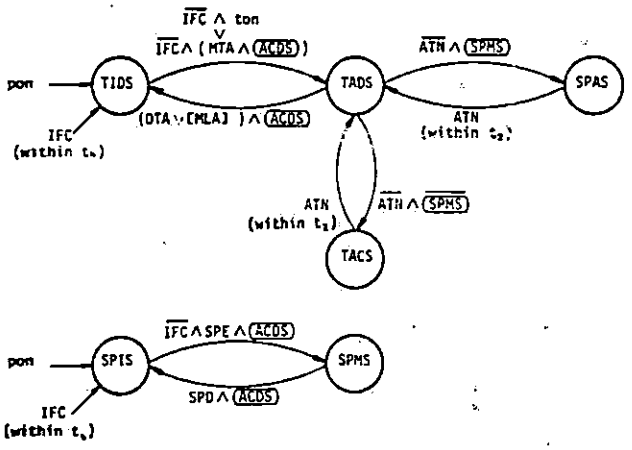


Figura 3.6 Diagrama de estado de la función T

TIDS Talker Idle State (Estado talker inactivo)

En TIDS tanto la función T como la función TE no están encargadas de enviar datos o bytes de estatus. La función T y TE inician en TIDS.

En TIDS los mensajes END y request service RQS (Servicio requerido) serán enviados en falso pasivo y el mensaje NUL será enviado en cierto pasivo.

TADS Talker Addressed State (Estado de talker direccionado)

En TADS la función T ha recibido su dirección talk y está preparada (pero no encargada) para enviar datos o bytes de estatus. En TADS la función TE ha recibido ambas direcciones talk, la primaria y secundaria y está preparada para (pero no encargada) enviar datos o bytes de estatus.

En TADS los mensajes END y RQS serán enviados en falso pasivo y el mensaje NUL será puesto en cierto pasivo.

TACS Talker Active State (Estado talker activo)

En TACS la función T o TE, habilita la transferencia de los mensajes data byte DAB (byte de dato) y END (si es usado), desde la función de dispositivo a las líneas de señal dde la interfaz. El contenido del mensaje está determinado solamente por la función de dispositivo. La función SH determina cuando la función de dispositivo puede cambiar el contenido del mensaje de DAB (y END si es usado).

Durante TACS los mensajes DAB o EOS (end of string) y END pueden ser puestos por las funciones de dispositivo. El mensaje RQS será puesto en falso pasivo.

SPAS Serial Poll Active State (Estado sondeo serie activo)

En SPAS la función t o TE habilita la transferencia de un único mensaje de estatus de la función de dispositivo a las líneas de señal de la interfaz usando la función SH para controlar la transferencia del byte de estatus que contiene ambos mensajes, RQS y STB.

Aunque un controlador necesita solamente un byte para los mensajes STB y RQS de un dispositivo, este está disponible para que el dispositivo repita este mensaje combinado si el controlador no afirma ATN despues de la primera transferencia. En este caso el contenido del mensaje STB puede cambiar entre subsecuentes transferencias aunque el mensaje RQS es sostenido sin alteración por la función SR.

SPIS Serial Poll Idle State (Estado de Sondeo Serie inactivo)

En SPIS la función T o TE no está habilitada para participar en una consulta o sondeo serie. La función T se activa en SPIS. El estado SPIS no provee capacidad de enviar mensajes remoto.

SPMS Serial Poll Mode State (Estado modo de sondeo serie)

En SPMS la función T o Te está habilitada para participar en un sondeo serie.

El estado SPMS no proveerá capacidad de enviar mensaje remoto.

En la tabla 3.4 se muestran el set de mensajes y estados requeridos para efectuar las transiciones de un estado activo a otro, los mensajes y estados corresponden a ambas funciones, T y TE.

Tabla 3.4
Mnemónicos de las funciones T y TE

Mensajes	Estados de Interfaz
pon=power on	TIDS=talker idle state
ton=talk only	TADS=talker addressed state
IFC=interface clear	TACS=talker active state
ATN=attention	SPAS=serial poll active state
OTA=other talk address	SPIS=serial poll idle state
MLA=my listen address	SPMS=serial poll mode state
MSA=my secondary address	TPIS=talker primary state
SPE=serial poll enable	TPAS=talker primary addressed state
SPD=serial poll disable	ACDS=accept data state (función AH)
PCG=primary command group	LPAS=listen primary addressed state (función L)
OSA=other secondary address	

3.1.4 Función Listen

La función de interfaz le provee al dispositivo la capacidad para recibir datos dependiente de dispositivo (incluyendo dato de estatus). Sobre la interfaz desde otros dispositivos. Esta capacidad existe solamente cuando la función está direccionada como listen (escuchar).

Existen dos versiones alternativas de la función, una con extensión de dirección y otra sin extensión de dirección. La función normal L usa un byte de dirección en tanto que la función L con dirección extendida usa 2 bytes y se denomina función LE.

En la figura 3 se muestra el diagrama de estado para la fuanción L y la función LE.

Estados de la función L:

LIDS Listener Idle State (Estado de Listener inactivo)

En LIDS la función L o LE no está encargada en la transferencia de mensasjes dependientes de dispositivos. La función L o LE inicia en el estado LIDS.

LADS Listener Addressed State (Estado de Listener direccionado)

En LADS la función L ha recibido su dirección listen (de escucha) y está preparada (pero no encargada) para la transferencia de mensajes dependientes de dispositivo. En LADS la función LE ha recibido ambas direcciones listen, la primaria y la secundaria.

El estado LADS no provee capacidad para enviar mensajes remoto.

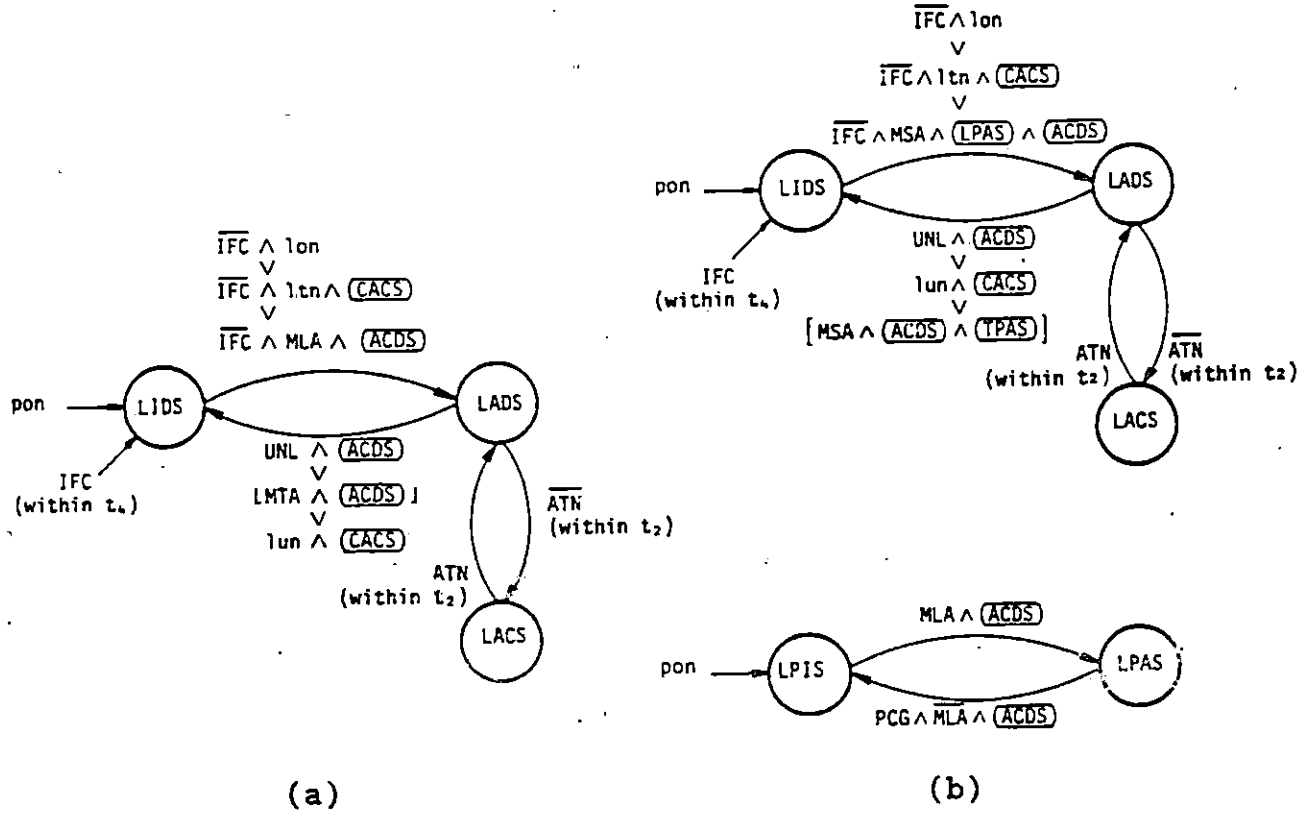


Figura 3.7 Diagrama de estado para la función
(a)L y (b) LE

LACS Listener Active State (Estado de listener activo)

En LACS la función L o LE, está habilitada para transmitir cualquier mensaje dependiente de dispositivo (DABe, EOS, STB, END o RQS) a las funciones de dispositivo como se recibió vía las líneas de señal de la interfaz. La función AH es usada por la función de dispositivo para controlar la transferencia de mensaje.

El estado LACS no provee capacidad para enviar mensajes remoto

LPIS Listener Primary Idle State (Estado de listener primario inactivo)

En LPIS la función LE está habilitada para reconocer su dirección primaria pero no su dirección secundaria. La función LE inicia en LPIS.

LPAS Listener Primary Addressed State (Estado de listener primario activo)

En LPAS la función LE está habilitada para reconocer y responder a su dirección secundaria.

En la tabla 3.5 se especifican el set de mensajes y estados requeridos para efectuar la transición de un estado activo a otro, tanto para la función L como para la función LE.

Tabla 3.5
Mnemónicos de la función L o LE

Mensajes	Estados de interfaz
pon=power on	LIDS=listener idle state
ltn=listen	LACS=listener active state
lun=local unlisten	LADS=listen addressed state
lon=listen only	LPIS=listener primary idle state
IFC=interface clear	LPAS=listener primary addressed state
ATN= attention	ACDS= accept data state (función AH)
UNL=unlisten	CACS=controller active state (función C)
MLA=my listen address	TPAS=talker primary addressed state (función T)
PCG=primary command group	
MTA=my talk address	
MSA=my secondary address	

3.1.5 Función Controller C (Controlador)

La función de interfaz C le provee a un dispositivo la capacidad para enviar direcciones de dispositivo, comandos universales y comandos direccionales a otros dispositivos sobre la interfaz. Esta también provee la capacidad para conducir sondeos paralelos y determinar que dispositivos requieren servicio.

Una función de interfaz C puede ejercer sus capacidades solamente cuando se está enviando el mensaje ATN sobre la interfaz.

Si más de un dispositivo en la interfaz tiene una función C, entonces todos menos uno de ellos estarán en el estado de controlador inactivo (CIDS) en algún tiempo dado. El dispositivo conteniendo la función C que no está en CIDS es llamado controlador en mandato (Controller-in-charge) del sistema de interfaz.

La función de interfaz C en uno de los dispositivos conectados a una interfaz (pero no más que uno) puede existir en el estado de control activo del sistema (SACS). Esta permanecerá en este estado a través de toda la operación de interfaz y también poseerá la capacidad para enviar los mensajes IFC y REN sobre la interfaz en cualquier momento, sea o no este el controlador en mandato. Este dispositivo es llamado el controlador del sistema.

La función C inicia en los estados CIDS, SRIS y SIIS,

aunque no todos los estados anteriores pueden existir en la función, y dependerá de los sub-sets disponibles en esa función.

Para comprenderlo mejor, es necesario observar detenidamente el diagrama de estado de la función C.

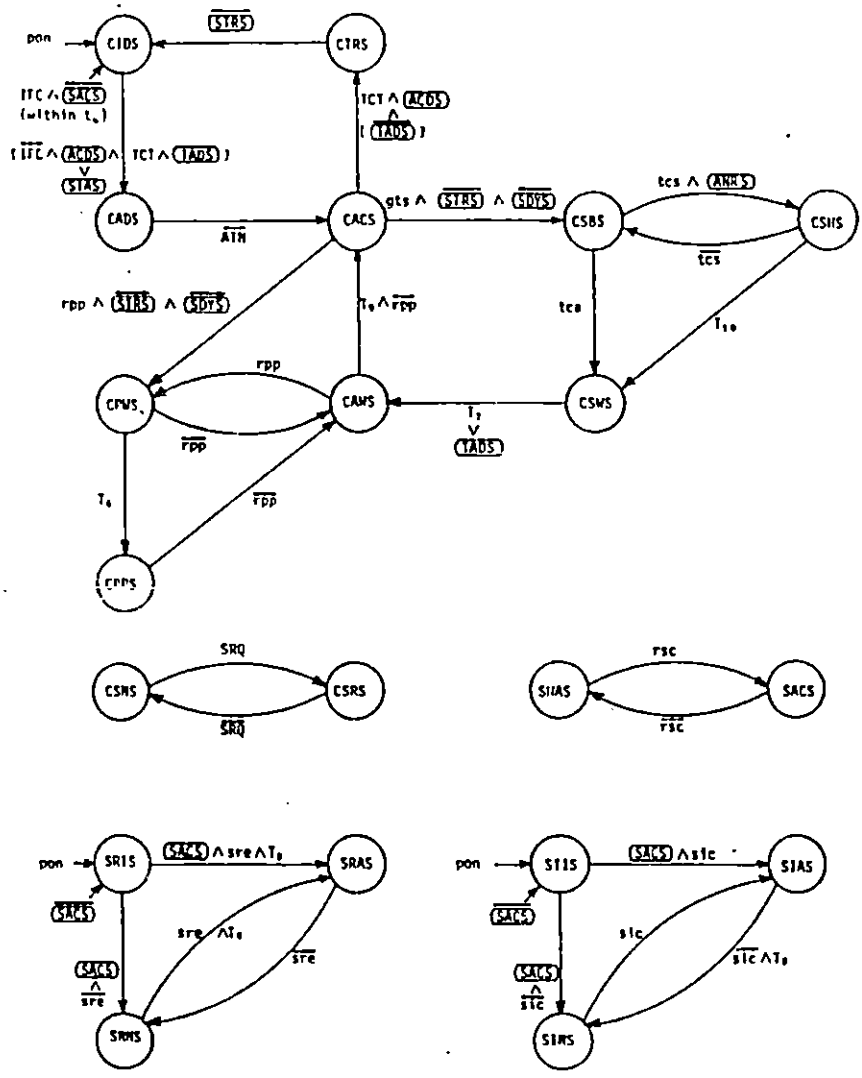


Figura 3.8 Diagrama de estado de la función C

Como se observa en la figura, existen 4 diagramas de estado para la misma función, esto se define asi por existir estados implícitos que ocurren al iniciar la función C. (por ejemplo: CIDS SRIS y SIIS).

El estado SACS (estado de control activo del sistema) tiene un propósito similar al estado CACS, estando el primero implícito sobre todos los demás estados.

Los estados CSNS y CSRS corresponden a la recepción de mensajes SRQ (Service Request).

Los estados SRIS, SRAS, y SRNS corresponden a la operación de colocar en estado remoto los dispositivos de la interfaz enviando el mensaje sre (send remote enable)j.

Los estados SIIS, SIAS y SINS definen la operación de limpiar la interfaz (Interface clear) enviando el mensaje sic (send interface clear).

Los estados CPWS y CPPS corresponden a la operación de sondeo paralelo conducida por el controlador.

El estado CSBS permite que dos o más dispositivos transfieran mensajes dependientes de dispositivo sobre la interfaz.

El estado CSWS permite tomar el control del bus por otro controlador de manera sincronizada.

En CACS la función C habilita la transferencia de mensajes de interfaz de multilínea desde las funciones de dispositivo a las líneas de señal de la interfaz. Estos mensajes incluyen direcciones de dispositivos, comandos universales, o comandos direccionados. La función SH determinará cuando la función de dispositivo puede cambiar el contenido del mensaje. Pero el contenido del mensaje es determinado solamente por la función de dispositivo.

El mensaje ATN será puesto continuamente cierto y el mensaje IDY será continuamente falso, mientras CACS está activo, dentro de cuyas condiciones cualquiera de los mensajes de multilínea pueden ser enviados por las funciones de dispositivo (ver sección 1.5.3 del capítulo I).

El estado SACS debe estar activo en todos los estados exceptuando en el estado CIDS de la función C.

En el anexo 4 se especifica el set de mensajes y estados requeridos para efectuar la transición de un estado activo a otro.

3.2 Protocolo de las Subrutinas básicas

Todas las subrutinas de las funciones a implementar se elaboran principalmente en lenguaje ensamblador. Las subrutinas de más bajo nivel son, como se mencionó anteriormente, las que corresponden a las funciones Source Handshake y Acceptor Handshake. Su estructura está definida por los estados de la función, existiendo además limitantes de tiempo determinados por los documentos estandar IEEE 488.1.

Las subrutinas en lenguaje ensamblador para estas secuencias de eventos (SH y AH) son simples, debido a que se ejecutan en una llamada, enlazando todos los estados comunes de las funciones y predeterminando los requerimientos de cada transición.

Dicho de otra forma; en una función la transición de un estado a otro requiere que cierta expresión sea verdadera o

falsa, dicha expresión puede estar enlazada con el estado de otra función.

Asi por ejemplo: en la función SH para pasar del estado SIDS al estado SGNS requiere que cualquiera de los estados TCAS (función T o TE), SPAS (función T o TE) o CACS (función C), estén activos.

Desde el punto de vista del software, esto significa que la función SH entrara a SGNS cuando sea invocada por las funciones T o TE o la función C.

Si se parte de estas premisas, las subrutinas a nivel de funciones básicas de interfaz tendrán la siguiente jerarquía:

I- Subrutinas básicas SH y AH, para comunicación entrelazada
II- Subrutinas T, L y C; para enviar datos y comandos; invocando las subrutinas SH y AH.
III- Subrutinas para funciones de dispositivo. Funciones SR, RL, GT y DC invocadas por C.

En conclusión, las funciones principales a implementar son las siguientes: SH, AH, T, L y C.

Debido a su funcionamiento como tarjeta controladora, las funciones SR (Service Request), RL (Remote/Local), GT (Group Trigger) y DC (Device Clear) son implementadas dentro de la función Controlador (C). Porque el controlador ejecuta todas estas funciones como comandos de interfaz, cuando el usuario las programa.

Por otro lado, la función PP (Parallel Poll-Sondeo Paralelo) no se implementará, debido a dos aspectos importantes; en primer lugar, son muy escasos los dispositivos capacitados para ejecutar la función PP y particularmente en la escuela de Ingeniería Eléctrica los dispositivos que pueden manejarse con el bus IEEE488 no cuentan con la función PP. En segundo lugar, la información que provee la función PP es solo para identificar que dispositivo requiere servicio, pero no para conocer que tipo de servicio es requerido, esto solo es posible lograrlo con un protocolo más complejo que lee el registro del byte de estatus del dispositivo utilizando un mensaje de requerimiento de servicio previamente identificado en el sondeo paralelo. La ventaja que presenta el sondeo paralelo es su rapidez para conocer que dispositivo requiere el servicio, cuando se trabaja en un sistema con una cantidad considerable de dispositivos.

Las funciones Source Handshake y Acceptor Handshake corresponden a las funciones básicas del bus, su estructura determina los tiempos límite para las transiciones de un estado a otro.

La secuencia entrelazada de estas dos funciones puede observarse mejor en un diagrama de flujo que corresponde a la secuencia handshake de la figura 3.9.

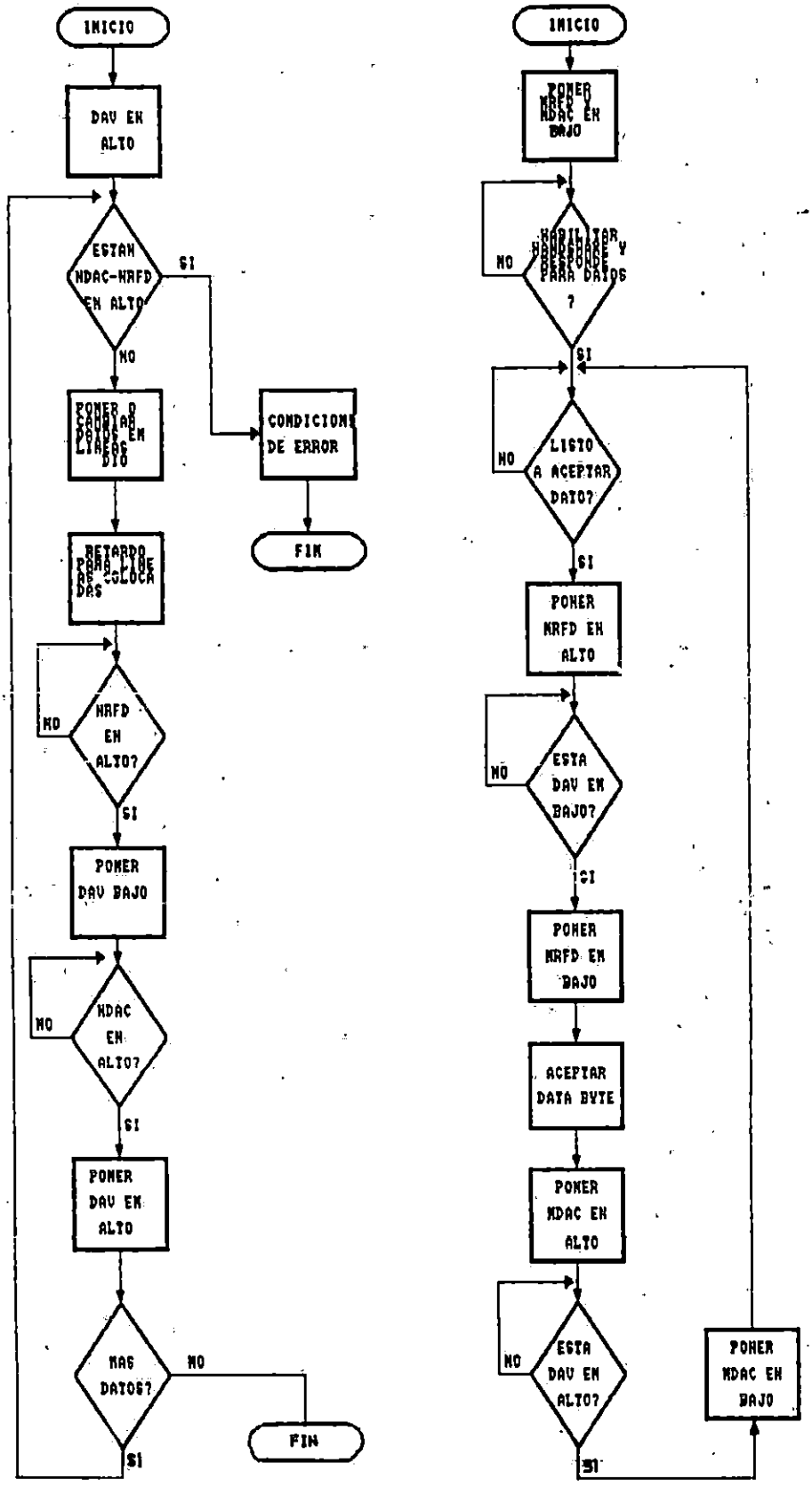


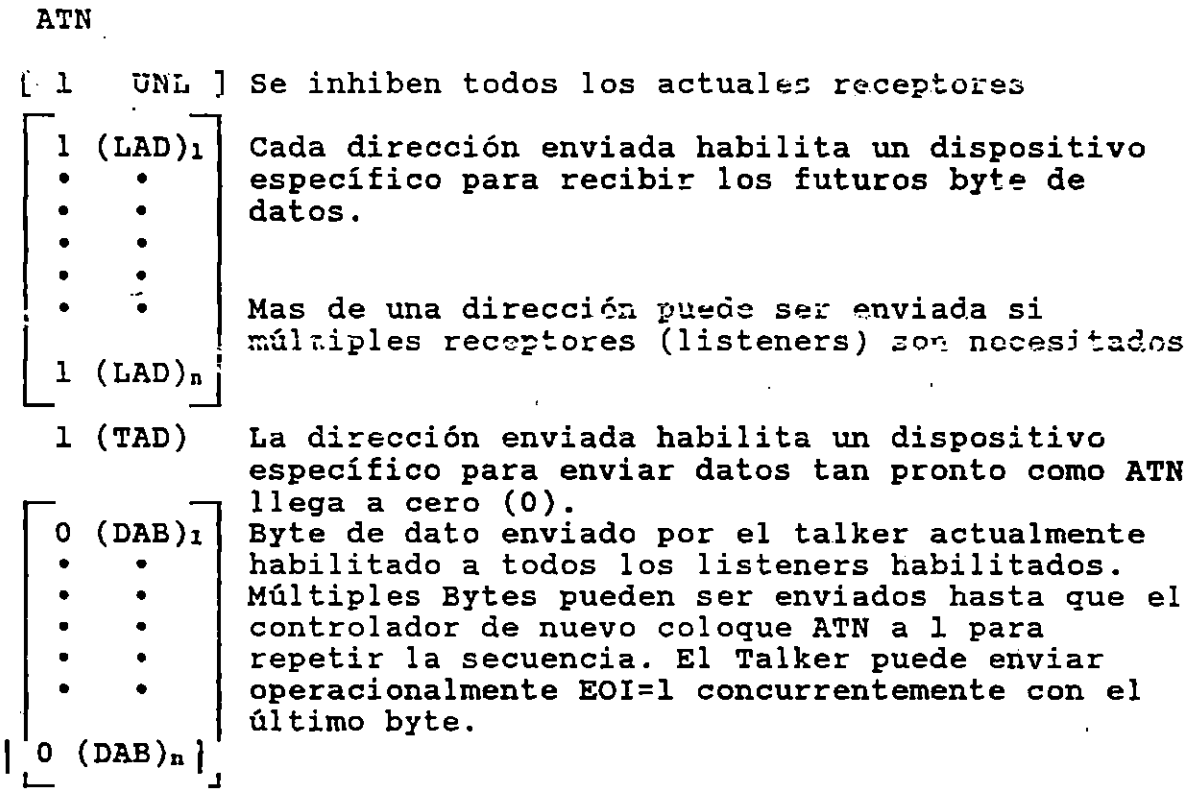
Figura 3.9 Diagrama de flujo de la secuencia handshake

Este flujo de eventos permite la transferencia acertada de datos dede un dispositivo emisor a otro receptor.

Comercialmente existen varios tipos de software para manejar interfaces IEEE 488, estos paquetes incluyen subrutinas o programas que permiten enviar o recibir mensajes, realizar sondeo serie o paralelo, enviar comandos, programas tareas, etc., estos programas pueden ser incluso invocados por lenguajes a través de llamadas a las subrutinas, la idea básica de estas subrutinas es que ejecutan automáticamente, las funciones necesarias para enviar o recibir mensajes completos.

Una subrutina para transmitir una cadena de strings, por ejemplo, solicita las direcciones de los dispositivos receptores, la cadena de strings a transmitir, direcciona automáticamente a los dispositivos y transmite el mensaje.

Asi la transferencia de datos puede tener la siguiente secuencia operacional:



Donde:

LAD=Listener Address (Dirección Listener)
TAD=Talker Address (Dirección Talker)
DAB=Data Byte (Byte de Dato)
UNL=Unlisten (Comando para inhabilitar direcciones de los dispositivos listeners)

La función talker se desarrolla a partir del momento en que se envía la dirección TAD. Cuando el controlador (la computadora) coloca dicha dirección, el dispositivo espera hasta que el controlador ordena la transmición de byte de datos (DAB) a través de la línea ATN.

En el caso particular, el computador como talker el procedimiento para transmitir el data byte (DAB) es más corto, el controlador direcciona los dispositivos para listen, activa la línea ATN e inmediatamente se ubica como talker para efectuar la rutina de Source Handshake.

El mismo caso ocurre, cuando el computador actúa como listener, este envía la dirección del dispositivo talker y se ubica como listener activado la línea ATN y efectuándo la rutina de Acceptor Handshake.

3.3 Protocolo para funciones Talker y Listen

Al momento de diseñar los programas para estas dos funciones debe comprenderse como el computador ejecuta dichas funciones asi se tiene que, para ejecutar la función talker (equivalente a transmitir datos desde el computador) la computadora "direcciona a los dispositivos como listen (que actúan como receptores y procede a enviar los bytes de datos utilizando para ello la función SH.

Para ejecutar la función listen se tienen dos opciones: la primera opción si solamente se requiere que el computador reciba el dato, se enviará una sola dirección listen; la segunda opción es cuando se desea que otros dispositivos reciban también el mensaje, se envían las direcciones listen de dichos dispositivos; finalmente en ambas opciones se utiliza la función AH para recibir datos.

Es importante dejar claro que el computador 'direcciona' a los dispositivos por medio de la función SH, dichas direcciones son enviadas por las líneas DIO como si fueran bytes de datos (DAB) pero con dos diferencias importantes:

1. La última dirección no se envía con un terminador de programa (PMT) ni con la línea EOI activada como sucede con la transmisión de datos.
2. La línea ATN es puesta en cierto (estado bajo) como corresponde a la transmisión de comandos y direcciones.

En la tabla 1.1 se puede apreciar la aseveración anterior en ella se observa que para MLA (my listen address) las líneas DIO señaladas con la letra L corresponden a las líneas en las que deberá enviarse la dirección del dispositivo listen (en

binario), las opciones son de 31 direcciones, las líneas DIO6 (=0) y DIO7 (=0), definen cuando es una dirección listen, la línea DIO 8 no se utiliza.

Por otro lado, para MTA (my talker address) las líneas DIO señaladas con T corresponden a las líneas en las que se deberá enviar la dirección del dispositivo talker (en binario) las opciones son también de 31 direcciones, las líneas DIO6 (=1) y DIO7 (=0), definen que la dirección enviada es para dispositivo talker, la línea DIO8 no se utiliza.

Las siglas MSA (my secondary address) corresponden a la dirección secundaria (para funciones extendidas de T o L) y es enviada después de haber transmitido una dirección MTA o MLA. Las direcciones OSA (other secondary address) y OTA (other talk address) cumplen con la finalidad de proporcionar más direcciones secundarias (para la primera_ o más direcciones talker (para OTA) en ambos casos se cumple al mismo fin, y es el de proporcionar más direcciones talker (ya sea de tipo secundaria o primaria) básicamente para cuando se procederá a realizar un sondeo serie.

Estos últimos tres tipos están muy relacionados con los estados de las líneas DIO6 y DIO7. Pero en general observando la tabla 1.1 para todas las direcciones la línea ATN es colocada en estado cierto.

Otra relación con las direcciones se encuentra en la tabla 1.2 correspondiente a la codificación ASCII para los datos en las líneas DIO; como las direcciones enviadas sobre dichas líneas, existe una relación implícita entre una dirección y un símbolo ASCII de esta tabla.

Observando detenidamente la parte baja de esta tabla, se puede encontrar que aparecen señalados los grupos de direcciones listen, talk y secundarias. Las direcciones listen se codifican desde el símbolo SP (32 en decimal) hasta el símbolo > (62 en decimal). Las direcciones talk se codifican desde el símbolo @ (64 en decimal) hasta el símbolo (94 en decimal). Las direcciones secundarias se codifican desde el símbolo ' (96 en decimal) hasta el símbolo ~ (126 en decimal).

La diferencia entre cualquiera de estos grupos da un valor de 31 direcciones. El significado que tiene esto es que si a un dispositivo se le ha dado la dirección 30 por ejemplo, el dispositivo tendrá en las líneas DIO una dirección listen de 62 en decimal o 3E en hexadecimal (equivalente al símbolo >) y una dirección talk en las líneas DIO de 94 en decimal o 5E en hexadecimal (equivalente al símbolo) y de igual forma para la dirección secundaria.

Básicamente la dirección colocada en el panel trasero del dispositivo por medio de los interruptores (ver figura 3.10)

no cambia. Que el dispositivo sea direccionado como talk, listen o con dirección secundaria, dependerá del valor en las líneas DIO6 y DIO7, las líneas DIO1 a la DIO8 permanecerán fijadas para un dispositivo específico (ver tabla 3.6).

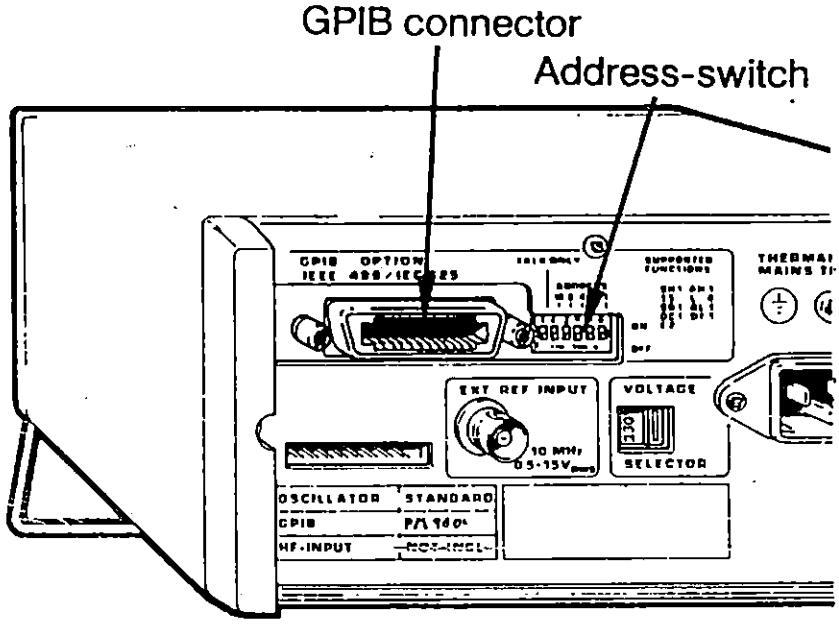


Figura 3.10 Aspecto de los interruptores para colocar la dirección del dispositivo.

Pero volviendo a los programas para las funciones talker y listen, en el primer caso (talker) el computasdor configura la interfaz para transmitir, solicita las direcciones de los dispositivos listen y luego el dato a transmitir, posteriormente se transmiten las direcciones con Source Handshake y a continuación los datos son transmitidos utilizando la función Source Handshake.

En el segundo caso el computador como listen se toma la opción de la comunicación bilateral, colocando una sola dirección talker, que es enviada con la función SH, a continuación es recibido el dato del dispositivo utilizando para ello la función AH.

El que una función opere como controlador o como talker gira muy en torno al estado de la línea ATN, pero no exclusivamente solo en esta, también existen los comandos o mensajes de unilínea (como GT, RL. SR) que son independientes de la línea ATN.

Tabla 3.6
Codificación de las direcciones talk y listen

CARACTER ASCII		Valor Decimal de 5 Bits
TALKER	LISTEN	
@	SP	00
A	!	01
B	"	02
C	#	03
D	$	04
E	%	05
F	&	06
G	'	07
H	(	08
I	)	09
J	*	10
K	+	11
L	,	12
M	-	13
N	.	14
O	/	15
P	0	16
Q	1	17
R	2	18
S	3	19
T	4	20
U	5	21
V	6	22
W	7	23
X	8	24
Y	9	25
Z	:	26
[	;	27
\	<	28
]	=	29
^	>	30

Un controlador le provee al usuario la capacidad para enviar direcciones y secuencias de control.

Las direcciones se refieren a los parámetros talk y listen determinados para una secuencia de control.

Mensajes de Dato:

1. Mensaje de Programa: es una secuencia de bytes de dato que es transferida desde el controlador a un dispositivo

con ATN falso.

2. Mensaje de Respuesta: es una secuencia de bytes de datos que es transferida desde el dispositivo al controlador con ATN falso.

Las secuencias de control envían uno o más mensajes remoto, asi algunas secuencias de control son combinaciones de otras secuencias de control. Si un controlador suple una secuencia de control que es la combinación de varias otras secuencias de control, el controlador también deberá suplir individualmente otras secuencias de control.

Un set de secuencias de control son proporcionadas por el estandar IEEE488.2:

1. Send Commands (Enviar Comandos)
2. Send Setup (Enviar Direcciones Listen)
3. Send Data Bytes (Enviar Bytes de Datos)
4. Send (Enviar un mensaje de Programa)
5. Receive Setup (Direcciones para Recibir una Respuesta)
6. Receive REsponse Message (Recibir Mensaje de Respuesta)
7. Receive (Ejecutar Mensaje de Respuesta)
8. Send IFC (Enviar IFC)
9. Device Clear (Limpiar Dispositivo)
10. Enable Local Controls (Habilitar Controles Locales)
11. Enable Remote (Habilitar Remoto)
12. Set RWLS (Deshabilitar mensaje local:retornar a local (rtl))
13. Send LLO (Enviar Comando LLO)
14. Pass Control (Pasar Control)
15. Perform Parallel Poll (Realizar sondeo paralelo)
16. Parallel Poll Configure (Configurar sondeo paralelo)
17. Parallel Poll Un Configure (Desconfigurar sondeo paralelo)
18. Read Status Byte (Leer el Byte de Estatus)
19. Trigger (Enviar el comando GET)

Dichas secuencias pueden intrepretarse también como un set de subrutinas de programa que el diseñador puede ejecutar dentro del programa que esté diseñando.

CONCLUSIONES

- Los protocolos definidos en el estandar requieren de cierto nivel de conocimiento en las areas de programacion, y en el diseño de dispositivos electronicos , particularmente de conocimiento sobre microprocesadores (como manejo de registros, buffer de datos, manejos de cola de datos, y una serie de comandos y codigos para dispositivos compatibles con este estandar.

- Es necesario conocer con mas detalle el funcionamiento de los protocolos de cada función de interfaz , para diseñar las rutinas que deberan ejecutar dichas funciones, resolviendo un gran número de detalles relacionados con los mismos. Como el caso de las funciones SR, RL y DC, cuyo protocolos son definidos dentro de las perspectivas del dispositivo, pero no como hay ninguna referencia para las mismas funciones ejecutadas desde el computador o función controladora.

REFERENCIAS BIBLIOGRAFICAS

-IEEE Standard Digital Interface for Programmable Intrumentation, ANSI/IEEE Std 488.1. Published by the Institute of Electrical and Electronics Engineers. June 16,1988.

-IEEE Standard Codes, Formats, Protocols and Common Comands for use with IEEE Std 488.1-1987, IEEE Standard Digital Interface for Programmable Instrumentation, IEEE Std 488.2-1992. Published by the Institute of Electrical and Electronics Engineers. December 28,1992.

CAPITULO IV
DISEÑO DEL PROGRAMA DE MANEJO DE LA INTERFAZ

Introducción

La interfaz diseñada en este trabajo, como se mencionó en el capítulo II, está basada fundamentalmente en el software, porque particularmente sus funciones y protocolos son desarrollados desde programas cuyos datos y señales son enviados al bus por medio de la PPI.

Esta PPI funciona como una interfaz que direciona los datos y las señales por varios puertos, lo que permite cierta reducción en el tamaño del software, lo que no ocurriría si se utilizaran solamente; compuertas lógicas, flip-flops y tranceivers.

Una interfaz basada en el software, estará en completa dependencia con la frecuencia o señal de reloj del microprocesador (en la computadora), esto obliga que al momento de elaborar los diversos programas, se deba tener un completo conocimiento de la duración de las instrucciones (número de ciclos de reloj del microprocesador) para que cuando se ejecuten las funciones con los protocolos correspondientes, éstos sean similares a los tiempos especificados por el estandar IEE488.1.

En este capítulo III se describen las subrutinas que manejan y comunican la interfaz con el bus. El software se desarrolla de manera evolutiva, teniendo de plataforma o base las funciones de bajo nivel, también llamadas funciones primitivas o de tablero, las cuales sirven de soporte para las funciones de alto nivel o funciones de dispositivo.

Estas son funciones compuestas que manipulan las operaciones de bajo nivel, manejando el bus automaticamente, estas funciones liberan al usuario de enfocarse en la colocación de mensajes dependientes de dispositivo que lo obligan a tener que conocer detalles de la GPBI, esto permite que las funciones de bajo nivel sean transparentes al usuario.

4.1 Descripción del Problema

El problema consiste principalmente en enviar y recibir señales a una interfaz por medio del computador, esta interfaz se comunica con otros dispositivos externos por medio de un bus cable estandarizado. Como se ha visto anteriormente esta comunicación debe cumplir con ciertos requerimientos, además el circuito de interfaz por su simplicidad introduce otras limitantes al programa.

Y por otro lado las señales recibidas deber ser interpretadas por el programa para que sean presentadas al usuario, también es necesario codificar las ordenes enviadas

por el usuario a la interfaz.

4.2 Análisis del Problema

Para elaborar el programa deben tomarse en cuenta los requerimientos señalados por el estandar. Dejando de lado aquellos requerimeintos que no son compatibles con el estandar.

En un inicio, algunas subrutinas que manejan la interfaz fueron escritas en lenguaje ensamblador porque permite un acceso más rápido y directo a la interfaz.

La subrutina para la función SH se presenta a continuación, en ella se han incluido los estados para identificar las transiciones, el registro BL contiene el estado de las líneas en el puerto B, de la PPI. El buffer de datos corresponde a DAB y ADR corresponde al buffer para las direcciones.

```
SIDS            MOV AL, BL AND 2H; ATN EN CIERTO O FALSO
                MOV DX, PORT B
                OUT DX, AL
SGNS;       MOV AL, BL AND OFEH; LINEA DAB
                MOV DX, PORTB
                OUT DX, AL
                MOV DX, PORTC; DETECTANDO SI NRFD Y NDAC
                IN AL, DX          ; ESTAN EN ALTO
                AND AL, 06H
                JE SDYS      ; NO
                MOV DX, OFFSET ER1; SI: ENTONCES ERROR
                MOV AH, 09H
                INT 21H
                JMP SALIDA
SDYS:       MOV AL, 80H      ; AVERIGUAR SI ES DATO
                AND AL, BL       ; O DIRECCION
            JE SDYS1
                LODS ADR         ; CARGAR DIRECCION
                JMP SDYS2
SDYS1:      LODS DABG
SDYS2:      MOV DX, PORTA
                OUTDX, AL   ; COLOCANDO EL DATO EN LINEAS DIO
                CMP AL, OAH ; ES EL TERMINADOR DEL PROGRAMA
                JNE NO_PMT
                MOV AL, BL OR 10H;  CARGAR EOI SI ES ASI
                MOV DX, PORTB
                OUT DX, AL
NO_PMT:     MOV CX, 1200H; TIEMPO DE RETARDO DE 6ms APROX.
                MOV DX, PORTC
DELAY1:     IN AL, DX ;  SENSANDO NFRD EN ALTO
                CMP AL, ODH
                JE STRS
                LOOPNE DELAY1  ;  RETARDO DE 6ms.
                MOV DX, OFFSET ER2;  ERROR DE TIMEOUT
```

```
                MOV AH, 09H
                INT 21H
                JMP SALIDA
STRS:       MOV AL, BL OR01; COLOCANDO DAV EN CIERTO
                MOV DX, PORTB  ; PARA VALIDAR DATOS
                OUTJ DX, AL
SWNS:       MOV CX, 2F0FH; TIEMPO DE RETARDO DE 21ms APROX.
                MOV DX, PORTC
DELAY2:     IN AL, DX;  SENSANDO SI NDAC ESTA EN FALSO
                AND AL, 04H
                JE SG1  ;  SALTAR SI LO ESTA
                LOOPNE DELAY2
                MOV DX, OFFSET ER3;  ERROR DE TIMEOUT
                MOV AH, 09H
                INT 21H
                JMP SALIDA
SG1:            CMP SI, DI;  VERIFICANDO SI FUE ULTIMO BYTE
                JNE SGNS
                MOV AL, BL OR 80H; SI ES ASI, LIMPIAR PUERTO B
                MOV DX, PORTB
                OUT DX, AL  ;  Y RETORNAR
SALIDA:     RET
SH              ENDP
```

La subrutina detecta tres tipos de errores: dos de timeout, que corresponden a los tiempos de espera excedidos y un error de ambiguedad en las líneas NDAC y NRFD. Estos errores provocan una salida abrupta de la subrutina, desplegando un mensaje de error en la pantalla.

El último byte se detecta por medio del registro DI que contiene la cuenta del buffer de datos o direcciones a transmitir y es comparado con el registro SI que lleva el conteo de cada byte cargado.

La siguiente subrutina corresponde a la función Aceptor Handshake, el buffer de datos se encuentra en DAB1.

```
AIDS:       MOV DI, OFFSET DAB1
                MOV AL, 06H;  NRFD Y NDAC EN BAJO
                MOV DX, PORTB
                OUT DX, AL
ANRS:       MOV AL, 04H;  NRFD EN ALTO
                MOV DX, PORTB
                OUT DX, AL
                MOV CX, 1200H; TIEMPO DE ESPERA DE 6ms
                MOV DX,PORTC
ACRS:       IN AL, DX;  SENSANDO DAV EN BAJO
                AND AL, 01H
                JNE ACDS
                LOOPNE ACRS; TIEMPO DE ESPERA
                MOV DX, OFFSET ERA; ERROR DE TIMEOUT
```

```
                MOV AH, 09
                INT 21H
                JPM FINAL
ACDS:       MOV DX, PORTC
                IN AL, DX; DETECTANDO SI ES EL ULTIMO
                AND AL, 02H; BYTE A TRAVES DE LINEA EOI
                MOV BL, AL
                MOV AL, 06H; NRFD EN BAJO
                MOV DX, PORTB
                OUT DX, AL; ACEPTANDO BYTE DE DATO
                IN AL, DX; EN LINEAS DIO
                STOS DABl; ALMACENARLO EN BUFFER
                MOV DX, PORTB; COLOCANDO NDAC EN ALTO
                OUT DX, AL
                MOV CX, 1200H; TIEMPO DE ESPERA DE 6cm
                MOV DX, PORTC
AWNS:       IN AL, DX
                AND AL, 01H; DETECTANDO DAV EN ALTO
                JE EOSB
                LOOPNE AWNS
                MOV DX, OFFSET ERS; ERROR DE TIMEOUT
                MOV AH,09
                UNT 21H
                JMP FINAL
EOSB:       CMP BL, 02H
                JE FINAL
                MOV AL, 06H
                MOV DX, PORTB
                OUT DX, AL
                JMP ANRS
FINAL:      RET
AH              ENDP
```

Esta subrutina es un poco más corta que la anterior, y solo detecta tres tipos de errores, es más simple, porque solo debe aceptar datos, no direcciones. Esto es así porque la función AH está orientada a operar en conjunto con la función Listener, en tanto que la función SH está orientada a operar con la función Talker y la función Controller.

Ambas subrutinas están elaboradas como procedimientos que pueden ser llamados por las respectivas funciones T, L, y C.

Un paso previo, a todas las subrutinas es configurar la PPI, ya sea para transmitir datos y comandos o para recibir datos, solamente.

Para transmitir datos (función T) y comandos o direcciones (función C) la configuración es la misma, y se presenta de la siguiente manera:

```
PORTA EQU 03E8H
PORTB EQU 03E9H
PORTC EQU 03EAH
PCONT EQU 03EBH
```

```
COMCT EQU 0011H ; MODO DE SALIDA
COMLS EQU 0019H ; MODO DE ENTRADA
     MOV AL, COMCT;  Cargar modo de salida
     MOV DX, PCONT;  al registro de control
     OUT DX, AL
```

Para recibir datos (función L), la configuración es la siguiente:

```
     MOV AL, COMLS; Cargar modo de entrada
     MOV DX, PCONT; al registro de control
     OUT DX, AL
```

Cualquiera de ellas debe encabezar la subrutina respectiva, ya sea para transmitir o recibir mensajes, a traves de la interfaz.

Por otro lado este lenguaje es de bajo nivel y el usuario debe comprender como funciona el bus IEEE488, y debe diseñarse una manera de interpretar los datos. Es por esto que el programa final se realizó en programa Pascal (Turbo Pascal 4.0), debido a que puede tomarse ventaja de las opciones de video, con las que es posible construir: ventanas, menus, gráficos, etc. Además Turbo Pascal ofrece una intrucción (Port), que permite enviar y recibir datos entre la computadora y los puertos de la interfaz.

Las posibilidades de comunicarse con la interfaz y presentan al usuario un ambiente más integrado y accesible, permite que se puedan:

a. accesar datos más fácilmente
b. cambiar parámetros de los dispositivos
c. tabular datos
d. imprimir señales o datos
e. crear librerías de dispositivos, etc.

Pero para elaborar el programa deben cumplirse los requerimientos que señala el estandar IEEE488.2.

4.2.1 Requerimientos del Estandar IEEE488.2

Según el estandar IEEE488.2 deben haber un set definido de funciones que deben ser realizadas por una tarjeta controladora.

Para el área de programación se abordan cuatro aspectos importantes en cuanto a requerimientos:

1. Requerimiento de la función controlador:
   a) Debe responder a IFC, REN y SRD
   b) Enviar mensajes de interfaz
   c) Tomar mandato del sistema
   d) Tomar control sincronamente
2. Requerimientos de la función talker:
   a) Debe poseer los subsets T5-T8 o TE5-TE8

b) Capacidad completa de Source Handshake (SH1)

3. Requerimientos de la función Listen:
   a) Debe poseer los subsets I3-L4 o LE3-LE4
   b) Capacidad completa de Acceptor Handshake (AH1)
4. Requerimientos para pasar Control:
   a) Debe poseer los subsets C1, C2, C3 y C4 para realizar los protocolos de pasar el control y tomar control sincronamente.
   b) Debe poseer los subsets T5, T6, o TE5-TE6 que también soportan sondeo serie.

También se encuentran requerimientos adicionales que proveen control de bajo nivel del bus al usuario, además de ciertas facilidades para detectar estados y enviar o recibir mensajes.

Estos son:

1. Control de bajo nivel del bus.
   a) Pulsar IFC cierto por un tiempo mayor de 100 us
   b) Colocar la línea de señal REN en cierto o falso
   c) Enviar separadamente o en cualquier combinación cualquier mensaje definido en el estandar IEEE488.1.
   d) Enviar y detectar el mensaje END (esto es, la línea de señal EOI en cierto, y la línea ATN en falso al mismo tiempo con un byte de dato.
2. Facilidad para enviar o recibir los códigos, formatos, protocolos y comandos comunes definidos en el estandar.
3. Facilidad para sensar el estado de la línea SRD.
4. Facilidad para sensar las trancisiones de la línea SRD.
5. Facilidad para examinar el byte de estatus en base a un bit.
6. Facilidad para detectar una condición de error.
7. Facilidad para colocar tiempos de espera (timeout) en los intercambios de mensaje. Un timeout ocurre cuando una o varias secuencias handshakes no son completadas en un tiempo determinado.
8. Debe poseer un buffer de entrada para mensajes de programa.

Existen muchos otros requerimientos relacionados con el controlador y la transmisión de mensajes, sin embargo no es objetivo de este trabajo, el introducirse en estos aspectos que son por otra parte muy extensos.

4.3 Programa principal (estructura)

El lenguaje Pascal tiene cierta particularidad, y es que generalmente el programa principal es corto, porque hace llamadas a procedimientos o funciones y estos a su vez pueden llamar a otros procedimientos y así sucesivamente, todos estos procedimientos generalmente no forman parte del bloque principal del programa y pueden estar contenidos en otras unidades separadas.

La estructura del programa principal se muestra en el flujograma de la figura 4.1, la descripción de sus procedimientos se muestra a continuación:

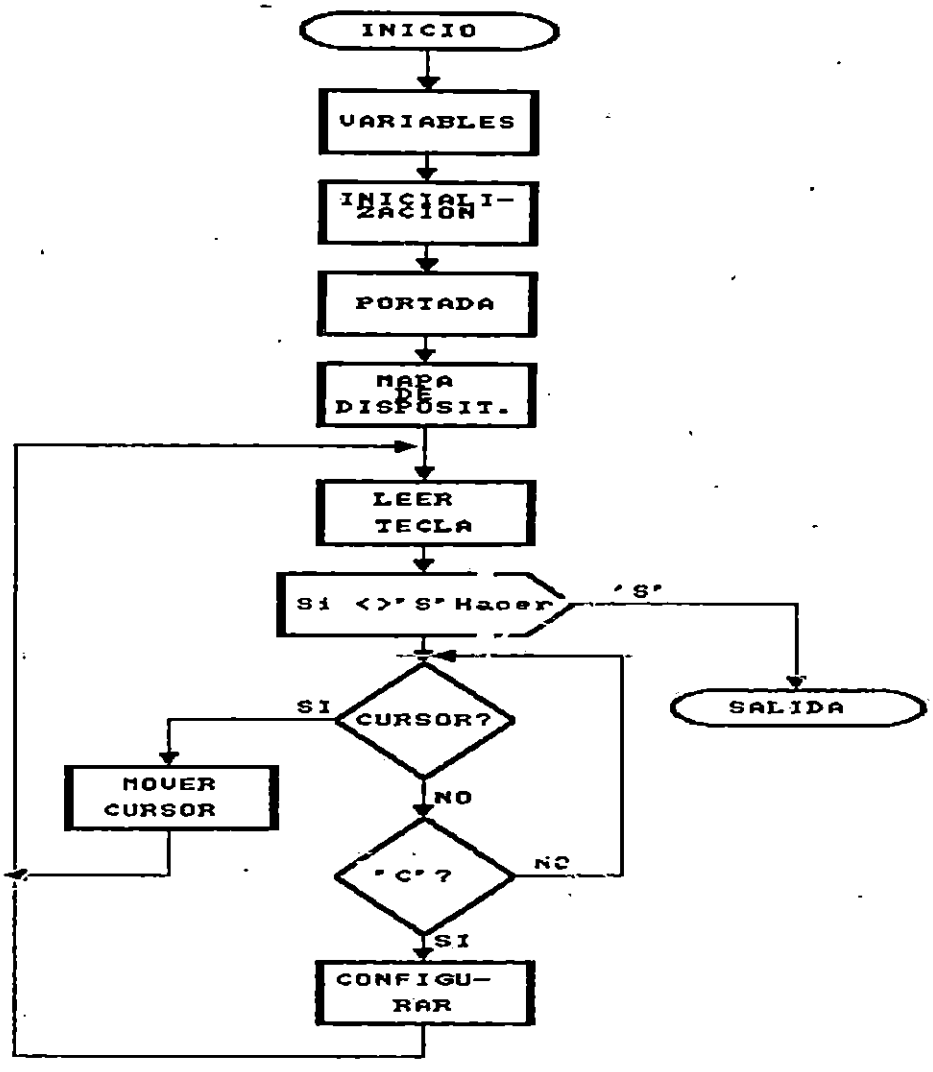


Figura 4.1 Diagrama de Flujo del Programa Principal

4.3.1 Variables

Este procedimiento inicializa todas las variables globales y arreglos utilizados en el programa.

4.3.2 Inicialización

Este procedimiento es importante, porque inicializa la operación en modo gráfico tanto para la unidad GRAPH asi como para la unidad GRAPHIX TOOLBOX. En este procedimiento se selecciona el tipo de tarjeta gráfica, para el caso en

particular la tarjeta escogida es una Hercules Monocromática, el modo gráfico escogido es entonces Hercmonohi para la unidad GRAPH y HRC para la unidad GRAPHIX TOOLBOX.

Si existe algún error en este punto el programa es abortado inmediatamente.

4.3.3 Portada

Este procedimiento presenta en la pantalla, la portada del programa, la que consta de las siglas del programa, el nombre completo del mismo y el año. Esta solo aparece por 3 segundos aproximadamente.

4.3.4 Mapa de Dispositivos

Es el siguiente procedimiento que aparece en la pantalla después de la presentación, este presenta el mapa de dispositivos del sistema, el que a su vez consta de la interfaz GPIB-1 instalada en el computador y que se conecta a 10 dispositivos. Este mapa permanece en la pantalla durante todo el programa para que el usuario conozca en todo momento que dispositivos están conectados al sistema.

4.3.5 Teclado

Este procedimiento lee el teclado y permite ejecutar otros dos procedimientos o bien salir del programa.

La estructura de este procedimiento es parte del programa principal (ver figura 4.1).

4.3.6 Mover Cursor

El nombre de mover cursor es para definir el movimiento de la ventana de un dispositivo a otro. Para este procedimiento desplaza una ventana pequeña con un color de fondo distinto y sobre el nombre de un dispositivo de esta manera se da la impresión de estarse desplazando de un dispositivo a otro escoger con que dispositivo trabajar.

4.3.7 Configurar

Este procedimiento es fundamental en el programa, porque anida otros procedimientos que ejecutan todas las funciones del bus y elaboran otro conjunto de tareas.

El procedimiento configurar determina con que tipo de dispositivo se está trabajando, le asigna la dirección de dispositivo correspondiente y llama a otro procedimiento: operaciones.

La estructura del procedimiento se presenta en la figura

4.2.

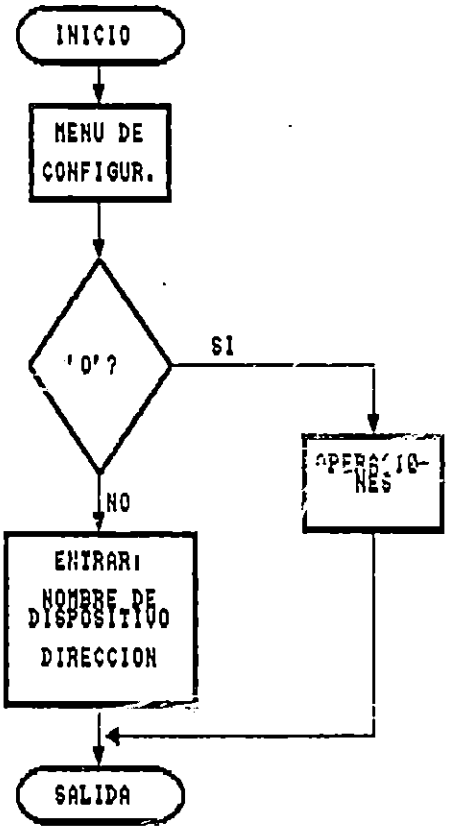


Figura 4.2 Flujograma del procedimiento Configurar

El procedimiento Operaciones ejecuta todas las tareas relacionadas con la operación del bus además crea listado de los datos recibidos, Envía comandos, Tabula datos, Grafica o reconstruye señales e Impresión.

El diagrama de flujo de este procedimiento se muestra en la figura 4.3.

Los procedimientos y funciones que ejecutan las funciones del bus son tres: Enviar Dato, Recibir Dato y Enviar Comando; estos procedimientos ejecutan los protocolos del estandar, y su estructura cumple las mismas funciones definidas en el capítulo III.

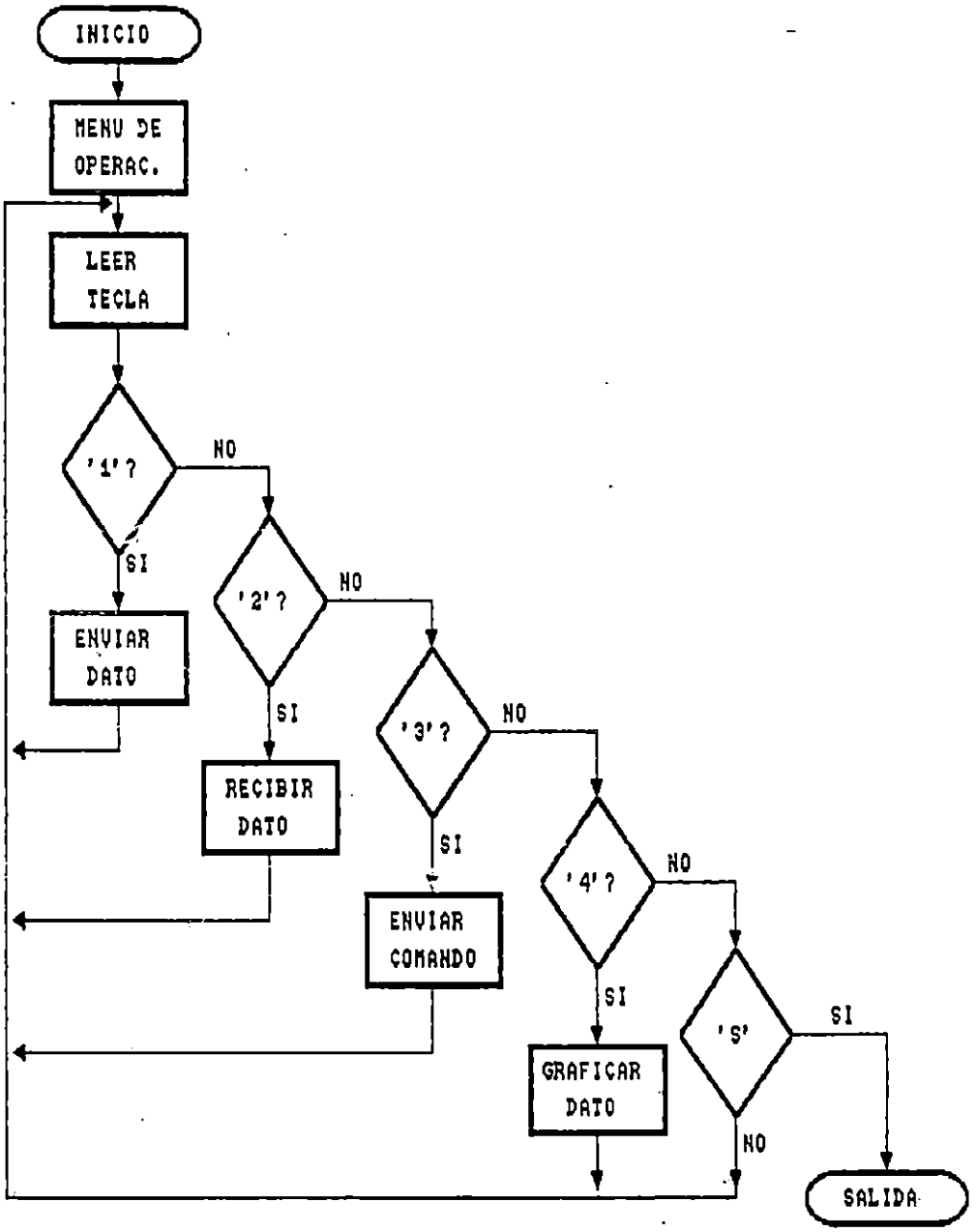


Figura 4.3 Flujograma del procedimiento Operaciones

Los otros dos procedimientos permiten que el usuario pueda darle formato a los datos obtenidos de los dispositivos, tabulando valores y reconstruyendo señales. El último procedimiento le permitirá al usuario imprimir los resultados

obtenidos de los procedimientos anteriores.

4.4 Guía de Usuario

La operación del programa es muy simple, para arrancar el programa se digita desde el prompt del DOS la palabra SM, y luego se presiona enter, el programa presenta una portada inicial por tres segundos, luego aparecerá el mapa de dispositivos conectados al sistema. En este mapa es necesario configurar el sistema por lo que solo es pueden utilizarse las teclas para mover el cursor, la tecla 's' para salir del programa, la tecla 'c' para configurar un dispositivo del sistema .

La tecla 'c' ofrece una segunda opción al presionarla aparece una ventana con las opciones a escoger:

- Nombre del dispositivo
- Dirección del dispositivo
- Siguiente dispositivo
- Operaciones

Para escoger cualquiera de las opciones anteriores basta con teclear la primera letra de cada opción.

Con las primeras dos opciones se le da al dispositivo seleccionado un nombre y una dirección. El nombre lo determina el usuario y es arbitrario,la dirección depende del dispositivo (debe ser la misma que especifica el manual del dispositivo o la que se le haya colocado en el panel trasero por medio de los interruptores dispuestos para este fin).

La tercera opción permite al usuario salir de este menú para seleccionar otro dispositivo del sistema.

La opción de operaciones es la que ejecuta la comunicación con la interfaz, esta presenta un menú como el siguiente:

1-Enviar Dato
2-Recibir Dato
3-Enviar Comando
4-Graficar Dato
5-Tabular Dato

Al escoger un número del menu se ejecuta la operación definida en el menú. La primera operación Enviar Dato solicita el nombre del dispositivo dado por el usuario en la configuración, luego el dato a enviar que finaliza tecleando enter inmediatamente el dato es enviado por la interfaz.

La segunda opción es mas simple, se solicita el nombre del dispositivo del que se recibirá el mensaje e inmediatamente se ejecuta la función listen, si existe un dato en el buffer del dispositivo la operación será satisfacoria y este será almacenado en el computador, en caso contrario aparecerá un mensaje de error.

La tercera operación es para enviar un comando o mensaje de interfaz (ver capitulo I). Se solicita el nombre del dispositivo y el nombre del comando (ver tabla 1.3).

La cuarta operación es en el caso de que se esté trabajando con el osciloscopio digital, esta operación es interna y se debe estar seguro de que el último dato recibido es la cadena de bytes correspondientes al dato de la pantalla del mismo,

esta operación toma los bytes almacenados en el buffer de la memoria del computador correspondientes a la señal del osciloscopio, y reconstruye la señal en el monitor de video. Esta operación es exclusiva de este dispositivo.

La quinta operación permite tabular uno o mas datos recibidos en la operación 2. Presentando en la pantalla el dato introducido desde la interfaz.

Para salir del menú de operaciones basta con presionar la tecla 's', y retorna al mapa de dispositivos, para trabajar con las opciones originales.

CONCLUSIONES

- El problema para crear el programa se basa en las limitantes del circuito y los requerimientos del estandar. El programa debe producir un resultado más accesible al usuario del bus IEEE488.

- El programa puede incluir más procedimientos que faciliten al usuario manejar los dispositivos desde el computador, creando procedimientos de un nivel más alto que le permitan construir y archivar librerías sin tener que modificar el programa fuente.

REFERENCIAS BIBLIOGRAFICAS

-Murray, William H.y Pappas, Chris H., 80386/80286 Programación en Lenguaje Ensamblador. Traducido por:Juan Manuel Sanchez Perez. Madrid, España. Editorial Osborne /McGraw Hill.1987.

-Turbo Pascal Graphix Toolbox versión 4.0, Borland International versión 1987.

-Turbo Pascal Owner's Handbook versión 4.0, Borland International versión 1987.

CONCLUSIONES GENERALES Y RECOMENDACIONES

CONCLUSIONES

- El estudio del bus IEEE488 no forma parte de los programas de las asignaturas de la carrera, además la escuela no cuenta con la suficiente documentación sobre el funcionamiento y las propiedades de este bus.

- Actualmente los circuitos de interfaz del bus IEEE488 asi como los programas de manejo del mismo han evolucionado mucho y comercialmente se encuentran programas muy sofisticados y con muchas ventajas de operación pero la adquisición de éstos es en algunos casos costosa.

- El circuito construido en este trabajo es sencillo, particularmente en su operación, pero esta simplicidad provoca que el programa de manejo sea más complicada al no poder la interfaz por si misma ejecutar los protocolos de las funciones de interfaz.

- El programa de manejo de la interfaz presentado en este trabajo es relativamente corto, anida una serie de procedimientos, desde un nivel alto hasta uno más bajo, en donde la interfaz se comunica directamente con el bus. El programa ofrece muchas perspectivas principalmente en la creación de ambientes más integrados, que permitan crear librerías, hacer análisis y controlar los dispositivos más rápidamente.

RECOMENDACIONES

-Se recomienda incorporar el estudio del bus IEEE488 dentro de la asignatura de Instrumentación Electrónica, no sólo como parte teórica de la misma, sino también en las prácticas de laboratorio, proporcionando al estudiante el conocimiento teórico-práctico para operar a distancia aparatos de instrumentación por medio del computador.

-Es necesario que la escuela trabaje en la recopilación bibliográfica sobre la información concerniente a este bus, y se actualize sobre los avances logrados hasta este momento, no solo en lo relacionado al hardware, sino también al software.

-Se recomienda investigar la posibilidad de adquirir microprocesadores que ejecuten las funciones de este bus, para mejorar diseños posteriores o que puedan ser incorporados dentro de dispositivos diseñados en trabajos posteriores.

-Sobre el programa de manejo, se recomienda introducir los cambios necesarios para crear librerías de dispositivos, herramientas para manejar datos y archivos y la posibilidad de ejecutar análisis numéricos con los datos obtenidos.

ANEXO A

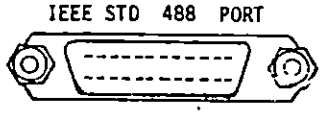
SUBSETS DE LAS FUNCIONES DE INTERFAZ

The reader is further reminded that full operational systems require detailed knowledge of the device dependent characteristics of each device in a system. These specifications are beyond the scope of this standard.

C2. Capability Identification Codes

It is recommended that an ANSI/IEEE Std 488.1-1987 capability code be placed below or near the interface connector on each device to identify the complete set of interface functions contained within that device. In this standard each interface function and allowable subset thereof has an equivalent alphanumeric code to identify that particular capability. All such interface function capability codes may be expressed in a concise alphanumeric string and marked on the exterior of the device to facilitate user system assembly.

For example, a device with the basic talker function, the ability to send status bytes, the basic listener function, a listen only mode switch, service request capability, remote local capability without local lockout, manual configuration of the parallel poll capability, complete device clear capability, no capability for device trigger, and no controller capability would be identified with the following code:



SH1, AH1, T2, L1, SR1, RL2, PP2, DC1, DT0, C0, E1

The code identifies the eight specific interface functions implemented. In addition, the type of electrical interface contained within the device is specified. The notation E1 is used to identify that open collector drivers are used (everywhere there is a choice) and the notation E2 is used to identify that three-state drivers are used (everywhere there is a choice). See 3.3.1.

The device designer can place any further device capability information, useful for system configuration, at the appropriate places on the physical equipment and in the relevant documentation for that equipment.

C3. SH Function Allowable Subsets

Identification	Description	States Omitted	Other Requirements	Other Function Subsets Required
SH0	no capability	all	none	none
SH1	complete capability	none	none	T1-T8, TE1-TE8, or C5-C28

C4. AH Function Allowable Subsets

Identification	Description	States Omitted	Other Requirements	Other Function Subsets Required
AH0	no capability	all	none	none
AH1	no capability	none	none	none

C7. L Function Allowable Subsets

Identification	Description			States Omitted	Other Requirements	Other Function Subsets Required
	Capabilities					
	Basic Listener	Listen Only Mode	Unaddress If MTA			
L0	N	N	N	all	none	none
L1	Y	Y	N	none	omit [MTA ∧ (ACDS)]	AH1
L2	Y	N	N	none	omit [MTA ∧ (ACDS)] lon always false	AH1
L3	Y	Y	Y	none	include [MTA ∧ (ACDS)]	AH1 and T1-T8 or TE1-TE8
L4	Y	N	Y	none	include [MTA ∧ (ACDS)] lon always false	AH1 and T1-T8 or TE1-TE8

C8. L Function (with Address Extension) Allowable Subsets

Identification	Description			States Omitted	Other Requirements	Other Function Subsets Required
	Capabilities					
	Basic Extended Listener	Listen Only Mode	Unaddress If MSA ∧ (TPAS)*			
LE0	N	N	N	all	none	none
LE1	Y	Y	N	none	omit [MSA ∧ (TPAS) ∧ (ACDS)]	AH1
LE2	Y	N	N	none	omit [MSA ∧ (TPAS) ∧ (ACDS)] lon always false	AH1
LE3	Y	Y	Y	none	include [MSA ∧ (TPAS)] ∧ (ACDS)]	AH1 and T1-T8 or TE1-TE8
LE4	Y	N	Y	none	include [MSA ∧ (TPAS) ∧ (ACDS)] lon always false	AH1 and T1-T8 or TE1-TE8

*Replaced by MTA when used together with the T function.

C9. SR Function Allowable Subsets

Identification	Description	States Omitted	Other Requirements	Other Function Subsets Required
SR0	no capability	all	none	none
SR1	complete capability	none	none	T1, T2, T5, T6, TE1, TE2, TE5 or TE6

C14. C Function Allowable Subsets

Identification*	Capabilities										Notes	State Required											Other Requirements		Other Function Subsets Required			
	System Controller	Send IFC and Take Charge	Send REN	Respond to SRQ	Send I.F. Messages	Receive Control	Pass Control	Pass Control to Self	Parallel Poll	Take Control Synchronously		SNAS, SACS	SIIS, SIAS, SINS	SRIS, SRAS, SRNS	CSNS, CSRS	CACS, CSBS, CSHS, CSWS, CAWS	CADS	CIDS	CTRS	CPWS, CPPS	[TCT ∧ (ACDS) ∧ (TADS)]†	[(TADS)]‡	ics not always false	C1	C2	AH1, L1-L4, or LE1-LE4	SH1	T1-T8, TE1-TE8
C0	N	N	N	N	N	N	N	N	N	N		O	O	O	O	O	O	O	O	O	O	O	O	O	—	—	—	—
C1	Y	—	—	—	—	—	—	—	—	—	(1)	R	—	—	—	—	—	—	—	—	—	—	—	R	—	—	—	—
C2	—	Y	—	—	—	—	—	—	—	—	(1),(6)	—	R	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
C3	—	—	Y	—	—	—	—	—	—	—	(1)	—	—	R	—	—	—	—	—	—	—	—	—	R	—	—	—	—
C4	—	—	—	Y	—	—	—	—	—	—	(1)	—	—	—	R	—	—	—	—	—	—	—	—	R	—	—	—	—
C5	—	—	—	—	Y	Y	Y	Y	Y	Y	(2),(3)	—	—	—	—	R	R	R	R	R	R	R	R	—	—	R	R	R
C6	—	—	—	—	Y	Y	Y	Y	Y	N	(2),(3)	—	—	—	—	R	R	R	R	R	R	R	O	—	—	R	R	R
C7	—	—	—	—	Y	Y	Y	Y	N	Y	(2),(3)	—	—	—	—	R	R	R	R	O	R	R	R	—	—	—	R	R
C8	—	—	—	—	Y	Y	Y	Y	N	N	(2),(3)	—	—	—	—	R	R	R	R	O	R	R	O	—	—	R	R	R
C9	—	—	—	—	Y	Y	Y	N	Y	Y	(2),(3)	—	—	—	—	R	R	R	R	O	R	R	O	—	—	—	R	R
C10	—	—	—	—	Y	Y	Y	N	Y	N	(2),(3)	—	—	—	—	R	R	R	R	O	R	O	O	—	—	R	R	R
C11	—	—	—	—	Y	Y	Y	N	N	Y	(2),(3)	—	—	—	—	R	R	R	R	O	R	O	O	—	—	—	R	R
C12	—	—	—	—	Y	Y	Y	N	N	N	(2),(3)	—	—	—	—	R	R	R	R	O	R	O	O	—	—	R	R	R
C13	—	—	—	—	Y	Y	N	N	Y	Y	(2)	—	—	—	—	R	R	R	O	R	O	O	O	—	—	—	R	R
C14	—	—	—	—	Y	Y	N	N	Y	N	(2)	—	—	—	—	R	R	R	O	R	O	O	O	—	R	R	R	—
C15	—	—	—	—	Y	Y	N	N	N	Y	(2)	—	—	—	—	R	R	R	O	R	O	O	O	—	R	—	R	—
C16	—	—	—	—	Y	Y	N	N	N	N	(2)	—	—	—	—	R	R	R	O	R	O	O	O	—	R	—	R	—
C17	—	—	—	—	Y	N	Y	Y	Y	Y	(2),(3),(4)	—	—	—	—	R	O	R	R	R	R	R	R	—	—	R	R	R
C18	—	—	—	—	Y	N	Y	Y	Y	N	(2),(3),(4)	—	—	—	—	R	O	R	R	R	R	R	R	—	—	—	R	R
C19	—	—	—	—	Y	N	Y	Y	N	Y	(2),(3),(4)	—	—	—	—	R	O	R	R	R	R	R	R	—	—	R	R	R
C20	—	—	—	—	Y	N	Y	Y	N	N	(2),(3),(4)	—	—	—	—	R	O	R	R	R	R	R	R	—	—	—	R	R
C21	—	—	—	—	Y	N	Y	N	Y	Y	(2),(3),(4)	—	—	—	—	R	O	R	R	O	R	R	R	—	—	R	R	R
C22	—	—	—	—	Y	N	Y	N	Y	N	(2),(3),(4)	—	—	—	—	R	O	R	R	O	R	O	O	—	—	—	R	R
C23	—	—	—	—	Y	N	Y	N	N	Y	(2),(3),(4)	—	—	—	—	R	O	R	R	O	R	O	O	—	—	R	R	R
C24	—	—	—	—	Y	N	Y	N	N	N	(2),(3),(4)	—	—	—	—	R	O	R	R	O	R	O	O	—	—	—	R	R
C25	—	—	—	—	Y	N	N	N	Y	Y	(2),(5)	—	—	—	—	R	O	R	O	R	O	O	R	—	—	R	R	R
C26	—	—	—	—	Y	N	N	N	Y	N	(2),(5)	—	—	—	—	R	O	R	O	R	O	O	R	—	—	—	R	R
C27	—	—	—	—	Y	N	N	N	N	Y	(2),(5)	—	—	—	—	R	O	R	O	R	O	O	R	—	—	R	R	R
C28	—	—	—	—	Y	N	N	N	N	N	(2),(5)	—	—	—	—	R	O	R	O	R	O	O	R	—	—	—	R	R

*Typical notation to describe a controller consists of the letter C followed by one or more of the numbers indicating the subsets selected. For example; C1, 2, 3, 4, 8.

†This is part of the CIDS to CADS transitional expression.

‡This is part of the CACS to CTRS transitional expression.

NOTES:

(1) One or more of subsets C1 through C4 may be chosen in any combination with any one of C5 through C28.
(2) Only one subset may be chosen from C5 through C28.
(3) The CTRS state must be included in devices which are to be operated in multicontroller systems.
(4) These subsets are not allowed unless C2 is included.
(5) These subsets are intended to be used in devices and systems where no control passage is possible.
(6) When a system controller asserts IFC during the time another physical device is operating as controller-in-charge, the system controller should refrain from active assertion of the source handshake and ATN until the removal of the IFC message to preclude multiple controller contention.

O = omit, R = required, hyphen = not applicable or not required, Y = yes, N = no.

C10. RL Function Allowable Subsets

Identification	Description	States Omitted	Other Requirements	Other Function Subsets Required
RL0	no capability	all	none	none
RL1	complete capability	none	none	L1-L4 or LE1-LE4
RL2	no local lock out	LWLS and RWLS	rtl always false	L1-L4 or LE1-LE4

C11. PP Function Allowable Subsets

Identification	Description	States Omitted	Other Requirements	Other Function Subsets Required
PP0	no capability	all	none	none
PP1	remote configuration	none	include [((PPD ∧ (PACS) ∨ PPU) ∧ (ACDS)] include [PPE ∧ (PACS) ∧ (ACDS)] exclude lpe	L1-L4 or LE1-LE4
PP2	local configuration	PUCS, PACS	include lpe exclude [((PPD ∧ (PACS)) ∨ PPU) ∧ (ACDS)] exclude [PPE ∧ (PACS) ∧ (ACDS)] local messages shall be substituted for S, P1, P2, P3	none

C12. DC Function Allowable Subsets

Identification	Description	States Omitted	Other Requirements	Other Function Subsets Required
DC0	no capability	all	none	none
DC1	complete capability	none	none	L1-L4 or LE1-LE4
DC2	omit selective device clear	none	omit [SDC ∧ (LADS)]	AH1

C13. DT Function Allowable Subsets

Identification	Description	States Omitted	Other Requirements	Other Function Subsets Required
DT0	no capability	all	none	none
DT1	complete capability	none	none	L1-L4 or LE1-LE4

C5. T Function Allowable Subsets

Identification	Description				States Omitted	Other Requirements	Other Function Subsets Required
	Capabilities						
	Basic Talker	Serial Poll	Talk Only Mode	Unaddress If MLA			
T0	N	N	N	N	all	none	none
T1	Y	Y	Y	N	none	omit [MLA ∧ (ACDS)]	SH1 and AH1
T2	Y	Y	N	N	none	omit [MLA ∧ (ACDS)]	SH1 and AH1
T3	Y	N	Y	N	SPIS, SPMS, SPAS	ton always false omit [MLA ∧ (ACDS)]	SH1 and AH1
T4	Y	N	N	N	SPIS, SPMS, SPAS	omit [MLA ∧ (ACDS)]	SH1 and AH1
T5	Y	Y	Y	Y	none	include [MLA ∧ (ACDS)]	SH1 and L1-L4 or LE1-LE4
T6	Y	Y	N	Y	none	include [MLA ∧ (ACDS)]	SH1 and L1-L4 or LE1-LE4
T7	Y	N	Y	Y	SPIS, SPMS, SPAS	ton always false include [MLA ∧ (ACDS)]	SH1 and L1-L4 or LE1-LE4
T8	Y	N	N	Y	SPIS, SPMS, SPAS	include [MLA ∧ (ACDS)] ton always false	SH1 and L1-L4 or LE1-LE4

C6. T Function (with Address Extension) Allowable Subsets

Identification	Description				States Omitted	Other Requirements	Other Function Subsets Required
	Capabilities						
	Basic Extended Talker	Serial Poll	Talk Only Mode	Unaddress if MSA ∧ (LPAS)			
TE0	N	N	N	N	all	none	none
TE1	Y	Y	Y	N	none	omit [MSA ∧ (LPAS) ∧ (ACDS)]	SH1 and AH1
TE2	Y	Y	N	N	none	omit [MSA ∧ (LPAS) ∧ (ACDS)]	SH1 and AH1
TE3	Y	N	Y	N	SPIS, SPMS, SPAS	ton always false omit [MSA ∧ (LPAS) ∧ (ACDS)]	SH1 and AH1
TE4	Y	N	N	N	SPIS, SPMS, SPAS	omit [MSA ∧ (LPAS) ∧ (ACDS)]	SH1 and AH1
TE5	Y	Y	Y	Y	none	ton always false include [MSA ∧ (LPAS) ∧ (ACDS)]	SH1 and L1-L4 or LE1-LE4
TE6	Y	Y	N	Y	none	include [MSA ∧ (LPAS) ∧ (ACDS)]	SH1 and L1-L4 or LE1-LE4
TE7	Y	N	Y	Y	SPIS, SPMS, SPAS	ton always false include [MSA ∧ (LPAS) ∧ (ACDS)]	SH1 and L1-L4 or LE1-LE4
TE8	Y	N	N	Y	SPIS, SPMS, SPAS	include [MSA ∧ (LPAS) ∧ (ACDS)] ton always false	SH1 and L1-L4 or LE1-LE4

ANEXO B

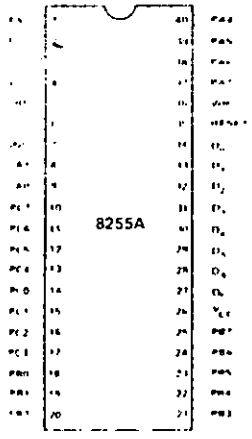
ESPECIFICACIONES DEL 8255A Y EL 74LS640

8255A/8255A-5
PROGRAMMABLE PERIPHERAL INTERFACE

- MCS-85™ Compatible 8255A-5
- 24 Programmable I/O Pins
- Completely TTL Compatible
- Fully Compatible with Intel® Microprocessor Families
- Improved Timing Characteristics
- Direct Bit Set/Reset Capability Easing Control Application Interface
- 40-Pin Dual In-Line Package
- Reduces System Package Count
- Improved DC Driving Capability

The Intel® 8255A is a general purpose programmable I/O device designed for use with Intel® microprocessors. It has 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation. In the first mode (MODE 0), each group of 12 I/O pins may be programmed in sets of 4 to be input or output. In MODE 1, the second mode, each group may be programmed to have 8 lines of input or output. Of the remaining 4 pins, 3 are used for handshaking and interrupt control signals. The third mode of operation (MODE 2) is a bidirectional bus mode which uses 8 lines for a bidirectional bus, and 5 lines, borrowing one from the other group, for handshaking.

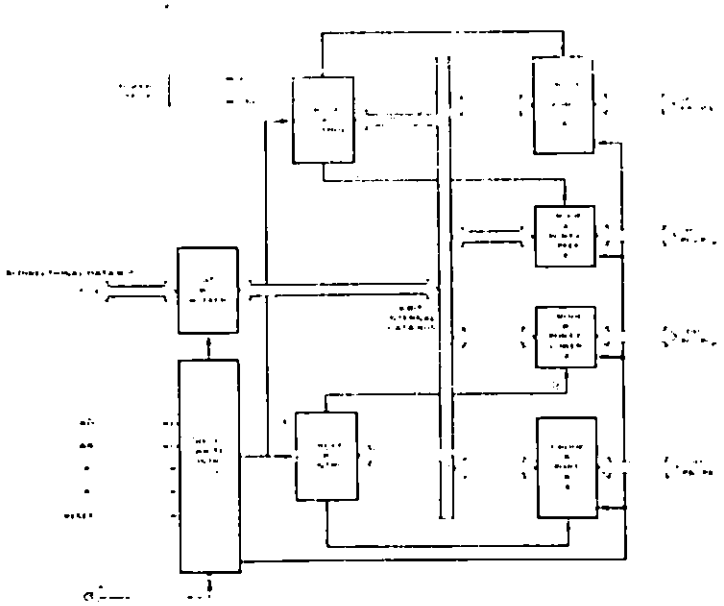
PIN CONFIGURATION



PIN NAMES

	DATA BUS (BI-DIRECTIONAL)
RESET	RESET INPUT
$\overline{CS}$	CHIP SELECT
$\overline{RD}$	READ INPUT
$\overline{WR}$	WRITE INPUT
A0, A1	PORT ADDRESS
PA7-PA0	PORT A (BIT)
PB7-PB0	PORT B (BIT)
PC7-PC0	PORT C (BIT)
V_{CC}	+5 VOLTS
GND	0 VOLTS

8255A BLOCK DIAGRAM



A.C. CHARACTERISTICS

T_A = 0°C to 70°C; V_{CC} = +5V ±5%; GND = 0V

NOTE: The 8255A-5 specifications are not final; some parametric limits are subject to change.

Bus Parameters

Read:

SYMBOL	PARAMETER	8255A MIN.	8255A MAX.	8255A-5 MIN.	8255A-5 MAX.	UNIT
t_{AR}	Address Stable Before READ	0		0		ns
t_{RA}	Address Stable After READ	0		0		ns
t_{RR}	READ Pulse Width	300		300		ns
t_{RD}	Data Valid From READ[1]		250		200	ns
t_{DF}	Data Float After READ	10	150	10	100	ns
t_{RV}	Time Between READs and/or WRITEs	850		850		ns

Write:

SYMBOL	PARAMETER	8255A MIN.	8255A MAX.	8255A-5 MIN.	8255A-5 MAX.	UNIT
t_{AW}	Address Stable Before WRITE	0		0		ns
t_{WA}	Address Stable After WRITE	20		20		ns
t_{WW}	WRITE Pulse Width	400		300		ns
t_{DW}	Data Valid to WRITE (T.E.)	100		100		ns
t_{WD}	Data Valid After WRITE	30		30		ns

Other Timings:

SYMBOL	PARAMETER	8255A MIN.	8255A MAX.	8255A-5 MIN.	8255A-5 MAX.	UNIT
t_{WB}	WR = 1 to Output[1]		350		350	ns
t_{IR}	Peripheral Data Before RD	0		0		ns
t_{HR}	Peripheral Data After RD	0		0		ns
t_{AK}	ACK Pulse Width	300		300		ns
t_{ST}	STB Pulse Width	500		500		ns
t_{PS}	Per. Data Before T.E. of STB	0		0		ns
t_{PH}	Per. Data After T.E. of STB	180		180		ns
t_{AD}	ACK = 0 to Output[1]		300		300	ns
t_{KD}	ACK = 1 to Output Float	20	250	20	250	ns
t_{WOB}	WR = 1 to OBF = 0[1]		650		650	ns
t_{AOB}	ACK = 0 to OBF = 1[1]		350		350	ns
t_{SIB}	STB = 0 to IBF = 1[1]		300		300	ns
t_{RIB}	RD = 1 to IBF = 0[1]		300		300	ns
t_{RIT}	RD = 0 to INTR = 0[1]		400		400	ns
t_{SIT}	STB = 1 to INTR = 1[1]		300		300	ns
t_{AIT}	ACK = 1 to INTR = 1[1]		350		350	ns
t_{WIT}	WR = 0 to INTR = 0[1]		850		850	ns

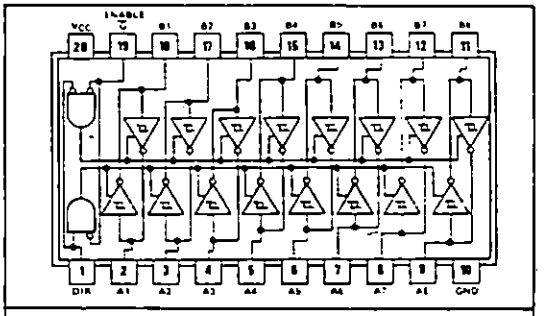
Notes: 1. Test Conditions: 8255A: C_L = 100pF; 8255A-5: C_L = 150pF.
2. Period of Reset pulse must be at least 50μs during or after power on. Subsequent Reset pulse can be 500 ns min.

TYPES SN54LS640 THRU SN54LS645, SN74LS640 THRU SN74LS645 OCTAL BUS TRANSCEIVERS

BULLETIN NO. DL-S 12574, APRIL 1979

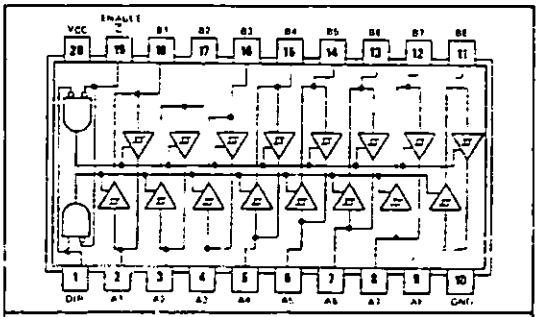
- SN74LS64X-1 Versions Rated at I_{OL} of 48 mA
- Bi-directional Bus Transceivers in High-Density 20-Pin Packages
- Hysteresis at Bus Inputs Improves Noise Margins
- Choice of True or Inverting Logic
- Choice of 3-State or Open-Collector Outputs

SN54LS640, SN54LS642 . . . J PACKAGE
SN74LS640, SN74LS642 . . . J OR N PACKAGE
(TOP VIEW)



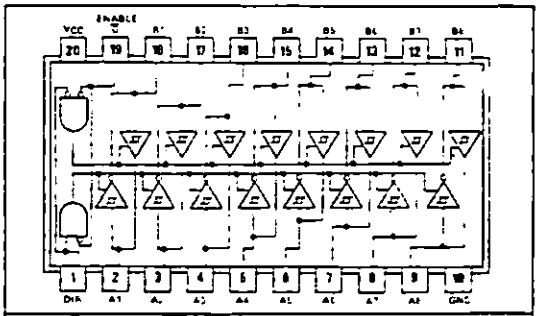
positive logic: see function table

SN54LS641, SN54LS645 . . . J PACKAGE
SN74LS641, SN74LS645 . . . J OR N PACKAGE
(TOP VIEW)



positive logic: see function table

SN54LS643, SN54LS644 . . . J PACKAGE
SN74LS643, SN74LS644 . . . J OR N PACKAGE
(TOP VIEW)



positive logic: see function table

description

These octal bus transceivers are designed for asynchronous two-way communication between data buses. The devices transmit data from the A bus to the B bus or from the B bus to the A bus depending upon the level at the direction control (DIR) input. The enable input ($\overline{G}$) can be used to disable the device so that the buses are effectively isolated.

DEVICE	OUTPUT	LOGIC
'LS640	3-State	Inverting
'LS641	Open-Collector	True
'LS642	Open-Collector	Inverting
'LS643	3-State	True and inverting
'LS644	Open-Collector	True and inverting
'LS645	3-State	True

FUNCTION TABLE

CONTROL INPUTS		OPERATION		
$\overline{G}$	DIR	'LS640 'LS642	'LS641 'LS645	'LS643 'LS644
L	L	B data to A bus	B data to A bus	B data to A bus
L	H	A data to B bus	A data to B bus	$\overline{A}$ data to B bus
H	X	Isolation	Isolation	Isolation

H = high level, L = low level, X = irrelevant

absolute maximum ratings

Same as SN54LS245 and SN74LS245 maximum ratings on page 6-13.

TEXAS INSTRUMENTS
INCORPORATED
POST OFFICE BOX 225012 • DALLAS, TEXAS 75265

recommended operating conditions

PARAMETER	SN54LS640 SN54LS643 SN54LS645			SN74LS640 SN74LS643 SN74LS645			SN74LS640-1 SN74LS643-1 SN74LS645-1			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC} (see Note 1)	4.5	5	5.5	4.75	5	5.25	4.75	5	5.25	V
High-level output current, I_{OH}			−12			−15			−15	mA
Low-level output current, I_{OL}			12			24			48	mA
Operating free-air temperature, T_A	−55		125	0		70	0		70	°C

NOTE 1: Voltage values are with respect to the network ground terminal.

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER			TEST CONDITIONS†		SN54LS640 SN54LS643 SN54LS645			SN74LS640 SN74LS643 SN74LS645			SN74LS640-1 SN74LS643-1 SN74LS645-1			UNIT
					MIN	TYP‡	MAX	MIN	TYP‡	MAX	MIN	TYP‡	MAX	
V_{IH}	High-level input voltage				2			2			2			V
V_{IL}	Low-level input voltage						0.5			0.6			0.6	V
V_{IK}	Input clamp voltage		V_{CC} = MIN,	I_I = −18 mA			−1.5			−1.5			1.5	V
	Hysteresis ($V_{T+} - V_{T-}$) A or B input		V_{CC} = MIN		0.1	0.4		0.2	0.4		0.2	0.4		V
V_{OH}	High-level output voltage		V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = V_{IL} max	I_{OH} = −3 mA	2.4	3.4		2.4	3.4		2.4	3.4		V
				I_{OH} = MAX	2			2			2			
V_{OL}	Low-level output voltage		V_{CC} = MIN V_{IH} = 2 V, V_{IL} = V_{IL} max	I_{OL} = 12 mA		0.25	0.4		0.25	0.4		0.25	0.4	V
				I_{OL} = 24 mA					0.35	0.5		0.35	0.5	
				I_{OL} = 48 mA								0.4	0.5	
I_{OZH}	Off-state output current, high-level voltage applied		V_{CC} = MAX, V_O = 2.7 V	$\overline{G}$ at 2 V,			20			20			20	μA
I_{OZL}	Off-state output current, low-level voltage applied		V_{CC} = MAX, V_O = 0.4 V	$\overline{G}$ at 2 V,			−400			−400			−400	μA
I_I	Input current at maximum input voltage	A or B	V_{CC} = MAX	V_I = 5.5 V			0.1			0.1			0.1	mA
		DIR or $\overline{G}$		V_I = 7 V			0.1			0.1			0.1	
I_{IH}	High-level input current		V_{CC} = MAX,	V_{IH} = 2.7 V			20			20			20	μA
I_{IL}	Low-level input current		V_{CC} = MAX,	V_{IL} = 0.4 V			−400			−400			−400	μA
I_{OS}	Short-circuit output current§		V_{CC} = MAX		−40		−225	−40		−225	−40		−225	mA
I_{CC}	Total supply current	Outputs high	V_{CC} = MAX,	Outputs open		48	70		48	70		48	70	mA
		Outputs low				62	90		62	90		62	90	
		Outputs at Hi-Z				64	95		64	95		64	95	

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.
‡ All typical values are at V_{CC} = 5 V, T_A = 25°C.
§ Not more than one output should be shorted at a time, and duration of the short-circuit should not exceed one second.

ANEXO C

LISTADO DEL PROGRAMA PRINCIPAL

```
Program SM488;

{+I Float.inc}
uses

dos,crt,graph,gdriver,printer,gkernel,gwindow;

const
     PA = $3E8;
     PB = $3E9;
     PC = $3EA;
     CONT = $3EB;
     DCT = $02;
     DLS = $01;
     COMCT = $088;
     COMLS = $098;

var
   x,Ch:char; AdT:integer;
   erd,EOI,ATN,NRFD,NDAC,DAV,REN:Boolean;
   d1,d2,d3,d4,d5,d6,d7,d8,d9,d10:string;
   grdriver,grmode,mx,my,ny,m,n,k:integer;
   DAB,InDAB: array [1..256] of char;
   DAT,PBF,t1,t2,RegC,RegCH,I:Integer;
   AD:array [1..10] of integer;
   er1,er2,er3,er4,er5:boolean;

Procedure inicial;
var

   graphdriver : integer;
   graphmode : integer;
   errorcode : integer;
begin
    initgraphic;
    graphdriver:=detect;

    initgraph(graphdriver,graphmode,'');
    errorcode := GraphResult;

    if errorcode <> grok then
       begin
          writeln('graphics error:',grapherrormsg(errorcode));
          writeln('Programa abortado..');
          halt(1);

       end;
    end;

Procedure Portada;
begin
    settextstyle(1,0,10);

    settextjustify(centertext,centertext);
    setbkcolor (0);
    Setcolor (10);
```

```
    outtextxy (Succ(getMaxX)div 2,
               Succ(GetMaxY)div 2 - 20,
               'SMB-488');
    Setlinestyle (2);
    Setcolor (9);
    Line (45,Succ(GetMaxY) div 4*3,Succ (GetMaxX) - 45,
          Succ(GetMaxY) div 4*3);
    Settextstyle (1,0,2);
    setcolor (7);
    outtextxy (Succ(GetmaxX) div 2,
                  Succ(GetmaxY) div 4 * 3 + 15,
                  'IEEE-488 Software Package');
    Settextstyle (1,0,1);
    outtextxy (Succ(GetmaxX) div 2,
               Succ(getMaxY) div 4*3 + 40,
               'Programa de Manejo para el bus IEEE-488');
    setcolor (9);

    Line (45,Succ(GetMaxY) div 4*3 + 60,Succ(GetMaxX) - 45,
          Succ(GetMaxY) div 4*3 + 60);
    Settextstyle (1,0,1);
    outtextxy (Succ(GetMaxX) - 30,
               Succ(GetMaxY) div 4*3 + 80,
               '1993');
end;

{*************** mapa de dispositivos ******************}

Procedure Mapa1;

begin
     rectangle (0,0,GetMaxX,GetMaxY);

     line (0,30,GetMaxX,30);
     line (0,Getmaxy-25,getmaxx,getmaxy-25);
     settextstyle (1,0,1);

     setcolor (15);
     settextjustify (1,1);
     outtextxy (succ(getMaxx)div 2,15,'MAPA DE DISPOSITIVOS');
     setcolor (9);
     settextstyle (0,0,1);
     setcolor (7);
     outtextxy (succ(GetMaxX) div 8,
                succ(GetmaxY) div 8,
                'GPIB-1');
     setcolor (9);
     Rectangle (Succ(GetMaxX) div 8 - 35,Succ(GetMaxY) div 8 - 8,
                Succ(getMaxX) div 8 + 30,Succ(GetMaxY) div 8 + 10);
     Bar (Succ(GetmaxX) div 8 - 30,Succ(GetMaxY) div 8 + 12,
          Succ(GetMaxX) div 8 + 25,Succ(GetMaxY) div 8 + 16);
          setcolor (7);


     mx:= Succ (GetmaxX) div 8;
     my:= Succ (GetMaxY) div 8 + 50;
```

```
     ny:= Succ (GetMaxY) div 8 +10;
     line (mx-35,ny-4,mx-45,ny-4);
     line (mx-45,ny-4,mx-45,ny+30);
     line (mx-45,ny+30,mx+84,ny+30);
     line (mx+84,ny+30,mx+84,ny+240);
     line (mx-35,ny+30,mx-35,ny+240);

     Outtextxy (mx,my+40,d1);
     Outtextxy (mx,my+80,d2);
     Outtextxy (mx,my+120,d3);
     Outtextxy (mx,my+160,d4);
     Outtextxy (mx,my+200,d5);
     Outtextxy (mx+120,my+40,d6);
     Outtextxy (mx+120,my+80,d7);
     Outtextxy (mx+120,my+120,d6);
     Outtextxy (mx+120,my+160,d9);
     Outtextxy (mx+123,my+200,d10);

                SETCOLOR (9);
     end;

{*****************************************************}


function Ready (Val:integer):Boolean;
var
   pm:integer;
begin
     pm:=(Val and $80);

     If pm = $80 then Ready:= true
     else Ready:= false;
end;

function Accepted (Valor:integer):Boolean;
var
   op:integer;

begin
     op:= (Valor and $40);
     If op = $40 then Accepted:= true
     else Accepted:= false;
end;

{************************************}

function Valid (Lin:integer):Boolean;

var
   da:integer;
begin
     da:= (Lin and $10);
     If da=$10 then Valid:= true
     else Valid:=false;
end;
```

```
{**************************************}

function Bytefinal (Linea:integer):Boolean;
var
   pd:integer;

begin
     pd:= (Linea and $20);
     If pd=$20 then Bytefinal:=true
     else Bytefinal:=false;
end;

{******************************************}

Procedure InData;
var
   t4:integer;


begin
     PBF:= (PBF and $E6);
     Port[PB]:= PBF;
     InDAB[n]:= Chr (Port[PA]);
     RegCH:= Port[PC];
     EOI:= Bytefinal (RegCH);
     PBF:= (PBF and $E2);
     Port[PB]:= PBF;
     repeat
      begin
           RegCH:= Port[PC];
           DAV:= Valid (RegCH);
           t4:= t4+1;
      end;
     until (t4=1000) or (DAV=false);
     If (t4=1000) and (DAV=true) then er4:=true;
end;


{************************************************}

Procedure PMT;


begin
     PBF:= (PBF or $10);
     Port[PB]:=PBF;
     EOI:= true;
end;

{**************************************}


Procedure DAC;
```

```
begin
     PBF:= (PBF and $0FE);
     Port[PB]:= PBF;
     If (DAB[n]=#10) or (EOI=true) then
     begin
          PBF:= (PBF and $EF);
          Port[PB]:= PBF;
          EOI:= false;
     end;
end;


{*****************************************}


Procedure RFD;
var
   er2:boolean;

begin
        PBF:= (PBF or $01);
        Port[PB]:= PBF;
        repeat
           begin
               RegC:= Port[PC];
               NDAC:= Accepted (RegC);
               t2:= t2+1;
           end;
        until (t2=1000) or (NDAC=false);
        If NDAC=false then DAC
        else er2:=true;
end;

{*****************************************}


Procedure SourceData;

var
   t1,t2:integer;


begin
     er1:=false;
     er2:=false;
     for I:=1 to n do
     begin
     Port[PA]:= Ord (DAB[I]);
     t1:=0;
     t2:=0;
     if DAB[n]=#10 then PMT;
     repeat
      begin
           RegC:=Port[PC];
           NRFD:=Ready (RegC);
           t1:=t1+1;
```

```
       end;
      until (t1=1000) or (NRFD=false);
      If NRFD=false then RFD
      else er1:=true;
      end;
end;

{************************************}

Procedure AcceptData;
var
   t3:integer;

begin
     I:= 1;
     repeat
       begin
            PBF:= (PBF or $06);
            Port[PB]:= PBF;
            PBF:= (PBF and $64);
            Port[PB]:=PBF;
            repeat
              begin
                   RegCH:= Port[PC];
                   DAV:= Valid (RegCH);
                   t3:=t3+1;
              end;
            until (t3=1000) or (DAV=true);
            If DAV=true then InData
            else er3:=true;
            I:=I+1;

       end;
     until (er4=true) or (er3=true) or (EOI=true) or (I=256);

end;

{************************************************************}

Procedure Lunid;
var
    u,x,r:integer;
begin
     write (Ch);
     Ch:=readkey;
     u:=0;
     val (Ch,x,r);
     repeat
      begin
           if u=x then AD[m]:=(u+$20)
           else u:=u+1;
      end;
     until (u>9) or (u=x);
     if u<>x then erd:=true else write (Ch);
end;
```

```
Procedure Ldec;
var u,x,r:integer;
begin
     write(Ch);
     Ch:=readkey;
     u:=0;
     val(Ch,x,r);
     repeat
     begin
          if u=x then AD[m]:=(u+$2A)
          else u:=u+1;
     end;
     until (u>9) or (u=x);
     if u<>x Then erd:=true else write (Ch);
end;

{*****************************************}

Procedure Tunid;
var
   u,x,r:integer;

begin
     write (Ch);
     u:= 0;
     val (Ch,x,r);
     repeat
          begin
               If u=x then AD[m]:= (u+$40)
               else u:=u+1;
          end;
     until (u>9) or (u=x);
     iF U<>X then erd:=true
     else write (Ch);
end;

Procedure Tdec;
var
   u,x,r:integer;
begin
     write (Ch);
     Ch:=readkey;
     u:=0;
     val (Ch,x,r);
     repeat
          begin
               if u=x then AD[m]:=(u+$4A)
               else u:=u+1;
          end;
     until (u>9) or (u=x);
     if u<>x then erd:=true
     else write (Ch);
end;
```

```
{***********************************************}

Procedure SourceAdd;
var
   t1,t2:integer;

begin
     er5:=false;
     Port[PA]:= AD[m];
     t1:=0;
     t2:=0;
     EOI:=false;
     ATN:=true;
     PBF:= PBF and $A0;
     Port[PB]:=PBF;
     repeat
          begin
               RegC:=Port[PC];
               NRFD:=Ready(RegC);
               t1:=t1+1;
          end;
     until (t1=1000) or (NRFD=false);
     if NRFD=false then RFD
     else er5:=true;
end;


{*****************************************************}

procedure EnviarDato;
var
   test:Boolean; Ed:array [1..3] of string;ej:integer;
begin
     Port[CONT]:= comct;
     Port[PC]:= DCT;
     PBF:= PBF and $E0;
     Port[PB]:=PBF;
     Definewindow(5,xmaxglb div 5*3,ymaxglb div 5*2,
                    xmaxglb-5,ymaxglb-50);
     selectwindow (4);
     for ej:=1 to 5 do
     begin
          gotoxy (50,12+ej);
          write ('                    ');
     end;
     Ed[1]:= 'Direccion:';
     Ed[2]:= 'Dato:';
     Ed[3]:= 'Resultado:';
     for ej:=1 to 3 do
     begin
          gotoxy (50,12+ej);
          write (Ed[ej]);
     end;
     gotoxy (61,13);
     repeat
          begin
```

```
                Ch:=readkey;
                Case Ch of
                '0': Lunid;
                '1': Ldec;
                else erd:=true;
                end;
            end;
      until erd=false;
      gotoxy (58,14);
      n:=1;
      repeat
            begin
                Ch:=readkey;
                If Ch<>#0 then DAB[n]:=Ch
                else
                    begin
                        Ch:=readkey;
                        If Ch=#13 then DAB[n]:=#10
                        else DAB[n]:=Ch;
                    end;
                write (ch);
                if (Ch<>#13) or (n<255) then n:=n+1;
                end;
      until (Ch=#13) or (n=255);
      m:=1;
      SourceAdd;
      SourceData;
      Test:=true;
      gotoxy (58,16);
      if test in [er1,er2,er3,er5] then write('Error')
      else write ('Ok');
end;

{*********************************************************}


Procedure RecibirDato;
var
   jk:integer;Rd:array [1..4] of string; test:boolean;
   f:integer;

begin
      Port[CONT]:=COMLS;
      Port[PC]:=DLS;
      Port[PB]:= PBF and $E6;
      Definewindow (6,xmaxglb div 5*3,ymaxglb div 5*2,
                    xmaxglb-5,ymaxglb-50);
      selectwindow (6);
      for jk:=1 to 5 do
      begin
           gotoxy (50,12+jk);
           write ('                    ');
      end;
      Rd[1]:='Direccion:';
      Rd[2]:='Resultado:';
```

```
      Rd[3]:='Dato:';
      for jk:=1 to 3 do
      begin
           Gotoxy(50,12+jk);
           write (Rd[jk]);
      end;
      gotoxy (61,13);
      repeat
      begin
           Ch:= readkey;
           Case Ch of
           '0':Tunid;
           '1':Tdec;
           else erd:=true;
           end;
      end;
      until erd=false;
      gotoxy (6     m:=1;
      SourceAdd;
      AcceptData;
      test:=true;
      If test in [er3,er4] then write ('Error')
      else
        begin
        gotoxy (50,16);
        definetextwindow (7,45,15,74,23,2);
        for f:=1 to n do
        begin
             write (InDAB[f]);
        end;
        end;


end;



{*********************************************************}

Procedure EnviarComando;
var
   En:array [1..4] of string;
   mj:integer;tc:boolean;
begin

      mapal;
      Definetextwindow (3,xmaxglb div 5*3,Ymaxglb div 5*2,
                  xmaxglb-5,ymaxglb-50,4);setclippingon;
      selectwindow (3);
      drawborder;
      for mj:=1 to 5 do
      begin
           gotoxy (50,12+mj);
           write ('                    ');
      end;
```

```
      En[1]:='Direccion:';
      En[2]:='Comando:';
      En[3]:='Resultado:';
      for mj:=1 to 3 do
      begin
           Gotoxy (50,12+mj);
           write (En[mj]);
      end;
      Gotoxy(61,13);

      repeat
          begin
           Ch:=readKey;
           case Ch of
           '0':Lunid;
           '1':Ldec;
           else erd:=true;
           end;end;
      until erd=false;
      gotoxy (62,14);
      Ch:=readkey;
      Case Ch of
      'r':begin
                write ('REN');
                PBF:= PBF or $20;
                m:=1;
                SourceAdd; tc:=true;gotoxy (62,15);
                If tc in [er2,er5] then write ('Error')
                else write ('Ok');
          end;
       end;
end;

{*************************************************}

procedure TabularDato;
begin
end;

procedure GraficarDato;
begin
end;


{*************************************************************}

Procedure Operaciones;
var
   op:array [1..5] of string;kl:integer;


begin

     Definewindow (4,xMaxglb div 5*3,ymaxglb div 5*2,
                   xmaxglb-5,ymaxglb-50);
     Definetextwindow (2,xMaxglb div 5 * 3,ymaxglb div 5*2,
```

```
                    xmaxglb-5,ymaxglb-50,4);
     op[1]:='1. Enviar Dato';
     op[2]:='2. Recibir Dato';
     op[3]:='3. Enviar Comando';
     op[4]:='4. Tabular Dato';
     op[5]:='5. Graficar Dato';
     selectwindow (2);selectwindow(4);
     drawborder;
     for k1:=1 to 5 do
     begin
          gotoxy (50,12+k1);
          write (op[k1]);
     end;invertwindow;
     repeat
     begin
          Ch:= readkey;
          Case Ch of
          '1': EnviarDato;
          '2': RecibirDato;
          '3': EnviarComando;
          '4': TabularDato;
          '5': GraficarDato;end;
     end;
     until (Ch='s');
     end;


{*******Inicio codigo de Programa Principal********}

begin
    inicial;
     Setgraphmode(egahi);
     repeat portada until KeyPressed;
     cleardevice;
     clearviewport;

     d1:= '* DISPT 1';
     d2:= '* DISPT 2';
     d3:= '* DISPT 3';
     d4:= '* DISPT 4';
     d5:= '* DISPT 5';
     d6:= '* DISPT 6';
     d7:= '* DISPT 7';
     d8:= '* DISPT 8';
     d9:= '* DISPT 9';
     d10:= '* DISPT 10';


     repeat
     begin
          mapal;
          Ch:=readkey;
          if Ch='o' then operaciones;
     end;
     until Ch='q';

     CloseGraph;leavegraphic;
end.
```