29
2ej

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

DISEÑO Y CONSTRUCCION DE UNA RED CON ARQUITECTURA BUS LINEAL E INTERFACES I2C APLICADA A LA MEDICION DE PARAMETROS EN UN INMUEBLE

T E S I S P R O F E S I O N A L

QUE PARA OBTENER EL TITULO DE:

INGENIERO MECANICO ELECTRICISTA

EN EL AREA ELECTRICA-ELECTRONICA

P R E S E N T A N :

JACQUES BESSOUDO KORZENNY

JOSE LUIS RODRIGUEZ LOPEZ

ASESOR: M. EN I. LAURO SANTIAGO CRUZ

MEXICO, D. F. 1996

TESIS CON
FALLA DE ORIGEN

TESIS

COMPLETA

AGRADECIMIENTOS

A la UNAM

A la Facultad de Ingeniería

Al Instituto de Ingeniería

Al Ingeniero Lauro Santiago C.

A Rubén, Ulises, Gerardo López Z. y todos

quienes colaboraron con el desarrollo de este trabajo

JACQUES BESSOUDO KORZENNY

Mamá
Por tu apoyo, dedicación y cariño
en todo momento, bajo cualquier situación.

Angélica
Por estos últimos cinco años
maravillosos buscando un aún mejor futuro.

Al tio Felipe
Por siempre apoyarme
preocupándose por mi futuro.

Rodolfo y Claudia
Por hacer el esfuerzo día
con día por lograr una mejor familia.

A las familias Bessoudo, Korzenny y Ponzanelli
Por creer en mi brindándome su apoyo y afecto.

A todos mis amigos y sobre todo a José Luís
por su paciencia y tenacidad.

JOSÉ LUIS RODRÍGUEZ LÓPEZ

A mi madre, por su incondicional apoyo

A mi hermana
(Heee, heee, heee, heeel)

A mi padre

A Cynthia Lombardo Aburto

... y un agradecimiento especial a Jacques

Índice

Introducción

El hombre contemporáneo enfrenta situaciones que solamente la tecnología de nuestros tiempos puede resolver. La integración de complejos sistemas de diversas naturalezas ha traído consigo una variedad de soluciones a problemas de toda índole. Este trabajo pretende atacar un problema actual con una de las tecnologías más modernas. El proyecto permitirá un aprovechamiento más eficiente de recursos gracias a una red de información.

Evolución de la sociedad y los sistemas

Las necesidades del hombre han crecido considerablemente en las últimas décadas. Probablemente esto se deba a su concientización de la era en que vive, así como al contacto con herramientas sumamente poderosas. Es así que hoy en día se considera común la comunicación telefónica a larga distancia, la transmisión de imágenes y sonidos a través del aire, los automóviles de combustión interna, etc. Sin embargo, todos estos inventos, innovadores cada uno en su momento, tienen una característica común: permiten al hombre aprovechar mejor sus recursos.

Los recursos indispensables en cada momento de la historia también han sufrido modificaciones. Es posible pensar que la necesidad de resguardarse del tiempo inclemente hace miles de años es equiparable a la necesidad de tener electricidad en una residencia actual. Estas modificaciones, sin importar su proceso evolutivo, han hecho de los recursos económicos una herramienta vital para el desempeño de actividades esenciales en la vida moderna. Por lo tanto, el aprovechamiento eficiente de estos recursos es una necesidad vital.

La sociedad en que vivimos pone un énfasis especial en la eficiencia. Este concepto se ha extrapolado desde su definición matemática hasta ámbitos de gran cotidianeidad, pero siempre con el afán de mejorar la calidad de vida del hombre. Sea cual sea el campo de aplicación, el procedimiento para incrementar esta eficiencia se basa en la observación del fenómeno a optimar, obteniendo así información necesaria para una posterior toma de decisiones. Aún cuando este proceso de optimización se alejara por completo del método científico, sería difícil imaginarlo sin la información preliminar.

El primer intento de optimización de un proceso, como es el aprovechamiento de recursos, comienza con la observación. Este proceso, llevado a cabo manualmente, presenta un elnnúmero de problemas que van desde la poca precisión de las observaciones hasta la naturaleza aleatoria del sistema de adquisición de datos: un hombre. Aquí surge la necesidad de un registro fiel de las mediciones, mismas que deben realizarse con precisión.

Es entonces cuando los sistemas permiten un sustancial avance sobre técnicas anteriores. Gracias a la enorme capacidad de procesamiento de información que éstos brindan, las decisiones pueden ser cada vez más acertadas.

Descripción general del problema y su solución

El presente proyecto responde a la necesidad de obtener estadísticas adecuadas de diversos parámetros en un inmueble. Esta información es indispensable para tomar decisiones correctas conducentes a un mejor aprovechamiento del inmueble y sus instalaciones.

Las instalaciones modernas suelen tener una infraestructura compleja, ya que los requerimientos de los usuarios son cada vez más específicos y se satisfacen utilizando nuevas tecnologías. El constante monitoreo de parámetros en zonas determinadas provee información indispensable para el buen funcionamiento del inmueble.

En algunos casos resulta importante registrar los cambios de temperatura que se presentan en distintas áreas en función del tiempo. Esta información permite al administrador mejorar los horarios de uso de clima artificial, representando un ahorro de energía, por lo tanto económico, y persiguiendo un fin ecológico.

Este sistema se presenta aplicado a un inmueble. Sin embargo, el modelo puede adecuarse fácilmente a numerosas situaciones. La generalidad del sistema propuesto permite aplicaciones de índoles tan diversas como control de acceso, detección de presencia, medición de temperatura e iluminación, etc. Así, este proyecto puede conformar la parte medular de un sistema de seguridad, un monitor de procesos industriales y muchos otros sistemas que requieran transmisión de información.

La solución se obtendrá con un sistema basado en una computadora personal y varios módulos autónomos como se muestra en la figura 1.1. Cada módulo estará conectado a distintos sensores distribuidos estratégicamente. Además, todos los

módulos estarán conectados a un *bus* que lleva la información a la computadora y viceversa.

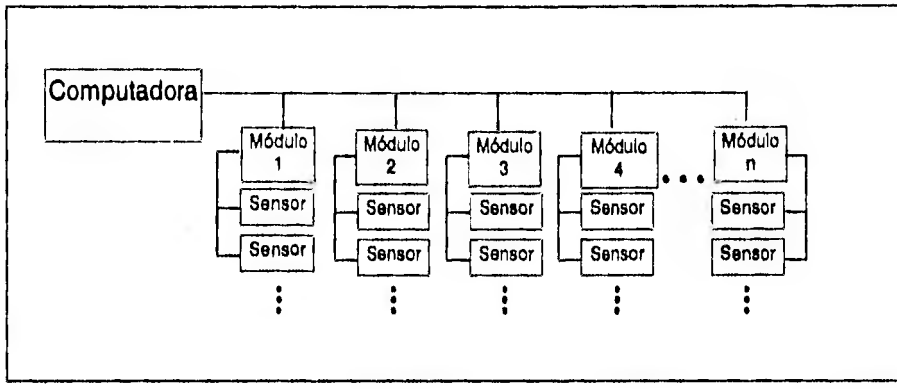


Figura 1. Diagrama general del sistema.

Los sensores permitirán la medición de parámetros ambientales dentro del inmueble. Pueden utilizarse adicionalmente lectores digitales para la implementación del control de acceso.

La información obtenida del entorno llegará al módulo correspondiente, donde podrá transmitirse inmediata o posteriormente a la computadora. Dicha información, resultado de la medición a través de sensores de temperatura, humedad o intensidad luminosa, se almacenará en archivos con propósitos estadísticos para posteriormente acceder a ella gráficamente o en un reporte. Además, como en el caso del control de acceso, la computadora alojará información necesaria para transmitir a los módulos. Tal sería el caso de las claves autorizadas de acceso.

Para la interconexión de los módulos se utilizará un *bus* lineal con interfaces I²C. Esta interfaz permite el flujo bidireccional de información en un formato serie. La norma establecida para esta interfaz brinda gran versatilidad en el diseño de sistemas, pues el protocolo permite que algunos o todos los módulos registren, reciban o envíen información a la computadora. Además, el protocolo permite la comunicación entre los módulos, permitiendo la omisión de la computadora en caso de ser necesario. Estas características hacen que la interfaz I²C sea idónea para el sistema en cuestión.

En conclusión, este trabajo presenta una solución económica, simple y versátil a un problema ancestral aprovechando las ventajas tecnológicas de nuestros tiempos. Es importante destacar que ésta es sólo una aplicación del sistema. No debe perderse de vista la potencialidad del sistema, consecuencia de su generalidad y motivo de su importancia.

Objetivos, alcances y limitaciones del proyecto

El objetivo de este trabajo es diseñar y construir una red que permita enlazar diversos módulos autónomos y concentrar la información como base de datos de una computadora personal. Posteriormente se construirá un prototipo para probar y experimentar con la operación del sistema. Sus características permiten su aplicación con cualquier computadora personal. De hecho, es posible enlazar la red con cualquier microcontrolador de 8 *bits*. Sin embargo, hay algunas limitaciones que deben observarse para el funcionamiento adecuado del equipo. A continuación se enumeran algunas de ellas.

1. El tipo de cable que se utilice para el *bus* limitará la distancia y la velocidad de transmisión del sistema.

2. El sistema pretende el monitoreo continuo de varios parámetros tanto como la concentración de información.
3. La adecuación del sistema a distintas plataformas computacionales requiere el desarrollo del *software* específico.
4. La velocidad máxima del sistema, en condiciones ideales, es de 100 kbps.

Este trabajo describe la etapa de diseño del sistema y la construcción de un prototipo. Para fines de prueba se utilizará un número reducido de módulos y sensores, pero es importante enfatizar la generalidad del esquema propuesto. Su campo de aplicación va desde el monitoreo de condiciones en una residencia hasta la adquisición de datos en procesos industriales remotos.

Estructura del trabajo

El primer capítulo, "Generalidades", plantea las necesidades que debe cubrir este proyecto. Se describirán brevemente - a nivel de diagramas de bloques - múltiples opciones que podrían satisfacer las necesidades propuestas. En el segundo capítulo, "Características del sistema", se describirá a grandes rasgos la integración del sistema a partir de la elección de algunos bloques propuestos previamente. Es aquí donde quedará definida la estructura del sistema.

El capítulo 3, "Diseño del Sistema", se enfoca a la caracterización detallada de cada uno de los subsistemas que conforman este proyecto. En este capítulo se explicarán las características que presentan los diferentes elementos del proyecto, así como las del sistema en su conjunto. Además de la discusión del diseño, se expone el proceso de construcción de un prototipo. En el capítulo cuarto, "Pruebas y resultados", se detallan las pruebas que se realizaron sobre el prototipo armado, que incluye a la estación central y un módulo autónomo de adquisición de datos.

Inmediatamente después se presentan las conclusiones y resultados del trabajo realizado, así como la bibliografía consultada para su elaboración. Finalmente se incluyen las hojas de especificaciones, diagramas electrónicos y listado de programas que forman parte vital del diseño del sistema y la construcción del prototipo.

Capítulo 1

GENERALIDADES

Este capítulo está enfocado a describir las bases teóricas en que se fundamenta la operación de cada uno de los elementos del sistema. En la etapa de transducción, estas bases serán principios físicos, mientras que en la etapa de acondicionamiento de la información las bases serán teóricas. Para la transmisión existe un gran número de modelos que pueden adecuarse al proyecto, algunos de los cuales se describen brevemente. También se discuten brevemente las características deseables del equipo de cómputo que regulará la operación del sistema.

1.1. Caracterización del problema

A pesar de la generalidad del modelo, en adelante se considerará su aplicación en un inmueble. Las adecuaciones necesarias para aplicaciones distintas son únicamente en la etapa de transducción, dependiendo de las variables que sean de importancia.

A grandes razgos, el sistema obtendrá una señal eléctrica proporcional a un parámetro físico que se desea medir. Hecho esto, la señal eléctrica sufrirá una conversión analógico-digital para su tratamiento en forma binaria. Dicho tratamiento incluye el almacenamiento de información en memoria, su transmisión a una unidad de concentración de información y su posterior interpretación y despliegue a través de una computadora personal.

Cada una de las etapas por las que pasa la información está a cargo de un subsistema distinto, con muy diferentes principios de operación que se describen a lo largo de este capítulo.

1.2. Variables a medir

Indudablemente la definición de las variables a medir debe ser el primer paso en la solución del problema. Dadas las características del sitio de aplicación, se proponen las siguientes variables:

- Presencia: Ésta será una de las variables más útiles, pues determinará los instantes en que un usuario requiere los servicios del inmueble. Esta información permitirá organizar horarios que eviten el desperdicio de energía.
- Temperatura: Esta variable permitirá modificar adecuadamente el uso de climas artificiales, la apertura de persianas automatizadas y otros dispositivos directamente relacionados con la temperatura del recinto.
- Iluminación: El medir la intensidad luminosa proporciona información que permite restringir el encendido de las luces artificiales. Además, en conjunción con el detector de presencia, podrá evitarse tener lámparas encendidas en lugares desocupados. Es importante que sólo se regulen las lámparas, y no los contactos, pudiendo afectar la operación de equipos de cómputo, entre otros.

Además de los parámetros anteriores, cuya medición forma parte del proyecto, se sugiere la medición de algunos otros parámetros como calidad del aire y ruido. Debido a la gran complejidad de su evaluación, algunos de ellos se simularán con un potenciómetro.

1.3. Sensores

Un sensor debe manifestar en forma eléctrica las variaciones que presente un parámetro determinado. Esta manifestación eléctrica puede ser una corriente o un voltaje. Idealmente se espera que la respuesta del sensor ante la variación del parámetro sea lineal pues esto permite una interpretación más directa de los resultados. Los sensores que presenten este comportamiento se denominarán en lo sucesivo *sensores analógicos*.

En algunas instancias es necesario detectar la existencia o ausencia de distintos parámetros. Para tales fines se utilizarán sensores que entregan señales digitales: unos y ceros lógicos. Éstos se denominarán en lo sucesivo *sensores digitales*.

A continuación se describen algunos de los sensores disponibles para la transducción de parámetros, así como sus principios de funcionamiento. El objetivo de esta sección no es el análisis exhaustivo de sensores de instrumentación. La construcción del prototipo involucrará la realización práctica de algunos sensores, así como la simulación de otros. Se considerarán algunos de los parámetros aquí mencionados, sin perder de vista la existencia de muchos otros que pudieran llegar a ser de utilidad según la aplicación específica.

- **TEMPERATURA**

La temperatura es uno de los parámetros más sencillos de medir. Se tiene una gran variedad de sensores con características que permiten desde una simple aproximación de la temperatura hasta su medición precisa. Para los demás parámetros analógicos se buscarán sensores económicos, o bien se simularán sus señales.

Los termistores y termopares son elementos comunes, mas requieren cantidades considerables de potencia y no son muy precisos.

Un termopar se forma si dos metales distintos, inclusive alambres, son unidos en un extremo. El punto de contacto se denomina "termounión", y a los extremos que quedan libres se les da el nombre de unión fría. Existirá una diferencia de potencial entre las terminales de la unión fría que es principalmente función de la temperatura en la termounión, como muestra la figura 1.1. Este efecto se conoce como el *efecto Seebeck*. Para evitar la formación de una nueva termounión, la unión fría debe mantenerse a la misma temperatura, y debe tratarse de que no circule una corriente importante a través del termopar.

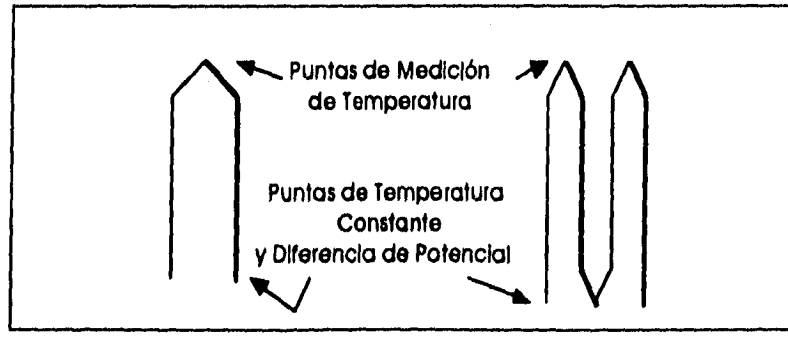


Figura 1.1. Termopar.

Los termistores se forman a partir de una unión de silicón cuya resistencia varía proporcionalmente a la temperatura. La figura 1.2 muestra un termistor típico. Los hay de coeficiente positivo y negativo, lo cual indica que la curva resistenca-temperatura que refleja su comportamiento puede tener pendiente positiva o negativa dependiendo de la especificación. Además, existen termistores con diferentes coeficientes para obtener distinta sensibilidad en el intervalo de medición deseado.

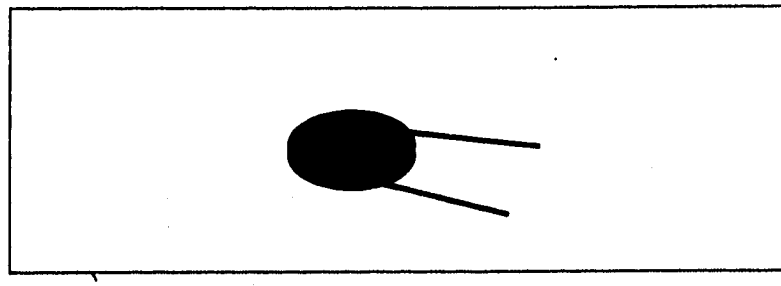


Figura 1.2. Termistor.

Por otro lado, existen los sensores ya integrados que sirven para entregar lecturas directamente en grados Farenheit o de Celsio, como lo son el LM34 y el LM35 respectivamente. Las configuraciones básicas de estos circuitos integrados son sumamente sencillas, puesto que requieren esencialmente polarización, como muestra la figura 1.3. En su configuración más completa podrían requerir una etapa de amplificación y una de conversión de voltaje a corriente. La resolución y la precisión de estos sensores es mejor que la de los termistores y termopares; llegan a tener una precisión de ±1.2ºC a temperaturas ambientales.

Existen otros sensores como el LM135, que se enfoca a la medición más precisa de temperaturas en el rango de los 0ºC a los 100ºC con una precisión de ±1ºC. Este dispositivo requiere más potencia, pero puede ser utilizado en aplicaciones específicas

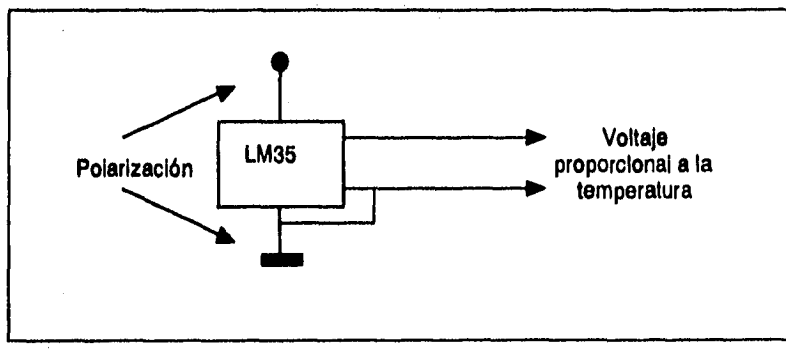


Figura 1.3. Configuración de circuitos integrados.

como el monitoreo de temperatura en alguna reacción industrial. Su configuración básica es un poco más elaborada, puesto que requiere un voltaje de referencia y un potenciómetro de calibración.

Un último método para medición de temperatura es el que aprovecha la radiación como variable de transducción. La radiación térmica es radiación electromagnética emitida por un cuerpo como resultado de su temperatura. Esta radiación tiene una longitud de onda de microondas, y puede medirse utilizando pirómetros de radiación como el mostrado en la figura 1.4. Este equipo absorbe la radiación a través de una lente y un filtro. Se coloca en la trayectoria de la radiación una fuente de luz cuyo color depende de la corriente que circula a través de ella. Cuando el color de la radiación y el color de la lámpara son iguales se tiene la lectura de corriente correspondiente.

- **HUMEDAD**

La medición de la humedad relativa requiere sensores de costo elevado. Debe utilizarse uno cuya lectura permita la detección de niveles excesivos que pudieran dañar algunos equipos. Si la aplicación lo requiriera, se podría instalar un sensor de

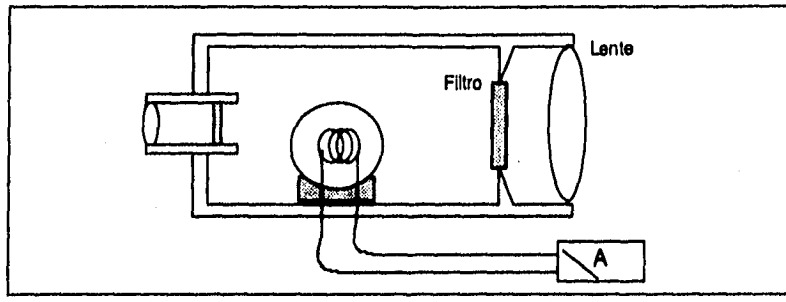


Figura 1.4. Pirómetro de radiación.

gran precisión, incrementando aun más el costo.

La humedad es la cantidad de vapor de agua que puede contener una unidad masa de aire. El aire caliente puede contener más vapor de agua que el aire frío, y cuando el aire caliente contiene un máximo de vapor de agua se dice que el aire está saturado. A partir de estas condiciones se define el término *humedad relativa* como el porcentaje de saturación del aire - vapor de agua contenido entre vapor de agua necesario para saturar la masa de aire.

Se pretende medir la humedad relativa y no la absoluta, puesto que resulta mucho más útil. La humedad relativa nos permite evaluar que tan agradable resulta estar en un determinado ambiente, así como conocer aproximadamente la concentración de cargas electrostáticas en el punto de medición.

Existen sensores simples cuyo principio de funcionamiento se basa en la la capacitancia que existe entre dos placas planas paralelas cuyo dieléctrico es el aire del ambiente. Cuando el aire presenta cambios de humedad, las características del dieléctrico cambian, por lo que cambia la capacitancia. Este sistema debe estar

encapsulado en un filtro extremadamente fino para evitar la acumulación de polvo en el sensor.

Otro tipo de sensor se fabrica con dos termómetros de resistencia de platino. Uno de ellos se mantiene seco y el otro mojado (sumergido). El cambio de resistencia en ambos termómetros indica un cambio en la humedad y en la temperatura del aire. Para asegurar que las mediciones sean correctas, se debe mantener un flujo de aire a una velocidad de 4 m/s sobre los termómetros.

Existe otro tipo de sensor de humedad comercial. Su funcionamiento se basa en el cambio de resistencia que existe en un material suceptible a la humedad. Se colocan dos conductores normales como se muestra en la figura y se recubren del material conductor absorbente. Al absorber agua el material que rodea a los conductores, su resistencia aumenta o disminuye dependiendo del material que se utilice. Su comportamiento no es lineal, y puede ser como alguno de los mostrados en la figura 1.5 .

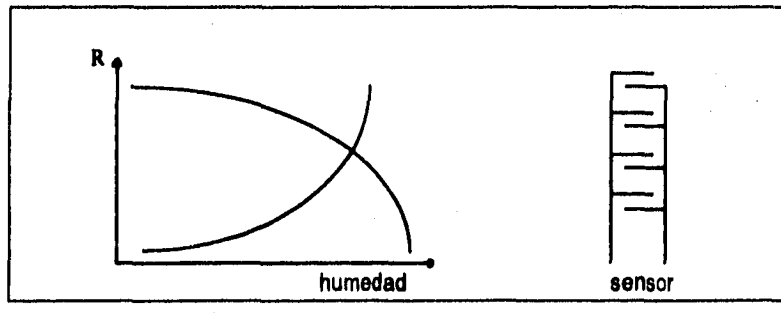


Figura 1.5. Comportamiento de un sensor de humedad no lineal.

Estos sensores tienen un costo muy elevado, por lo que en este trabajo serán simulados a falta de una alternativa más económica. Sin embargo, se puede fabricar un sensor que permita una medición aproximada. A continuación se describe la construcción y el funcionamiento del mismo.

El principio de funcionamiento es el flujo de corrientes eléctricas a través de un medio sensible a la humedad. Puede utilizarse sal común como medio, puesto que es propensa a absorber o liberar humedad para equilibrar la presión del vapor de agua en el ambiente. Gracias a la resistencia variable que presenta la sal al liberar o absorber agua, se puede determinar la humedad relativa mediante un circuito puente para medir resistencia.

Como se muestra en la figura 1.6, la construcción del sensor es bastante sencilla. El único detalle no mostrado en la ilustración es la aplicación de sal a la malla de fibra de vidrio. Se debe empapar la estructura con una solución de cloruro de litio para que la sal se impregne en la tela de fibra de vidrio. Al haber un cambio en la humedad relativa del aire, se modificará la resistencia de la malla. A pesar de lo aquí descrito, el prototipo empleará la simulación de este parámetro.

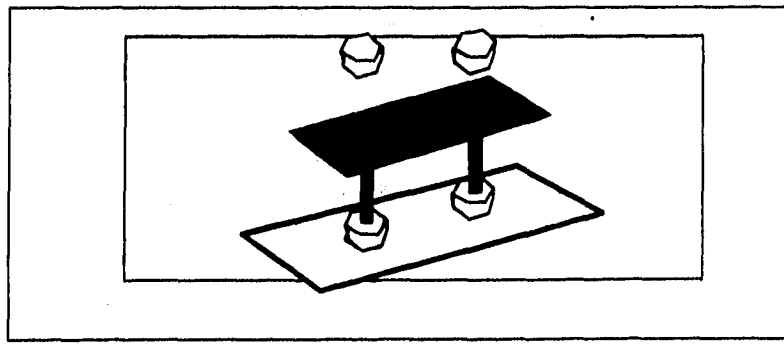


Figura 1.6. Sensor de humedad casero.

- CALIDAD DEL AIRE

La cuantificación de la calidad del aire se basa en la presencia excesiva de ciertos gases que representan algún peligro para la salud. Varios de estos gases pueden ser detectados gracias al efecto que tienen en la refracción y reflexión de ondas electromagnéticas, tales como la luz infrarroja o el ultrasonido - ya sea absorbiéndolas, desviándolas o ambas. En el caso de la detección de humo, el parámetro involucrado en la mayoría de los sensores comerciales es la temperatura del aire. La medición de "Calidad de aire" es el resultado de la ponderación de numerosas mediciones, desde gases hasta partículas suspendidas. Debido a esto, su medición se simulará con un potenciómetro.

- PRESENCIA

Para detectar presencia se puede trabajar con sensores digitales, pues no se requiere más que saber si el cuarto está siendo ocupado o no. Este tipo de señales no requieren adecuación, puesto que el sensor puede entregar niveles de voltaje compatibles con tecnología TTL.

Existe un detector de presencia que se basa en la emisión de ondas electromagnéticas de alta frecuencia. Dependiendo de qué tan rápidamente regresan las ondas, se puede determinar si hay o no personas en una habitación.

Otro sensor de frecuente utilización es el capacitivo, que se basa en medir la carga electrostática entre dos terminales al acercar una superficie al sensor. El área de detección es muy reducida, ya que los objetos deben estar extremadamente cerca para que exista un cambio en la capacitancia del sensor.

- ILUMINACIÓN

El objetivo de esta medición es determinar cuán iluminado está un espacio. Con esta lectura podrá determinarse qué tan necesaria es la iluminación artificial, por lo que se empleará como parámetro la iluminación relativa. En una habitación, al tener todas las luces apagadas y las ventanas bloqueadas, se considerará el 0 % de iluminación. Al tener el equivalente a un foco de 100W por cada 12 m^2 (con las ventanas totalmente bloqueadas) se considerará el 100% de iluminación. La escala no puede estar en función de la luz natural, pues su variación impide una evaluación objetiva.

Existen diversos métodos que pueden aplicarse a esta medición. El más simple de ellos utiliza una fotocelda como transductor primario. Ésta varía su resistencia en función de la intensidad luminosa incidente sobre su superficie. Si la fotocelda es de coeficiente negativo, la variación que presenta la resistencia será inversamente proporcional a la iluminación. Análogamente, las fotoceldas de coeficiente positivo tienen una variación de resistencia directamente proporcional a la iluminación. Ambos tipos son eumamente comunes y bastante económicos.

Para medir la iluminación también se utilizan celdas fotovoltáicas, como la mostrada en la figura 1.7. Estas convierten la intensidad luminosa en energía eléctrica, gracias a que están formadas por un semiconductor y metal, lo que constituye una juntura que manifiesta una diferencia de potencial proporcional a la exposición a la luz. Presentan una eficiencia baja, y su costo es extremadamente elevado.

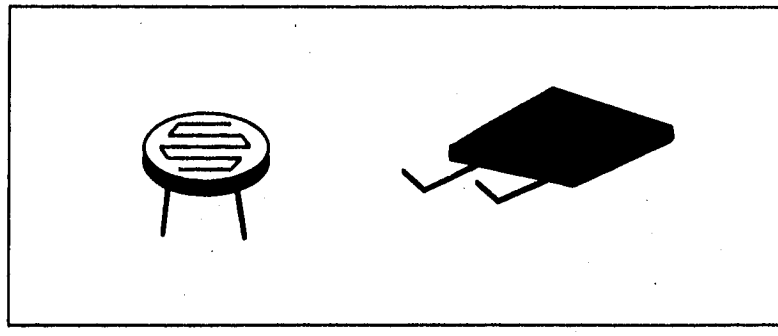


Figura 1.7. Fotocelda y pánel solar.

1.4. Conversión Analógico / Digital

Existe un gran número de métodos para convertir una señal analógica a una forma digital adecuada para su tratamiento por un sistema . Los más comunes son dos: el método de aproximación y el método *flash* o instantáneo.

Independientemente del método, todos los convertidores A/D incurren en un error de cuantización, inherente a su función. Este error es debido a que los convertidores tienen una resolución finita y generalmente pequeña.

En la figura 1.8 se grafica la curva de operación de un convertidor A/D de 2 *bits* con una resolución de 100 mV. En el eje de las ordenadas se registra el voltaje analógico de entrada al dispositivo, y en el de las abscisas los números correspondientes a algunos intervalos.

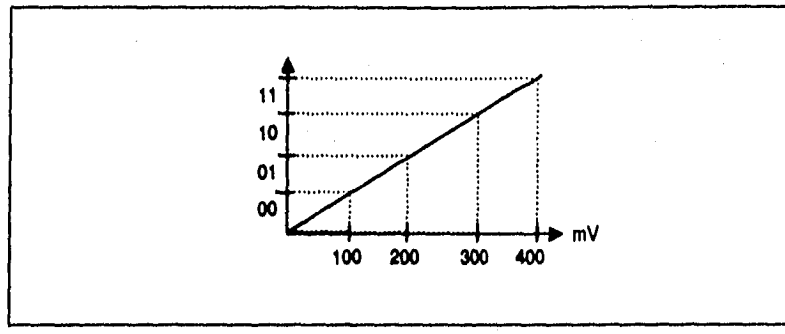


Figura 1.8. Error de cuantización.

De la figura puede observarse que el resultado "01" se obtiene para el intervalo de voltajes que va de 100 a 200 mV.

Todas las técnicas de conversión A/D se basan en un amplificador operacional configurado como comparador de malla abierta. Este arreglo permite determinar si el voltaje en una terminal es mayor o menor a una referencia, característica indispensable para la conversión.

El método de aproximación utiliza un convertidor D/A como elemento central del sistema, como se muestra en la figura 1.9. Se tiene un contador digital cuya salida pasa por el convertidor D/A. La salida analógica, que va creciendo conforme el contador opera, se compara con el voltaje que se desa digitalizar. Cuando el voltaje del contador rebasa al voltaje en estudio, el comparador deshabilita el conteo, fijando la lectura.

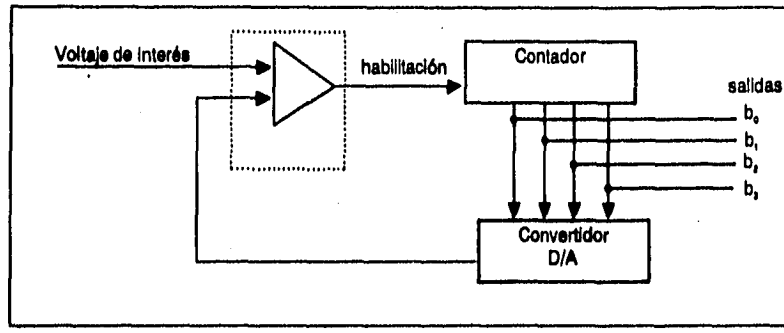


Figura 1.9. Convertidor A/D de aproximación.

En el segundo método, la conversión instantánea, el cambio de analógico a digital se verifica en paralelo, en lugar de en serie. Esto significa que en lugar de que el proceso sea el resultado de enviar una secuencia de *bits* y aproximarse al resultado, la conversión es inmediata. Este tipo de convertidor se ilustra en la figura 1.10.

La señal se alimenta a varios comparadores, todos con referencias distintas. El número de comparadores depende de la resolución deseada para la conversión. El voltaje de interés rebasará algunas de las referencias, provocando un conjunto de estados único en el conjunto de comparadores. Posteriormente esta información pasa por lógica combinacional que permite la obtención de un número binario como resultado.

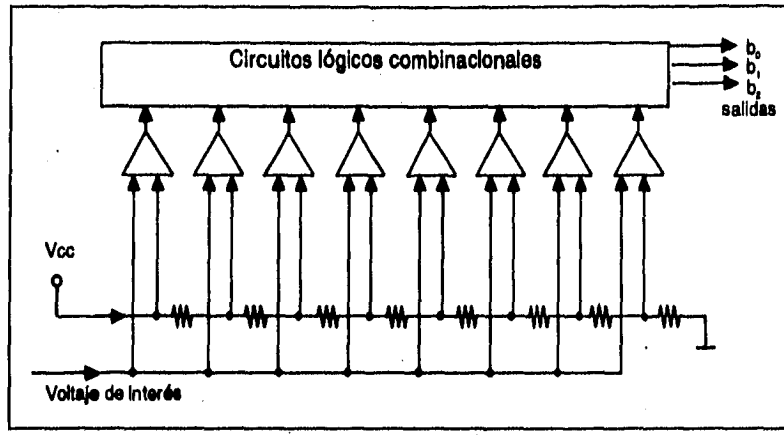


Figura 1.10. Conversión instantánea.

1.5. Microcontroladores

Los microcontroladores son circuitos integrados que concentran en un solo *chip* a una computadora completa, incluyendo varios periféricos además del microprocesador.

La parte medular de un microcontrolador es el microprocesador, que está constituido por una unidad aritmética / lógica (ALU) y otra que controla sus acciones. Además, los microcontroladores incluyen unidades de entrada y salida, así como unidades de memoria en algunos casos. La figura 1.11 muestra un microcontrolador básico representado a bloques.

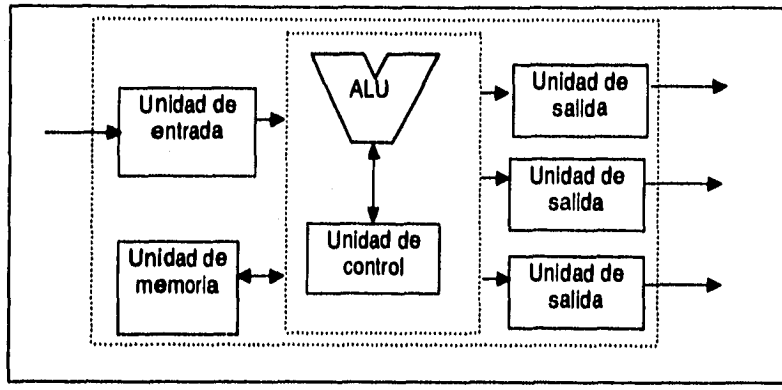


Figura 1.11. Microcontrolador

Las unidades de entrada y salida pueden ser de muy diversas naturalezas. Algunas de ellas son simplemente puertos, uni o bidireccionales, serie o paralelo, y pueden llegar a ser convertidores A/D, salidas moduladas para radiofrecuencia o hasta puertos bidireccionales adecuados a interfaces específicas. A través de estos módulos, intrínsecos todos, el microcontrolador es capaz de establecer comunicación con otros equipos, intercambiar información y procesarla, entregando resultados a través de los mismos u otros dispositivos.

Los microcontroladores forman la parte central de muchos equipos hoy en día debido a su versatilidad, simplicidad y reducido costo. Frecuentemente son subutilizados, pues la mayoría de estos cuentan con muchas más facultades de las que son necesarias para una sola aplicación. Por este motivo reciben el nombre de microcontroladores de propósito general.

También existen microcontroladores de propósito específico, y recientemente se han desarrollado *chips* que tienen muy pocas unidades de entrada, salida y memoria y por lo tanto tienen un costo mínimo. Esto permite la reducción del tamaño de sistemas

que realizan operaciones lógicas relativamente complejas y cuya implementación con lógica discreta sería extremadamente voluminosa.

Una de las características más deseables en un microcontrolador es la capacidad de expansión. Esto permite que el microprocesador haga uso de recursos externos al *chip*, incrementando su campo de aplicación enormemente.

Los principales fabricantes de microcontroladores de propósito general son Intel, Philips, Texas Instruments, National y Motorola. La mayoría de estas compañías desarrollaron sus microcontroladores como respuesta a necesidades específicas que presentaron diseños de mayor alcance.

1.6. Sistema de adquisición de datos

Es importante poder mantener un monitoreo de las variables a medir. Sin embargo, las variaciones que presentan los parámetros en estudio son relativamente lentas. Es entonces razonable considerar que un muestreo relativamente frecuente mostrará con precisión el comportamiento de las variables.

Se considerará un intervalo de tiempo entre mediciones de aproximadamente 15 segundos, mismo que permitirá que cada lectura sea registrada en la computadora. El monitoreo de cada uno de los módulos se logra de la misma manera, pues un intervalo de 15 segundos como máximo brinda una lectura bastante certera de las magnitudes de los distintos parámetros presentes en los puntos de muestreo.

Las mediciones se deberán concentrar en un punto común a los diferentes módulos colocados estratégicamente en el interior de un inmueble. Estos módulos coordinarán la transducción y digitalización de los datos adquiridos. Además, deben tener capacidad de memoria, de tal manera que operen autónomamente y eviten la

posible pérdida de información debido a fallas en el sistema. Para su conexión con el punto de concentración se debe instalar cableado que lleve la información al punto donde ha de ser analizada y revisada. Como se verá más adelante, existen varios criterios para seleccionar el tipo de cable y el modo de transmisión. Los parámetros fundamentales que determinarán las características del equipo son la velocidad, disponibilidad y costo de cada uno de los medios de transmisión.

La información concentrada deberá ser interpretada de manera clara y eficiente. Esto permitirá que el personal pueda tomar decisiones oportunas que mejoren las condiciones del inmueble. Para tal fin, la interfaz con el usuario debe ser tan amigable como sea posible, permitiendo que cualquier persona con conocimientos rudimentarios de computación opere el sistema e interprete adecuadamente la información.

A continuación se describen los parámetros de la arquitectura del sistema de adquisición de datos, como son las características eléctricas, físicas y lógicas.

1.6.1. FORMATO DE TRANSMISIÓN

Las opciones básicas que se deben considerar al proponer una red de información es el tipo de transmisión se desea lograr - serie o paralelo. Cada una de éstas presenta ventajas y desventajas, y algunas resultan determinantes en la elección de un formato de transmisión.

La comunicación en paralelo presenta principalmente la ventaja de poder alcanzar velocidades sumamente elevadas. Esto es gracias a que, como su nombre lo indica, los *bits* de un *byte* viajan al mismo tiempo a lo largo del medio de transmisón - en forma paralela. Desafortunadamente no se puede enviar información a distancias lo suficientemente grandes como para cubrir los amplios espacios de un inmueble. Esto se debe a la utilización de voltajes pequeños a potencias bajas en el bus de

comunicaciones, independientemente del *hardware*. Otra desventaja que presenta la comunicación en paralelo es el espacio físico del cable, así como su costo. Además, este formato de transmisión aumenta considerablemente la probabilidad de ruido en la información debido a la proximidad de los conductores, como se ve en la figura 1.12.

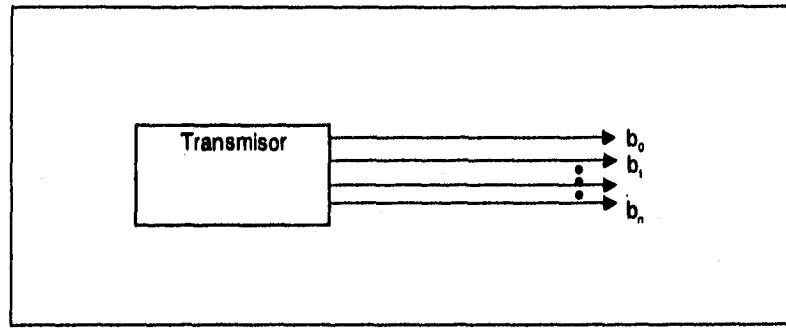


Figura 1.12. Transmisión en paralelo.

Al utilizar la comunicación serie, se logran velocidades de transmisión menores, pero la distancia y versatilidad de las interfaces crecen considerablemente, como se discute posteriormente. El esquema general se muestra en la figura 1.13.

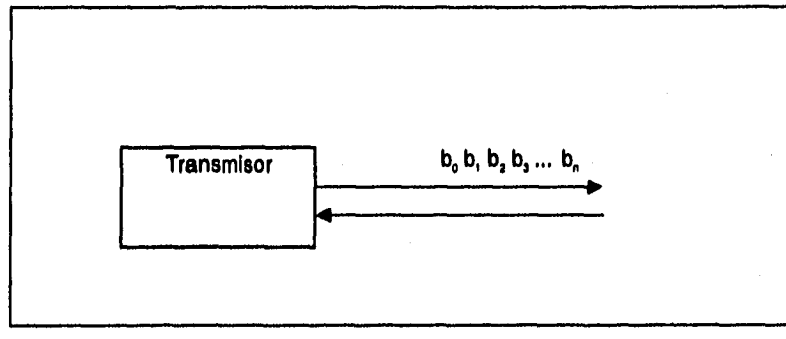


Figura 1.13. Transmisión en serie.

1.6.2. TOPOLOGÍAS

Al hablar de la arquitectura de un sistema de comunicaciones, es necesario discutir tanto su topología física como su configuración lógica. A continuación se evalúan opciones para ambas características.

- TOPOLOGÍA FÍSICA

  Existe una gran variedad de métodos en que se organizan módulos inteligentes para comunicarse, considerando las necesidades actuales y futuras del sistema. Las configuraciones más comunes que satisfacen estos requisitos son las siguientes:

  - Punto a punto

    Una red punto a punto es la configuración física más simple, pues tiene una computadora, una línea de comunicaciones - que puede ser directa o a través de un complejo sistema de conmutación - y un módulo en el otro extremo de la línea. Si bien esta configuración no es útil para nuestros propósitos debido a que únicamente comunica dos elementos, ésta es el punto de partida para configuraciones más complejas que se describen a continuación.

  - Estrella

    La estrella es una topología centralizada en la que las comunicacciones se establecen con un solo elemento, y cada módulo está conectado directamente a la computadora a través de un canal de comunicación independiente, como se muestra en la figura 1.14. Esta configuración es notablemente más versátil que la anterior, pues permite la interconexión de un número indeterminado de módulos. Sin embargo, existen consideraciones prácticas que hacen de la estrella una solución aparatosa. El problema principal consiste en alojar el gran número de cables que interconectan los

módulos. Además, la computadora recibirá un canal de comunicación por módulo, lo cual significa que debe contar con un número de puertos igual al número de módulos. Adicionalmente, la longitud total de los conductores sería mucho más elevada que si se optara por alguna de las configuraciones descritas a continuación. La desventaja más grande de esta configuración estriba en que si el nodo central fuese deshabilitado, el resto de los sistemas no puede comunicarse entre sí.

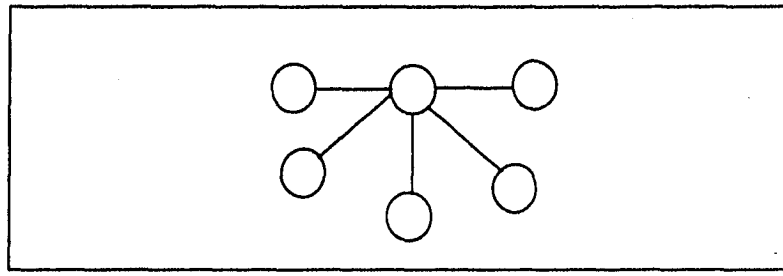


Figura 1.14. Configuración en estrella.

- Anillo

  Esta topología considera a los módulos como partes constituyentes de un *anillo* donde cada elemento está conectado a dos elementos adyacentes, como se muestra en la figura 1.15. En esta configuración no existe un elemento central, por lo que la computadora sería uno más de los nodos de la red.

  La principal ventaja del anillo estriba en las velocidades que pueden desarrollarse, su tolerancia a fallas en algunas partes del sistema y la simplicidad de la topología lógica, pues la mayoría de estas redes utilizan *token passing*, descrito posteriormente.

Si alguno de los nodos queda deshabilitado, las comunicaciones pueden viajar en el sentido contrario al convencional, por lo que aún pueden establecerse entre cualesquiera dos elementos.

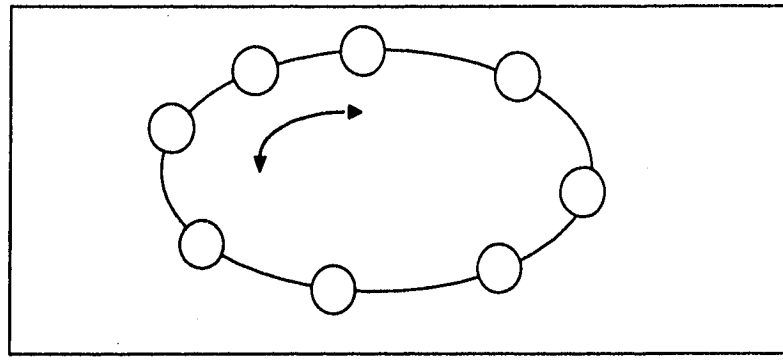


Figura 1.15. Configuración en anillo.

- Jerárquica

  Una red jerárquica representa una red totalmente distribuida en que algunos nodos dependen de otros, quienes dependen a su vez de otros, etc. En una red de cómputo donde cada uno de los sistemas puede tener capacidades específicas de procesamiento, esta topología permite distribuir las tareas de forma óptima. Sin embargo, dadas las características de los módulos en este proyecto, así como la naturaleza de centralización deseada, esta opción no presenta la flexibilidad y simplicidad de otras. Su representación esquemática se muestra en la figura 1.16.

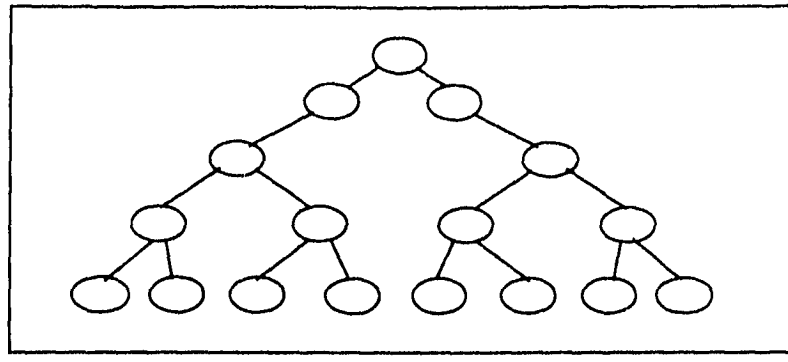


Figura 1.16. Configuración jerárquica.

- *Bus* lineal

Las topologías lineales, o de *bus*, se consideran las más flexibles. Generalmente se configuran de modo que cada uno de los nodos se alimenta a través de una rama del canal principal de comunicaciones, como se ve en la figura 1.17. Cuando la información viaja a través del medio, sea este cable coaxial, fibra óptica o par trenzado, todos los nodos reciben simultáneamente la información, por lo que esta topología también se conoce como "difusión" (*broadcast*) o "*daisy chain*", y generalmente se asocia con las redes Ethernet. Debido a las características eléctricas del bus, si el cable sufre una ruptura en algún punto, gran parte de la red quedará sin servicio.

Además de presentar la manera más simple de conexión entre elementos, la topología de bus permite la adición y remoción de módulos a placer sin alterar en absoluto la operación del sistema. Adicionalmente, las consideraciones lógicas que conllevan estas modificaciones son mínimas a comparación de una estrella, por ejemplo.

De lo anterior se desprende que el mal funcionamiento de uno de los módulos no afectará en absoluto la comunicación entre el resto de los elementos. Sin embargo, una falla en el *bus* ocasionará que solo algunos módulos permanezcan comunicados.

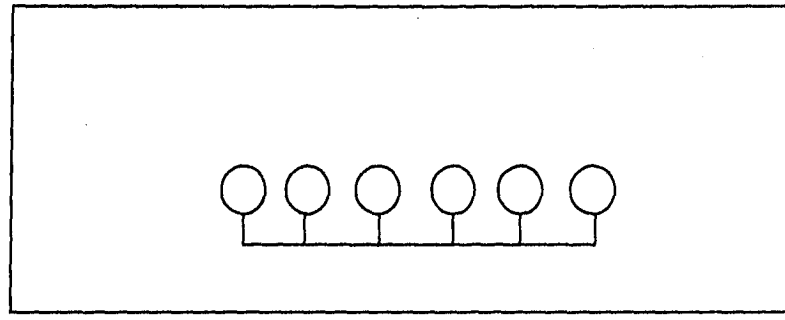


Figura 1.17. Configuración lineal o *bus*.

- **TOPOLOGÍA LÓGICA**

Sería lógico suponer que la forma en que están conectados los módulos determina inequívocamente la manera en que viaja la información. Sin embargo, esto no es necesariamente cierto. Existen dos formas generales en que se transmite la información: difundiéndola simultáneamente a todos los nodos o relevando los mensajes de nodo a nodo. A continuación se describen algunos de los esquemas de transmisión más comunes, así como los protocolos que emplean.

- *Token passing*

Esta técnica define la secuencia lógica con que la información viaja a través de la red, así como el protocolo de control para tener acceso al canal de comunicaciones. En una red de "turnos", siempre circula un mensaje que se denomina la "ficha libre" (*free token*) cuando el canal de comunicaciones

está libre. Si uno de los nodos desea transmitir un mensaje, debe esperar a recibir la "ficha libre", y posteriormente transmitir el mensaje junto con una "ficha ocupada" (*busy token*) . Mientras se transmite este mensaje, no hay "ficha libre" en el aistema, por lo que ningún otro elemento puede utilizar el canal de comunicaciones. Cuando el destinatario recibe el mensaje, el transmisor genera una nueva "ficha libre", permitiendo a otros nodos transmitir. Como consecuencia de este procedimiento, no existe la transmisión simultánea de varios mensajes, evitando colisiones.Esta topología lógica puede utilizarse en cualquier topología física (*Token Bus*), mas se emplea generalmente en sistemas configurados en estrella (*Token Ring*).

- CSMA/CD (*Carrier Sense Multiple Access / Collision Detection)*

  Este protocolo utiliza una topología de difusión, empleando datagramas para transmitir mensajes largos. La técnica de CSMA / CD (Acceso Múltiple por Detección de Portadora / Detección de Colisión) garantiza que nunca se envíen varios mensajes, subdivididos en datagramas, simultáneamente. Adicionalmente, en caso de ocurrir esta situación, el protocolo define cuál de los dos datagramas prevalecerá.

  Los datagramas son la unidad de tranamisión en este protocolo. Están conformados por una porción de mensaje y la dirección a donde viaja la información, como se muestra en la figura 1.18. Esta técnica de subdivisión es común en redes de conmutación por paquetes, donde se pueden comprobar sus ventajas de velocidad. En el caso de un bus, los datagramas establecen la comunicación entre dos nodos antes de transmitir el mensaje deseado .

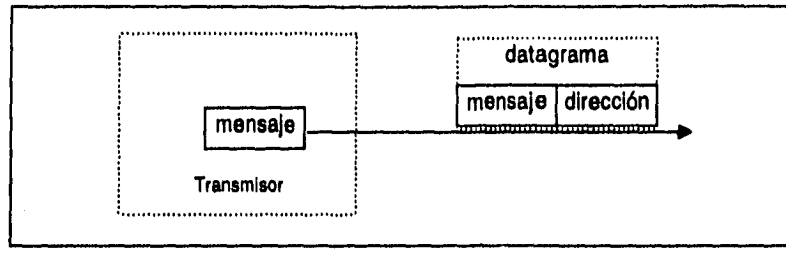


Figura 1.16.Transmisión por datagramas.

El concepto de datagrama se basa en el mejor esfuerzo de los nodos para hacer llegar la información a su destino. Sin embargo, no tiene ningún tipo de corrección ni detección de errores, cuestiones que pueden diseñarse con protocolos de nivel más elevado.

Antes de transmitir un datagrama, el protocolo CSMA/CD monitorea el bus antes de transmitir. Si debe transmitir un mensaje, espera a que ningún otro nodo esté transmitiendo para iniciar el proceso. Dos nodos podrían iniciar simultáneamente la transmisión de información, situación que también contempla este protocolo. Debido a las características eléctricas del bus, los nodos pueden detectar discrepancias entre lo que transmiten y lo que reciben. Al manifestarse una de estas diferencias, el nodo en conflicto suspende la transmisión y la realiza posteriormente.

1.6.3. INTERFACES

Los estándares mencionados a continuación permiten la transmisión de información entre dos dispositivos con un formato ajeno a ambos. Esto se realiza con el fin de conseguir mayores distancias, mejores tasas de error y mayores velocidades.

Como primera opción se consideró la interfaz RS232C, que se ha convertido en un estándar de las interfaces para computadoras personales desde hace ya muchos años. Ésta presenta la desventaja de que las distancias alcanzadas son demasiado cortas para fines de una red.

Texas Instruments desarrolló una interfaz alternativa: RS485. Su propósito es hacer más versátil la comunicación tipo RS232C existente. La interfaz RS485 presenta la importante ventaja de permitir cableados de hasta 1,200 metros de longitud. Estas interfaces permiten conectar hasta 40 usuarios a un bus simultáneamente. Los circuitos integrados que realizan la conversión eléctrica de la información son de bajo costo y tienen buena disponibilidad.

Otra opción es la basada en la tecnología I²C, desarrollada por Philips conjuntamente con Signetics. Ésta tuvo originalmente el propósito de comunicar módulos dentro de equipos electrodomésticos fácil y confiablemente.

La interfaz I²C, en conjunción con acopiadores de línea, presenta buen desempeño en distancias aproximadamente de 400 metros con un número adecuado de terminales acopladas al bus. No es posible predeterminar el número de usuarios debido a que las capacitancias de los dispositivos I²C afectan directamente la cantidad de terminales que soporta el sistema. Esta interfaz tiene como principal ventaja el contar con un protocolo que todos los circuitos integrados compatibles decodifican automáticamente. Gracias a esta característica, la transferencia de información entre dispositivos I²C es transparente, evitando al diseñador los problemas inherentes al protocolo.

Para una solución más versátil, se podrían combinar las características de varios sistemas. La salida I²C, ya con el protocolo deseado y con las características adecuadas añadiendo la longitud alcanzada por los circuitos RS485.

1.6.4. CABLEADO

Debido a que toda la transmisión de información en una red se realiza en banda base, es decir, sin modulación, las señales que viajan por el canal de comunicaciones son ondas cuadradas. Lamentablemente, el contenido armónico de este tipo de señal hace que la transmisión sea deteriorada por interferencia y radiación. Estos problemas se manifiestan en distintas proporciones en varios medios, como se describe a continuación.

- Cable coaxial

  El cable coaxial consta de un alambre central de cobre, ya sea macizo o de múltiples hilos, rodeado por un blindaje metálico. Su conformado se muestra en la figura 1.19. El blindaje conductor evita que el alambre reciba señales electromagnéticas radiadas por otros sistemas, haciendo las veces de un plano de tierra. Para separar el conductor del blindaje se utiliza un dieléctrico flexible, y en el exterior se tiene un recubrimiento ahulado que protege el conjunto.

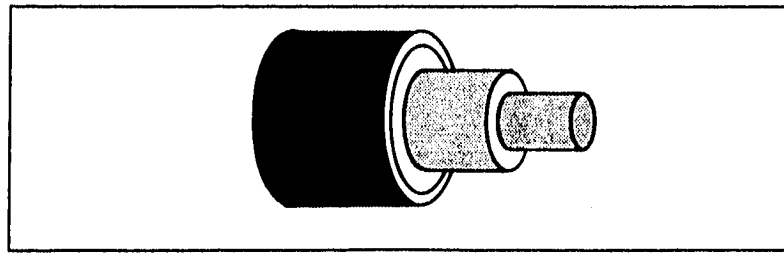


Figura 1.19. Cable coaxial.

- Par trenzado

Este tipo de cable consta de un par de alambres aislados entre sí y torcidos dentro de un forro dieléctrico. El *trenzado* minimiza la susceptibilidad de la información a la absorción de energía radiada, mas no es tan efectiva como el blindaje. El par trenzado puede soportar interferencia leve, mas cualquier motor es capaz de radiar suficiente energía para alterar los datos. Sin embargo, este tipo de cable es en extremo popular por su precio accesible (menos de la mitad del precio de un cable coaxial).

Existen varias categorías de par trenzado de acuerdo con sus características para transmisión de información digital. Dichas características se muestran en la tabla 1.1.

CABLE	Ancho de banda	Velocidad máxima	Comentarios
Categoría 3	16 MHz	más de 10 Mbps	Características mínimas aceptables
Categoría 4	20 MHz	más de 16 Mbps	Buen aislamiento
Categoría 5	100 MHz	más de 100 Mbps	Máximo desempeño

Tabla 1.1. Características del par trenzado.

- Par trenzado blindado

Para contrarrestar la gran desventaja que presenta el par trenzado, surge esta alternativa donde los pares trenzados están alojados dentro de un blindaje metálico diseñado específicamente para absorción de ruido electromagnético. A pesar de esto, no es un tipo de cable muy popular debido a su rigidez. Aún así, IBM lo utiliza en todas sus instalaciones de redes *Token Ring*. Su constitución es la mostrada en la figura 1.20.

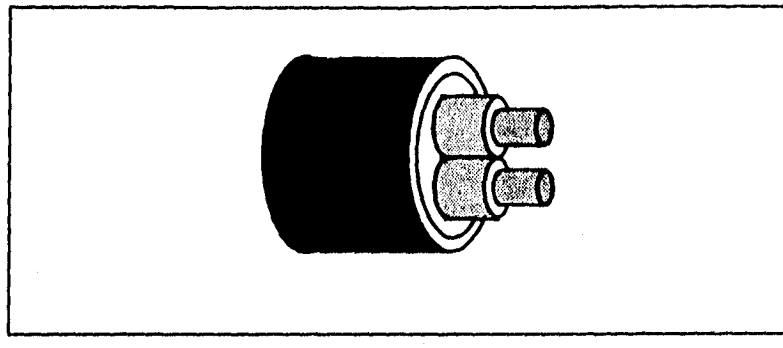


Figura 1.20. *Par trenzado blindado.*

Existen principalmente dos tipos de par trenzado blindado (STP). Ambos tienen excelente inmunidad al ruido y, por lo tanto, radiación mínima. El cable convencional tiene un ancho de banda de 20 MHz, mientras que el tipo "A" (STP-A) tiene desde 300 hasta 600 MHz , dependiendo del fabricante. Con este último tipo de cable se realizan conexiones para multimedia a velocidades de más de 100 Mbps.

- Fibra óptica

El objetivo fundamental del análisis de cableado es establecer qué esquema ofrece la mayor inmunidad al ruido. Al utilizar hilos de fibra óptica, puede transmitirse una cantidad de información varias veces mayor a la que puede transportar un par trenzado. Actualmente, las fibras ópticas se utilizan para comunicaciones de sistemas integrados con velocidades y anchos de banda inimaginables.

Dependiendo de la configuración y tipo de fibra (monomodo, multimodo, índice graduado, etc.) , una fibra óptica puede transportar desde uno hasta

miles de canales simultáneamente. La tendencia actual utiliza fibras monomodo por su capacidad de tener varios canales simultáneos a velocidades de gigabits por segundo. Todos los canales viajan por el mismo camino, pero modulados a distinta frecuencia.

Algunas de las características de las fibras más comunes se consignan en la tabla 1.2. Todas las figuras se consideran para una longitud de onda de 1300 nm.

Fibra	Atenuación (dB / km)	Ancho de banda (MHz)	Apertura numérica
Monomodo	1	inagotable	0.1
50 / 125	2	400	0.2
62.5 / 125	1.5	500	0.275
100 / 140	4	200	0.29

Tabla 1.2. Características de la fibra óptica.

La denominación de la fibra es la razón del diámetro de la médula al diámetro de la primera cubierta. (*core / clad ratio*). La apertura numérica (AN) representa la cantidad de modos que pueden seguirse en esa fibra. Como es de esperarse, la fibra monomodo tiene una AN muy pequeña.

Puesto que este proyecto no aprovecha las ventajas de ancho de banda y velocidad, el atractivo de la fibra óptica se reduce a una inmunidad total a interferencia electromagnética, pues los cilindros de vidrio no funcionan como antenas.

1.6.5. Computadora personal

Éste es el último de los elementos involucrados en la solución del problema. Una vez que la información ha sido digitalizada y transmitida, es aquí donde se realizará el almacenaje a largo plazo, así como la interpretación gráfica de los datos.

Debido al gran dominio de las computadoras IBM-compatibles en el mercado de las computadoras personales, la realización del prototipo está enfocada a esta plataforma. Esto tiene relevancia en el desarrollo del *software* y en el tipo de conector requerido por la interfaz, mas la operación del sistema es totalmente independiente del equipo que se emplee como elemento de concentración de información.

1.7. *Software*

El tratamiento que se le da a la información una vez que ha llegado a la computadora, elemento de concentración de datos, está regulado por el *software* que se ejecuta en tal elemento. Las bases sobre las que opera este conjunto de instrucciones se describen a continuación.

1.7.1. Sistemas operativos y ambientes

Después de viajar por el *bus* I^2C, la información llegará a la computadora para ser almacenada e interpretada. Dicha computadora debe ser capaz de manejar los datos adecuadamente, para lo cual se requieren programas de diseño específico. Estos diseños deben considerar las características del equipo de cómputo utilizado, y en particular del sistema operativo empleado.

Las computadoras utilizan algunas instrucciones llamadas *sistema operativo* para controlar y administrar los recursos con que cuentan. A través de este sistema operativo se realiza el intercambio de información entre el mundo exterior y la computadora, pues es él quien regula las acciones del teclado, monitor y cualquier otro dispositivo de entrada o salida. Además, el sistema operativo asigna la memoria requerida a distintos programas, posee rutinas para la administración de unidades de disco, tiene módulos de diagnóstico, etc.

Suponiendo que la plataforma computacional más común es la compatible con IBM-PC, puede considerarse como generalizado el uso del sistema operativo MS-DOS, desarrollado por Microsoft. En este sistema operativo el usuario tiene acceso a una línea de comandos desde la cual puede ordenar, a través de palabras escritas, acciones diversas.

A pesar de su enorme difusión, este sistema operativo tiene entre sus adeptos mayoritariamente a personas experimentadas. El uso de un sistema tan "árido" disuade al neófito fácilmente, pues resulta engorroso y complicado recordar los comandos, su sintaxis y aplicación. A raíz de esta situación, y para captar un mayor sector de la población, Microsoft diseñó el ambiente Windows.

Windows es una Interfaz Gráfica de Usuario (GUI) diseñada para que a través de dibujos representativos cualquier persona pueda acceder a los recursos ofrecidos por MS-DOS sin ninguna complicación. Las operaciones están representadas por íconos que, al seleccionarse, desempeñan muchas de las tareas más comunes: desde copiar un disco hasta iniciar una aplicación de procesamiento de texto.

Debido a su carácter "amigable", Windows se convirtió en la norma de operación en todas las computadoras IBM-compatibles con arquitectura AT. Esto propició que los grandes desarrollos de *software* se enfocaran a Windows, empezando por Microsoft

mismo. Actualmente Windows ha evolucionado a ser más que un ambiente, sino que es un sistema operativo de 32 *bits* llamado Windows 95.

La mayor parte de las computadoras personales nuevas incluyen el sistema operativo Windows 95, por lo que las aplicaciones para MS-DOS son cada vez más escasas. Para ampliar aún más las capacidades de Windows 95, éste es capaz de ejecutar cualquier programa para MS-DOS y Windows en sus versiones anteriores.

La utilización de una computadora personal IBM-compatible para el desarrollo de este proyecto es producto de su gran popularidad en nuestro país. Existen otros lugares donde computadoras como Macintosh son considerablemente más populares, en cuyo caso el desarrollo del prototipo se realizaría en torno a dicha plataforma. Como consecuencia de esto, los programas elaborados para la operación del sistema funcionarán bajo Windows 3.1 ó superior, con lo que simultáneamente se garantiza la presencia de MS-DOS.

Windows presenta herramientas de diseño de estupenda presentación, así como gran complejidad. Por su presentación, es un ambiente idóneo para el desarrollo de la interfaz con el usuario. Sin embargo, las operaciones que son transparentes para el usuario pueden basarse en MS-DOS para tener mayor velocidad y simplicidad de elaboración.

1.7.2. Lenguajes de programación

Es necesario distinguir dos etapas de tratamiento de la información para un mejor análisis de las ventajas y desventajas presentadas por un lenguaje específico de programación. Debido a que estas dos etapas tienen fines totalmente distintos, las características favorables para cada una son totalmente ajenas a las de la otra.

La primera de estas etapas comprende la adquisición de los datos, lograda a través de controlar el *bus*, coordinar las comunicaciones, brindar las escalas adecuadas a los parámetros medidos y almacenar los datos en archivos. La segunda etapa se enfoca a la interpretación de los datos para el usuario final. Al presentar la información amigablemente, es más fácil tomar decisiones fundamentadas.

Existe una tercera etapa, la programación del módulo concentrador, que necesariamente fue elaborada en lenguaje ensamblador. Es por este motivo que no se contempla en la siguiente discusión.

- ADQUISICIÓN DE DATOS

  Este módulo utilizará un lenguaje de alto nivel de programación para controlar la interacción de la computadora con el puerto paralelo, recibiendo y transmitiendo los datos requeridos. Para su elaboración puede emplearse cualquiera de los siguientes lenguajes:

  - Quick Basic

    BASIC (*Beginner's All-purpose Symbolic Instruction Code*) ha sido el lenguaje más utilizado para programación personal. La mayoría de los usuarios de computadoras personales han aprendido a programar en BASIC debido a su facilidad de aprendizaje y utilización. Su desarrollo pretendió llevar la computadora a un ambiente casero, y no solamente a los programadores expertos.

    Quick Basic (QB) es un gran avance sobre las versiones originales de BASIC, que fue diseñado para principiantes en el uso de lenguajes de programación, pues ofrece un lenguaje estructurado y poderoso. Algunas de las ventajas de QB son el poder editar, compilar, ejecutar y depurar

programas en un solo ambiente de trabajo. Todas las herramientas del lenguaje están contenidas en un solo paquete integral.

Algunas de las desventajas de este lenguaje incluyen su baja velocidad de ejecución. Esto se debe a que la mayoría de las versiones originales se escribieron como intérpretes, mas ahora se escriben como compiladores. En términos más tangibles, anteriormente cada una de las instrucciones se traducía a lenguaje de máquina individualmente, mientras que ahora se traduce o compila el programa completo.

- Turbo Pascal

Turbo Pascal es uno de los lenguajes más utilizados a partir de 1980, pues ofrecía dos características muy llamativas: un ambiente de compilación rápido y eficiente, y extendió el antiguo lenguaje Pascal para poder aprovechar tendencias modernas en la evolución del *hardware*. Esto significa que Turbo Pascal puede ejecutar todas las operaciones de Pascal y muchas otras, como control de DMA y puertos, que no existían previamente.

Turbo Pascal provee todo el *software* requerido para el desarrollo de programas: el editor, un compilador, depurador, documentación en línea y acceso eficiente a los archivos de la computadora. La interfaz con el usuario se basa en menús con diálogos, botones y otras características gráficas que lo hacen resaltar al compararse con MS-DOS, por ejemplo. Por si esto fuese poco, las versiones actuales incluyen una gran cantidad de demostraciones que facilitan el aprendizaje grandemente.

Una posible desventaja de este lenguaje es que el tiempo de aprendizaje es notablemente mayor al requerido para QB, y su estructura es bastante diferente. Sin embargo, son ligeras desventajas que pueden superarse para la obtención de un mejor producto final.

- C / C++

Este lenguaje fue creado en los laboratorios Bell en 1972. El lenguaje no fue diseñado por pasatiempo, sino con el fin de desarrollar el sistema operativo UNIX. Debido a su gran flexibilidad y poder, este lenguaje se difundió rápidamente para escribir todo tipo de programas en muchos lugares distintos. Esto propició el surgimiento de varias versiones de C, mismas que no son totalmente compatibles.

Las características elementales que hacen de C un estupendo lenguaje de programación son varias. En primer lugar, C es un lenguaje sumamente poderoso y flexible, pues no tiene barreras tangibles. Sus aplicaciones van desde sistemas operativos y compiladores hasta procesadores de texto o control de conmutación telefónica.

Otra ventaja de C es su gran popularidad, pues el intercambio de ideas e información se facilita al contar con una enorme comunidad de adeptos.

Por último, C es un lenguaje portátil, lo que significa que un programa escrito para una IBM-PC puede compilarse y ejecutarse en una VAX, por ejemplo. Esto es posible, entre otras cosas, por la regulación del Instituto Nacional Americano de Normas (ANSI), quiene marcaron las pautas para el desarrollo de ANSI-C.

La única posible desventaja de este lenguaje es que, debido a su robustez, es un lenguaje relativamente árido y difícil de aprender. Cuenta con grandes similitudes con Turbo Pascal, lo cual hace aún más difícil detectar las diferencias.

- INTERPRETACIÓN DE DATOS

Debido a las características que se desan de la interfaz con el usuario, puede utilizarse un lenguaje constructor de soluciones. Este tipo de lenguaje está orientado a soluciones que relacionen al usuario final con el problema de fondo a través de una interfaz fácil de manejar. Las herramientas más comúnmente utilizadas para este tipo de *software* son:

- Visual Basic

Este lenguaje está disponible para el ambiente Windows, y es la herramienta más rápida para crear aplicaciones generales. El tiempo de aprendizaje es breve, requiere pocos conocimientos de programación y brinda una gran facilidad para una rápida integración y reuso de componentes.

Visual Basic tiene gran flexibilidad para llevar a cabo llamadas a archivos *.DLL y agregar una enorme variedad de controles al gusto del programador y del usuario. Constituye un camino fácil para crear interfaces de usuario utilizando componentes preconstruidas, por lo que se tiene un lenguaje flexible que cuenta con un editor verificador de sintaxis y herramientas de perfeccionamiento.

Sus principales ventajas son:

- Controles visuales preconstruidos por terceros (controles Visual Basic eXtention (VBX))
- Permite ensamblar fácil y rápidamente una interfaz de usuario con componentes prefabricadas
- Ofrece gran capacidad y velocidad
- La interfaz de programación es muy amigable
- La ayuda en línea es sumamente completa
- Detecta errores en el momento de edición del programa
- El tiempo requerido para aprenderlo es mínimo

Sus principales desventajas son:

- El lenguaje no es demasiado flexible, pues necesita la ayuda de lenguajes más robustos, como Pascal, para manipular memorias, acceder a los puertos y a los registros del microcontrolador.
- Requiere Windows

- Visual C++

Este lenguaje constituye una herramienta flexible y poderosa para generar soluciones, además de ser rápida para programadores que utilizan C y C++.
Sus principales ventajas son:

- Incluye librerías preconstruidas
- Tiene controles visuales
- Maneja controles VBX (Visual Basic eXtension)
- Maneja multitarea
- Maneja variables locales y globales
- Total libertad de manejo de memoria y accesos a sistema operativo

Sus desventajas principales son:

- El aprendizaje de este lenguaje puede ser lento

- Otras posibilidades incluyen lenguajes de bases de datos, como podría ser FoxPro o Access. Sin embargo, la naturaleza del *software* deseado se obtiene más adecuadamente con un lenguaje visual.

A lo largo de este capítulo se han planteado diversas alternativas para la construcción del sistema. La elección de las opciones adecuadas a necesidades específicas permitirá un buen funcionamiento global del equipo.

Capítulo 2

DESCRIPCIÓN DEL SISTEMA

En el capítulo anterior se describieron los fundamentos de operación de cada uno de los bloques del sistema. También se enumeraron diversas opciones para integrar la red de adquisición de datos, relacionando cada opción con la teoría que lo respalda. En el presente capítulo se definirán los bloques que integrarán el sistema, elegidos a partir de los descritos previamente. Al final del capítulo se tendrá una visión general de la constitución del proyecto.

Al tratar con el diseño de un sistema de adquisición de datos es posible marcar dos grandes categorías para los subsistemas que lo componen: el *hardware* y el *software*.

2.1. *Hardware*

Se conoce por *hardware* al conjunto de dispositivos electrónicos que interactúan para lograr un propósito determinado. Este concepto se refiere a la integración física del sistema, y no así a las reglas de operación del mismo. Para poder caracterizar adecuadamente el *hardware*, es necesario tratar con cada una de sus partes constituyentes. La figura 2.1 muestra un diagrama a bloques del sistema.

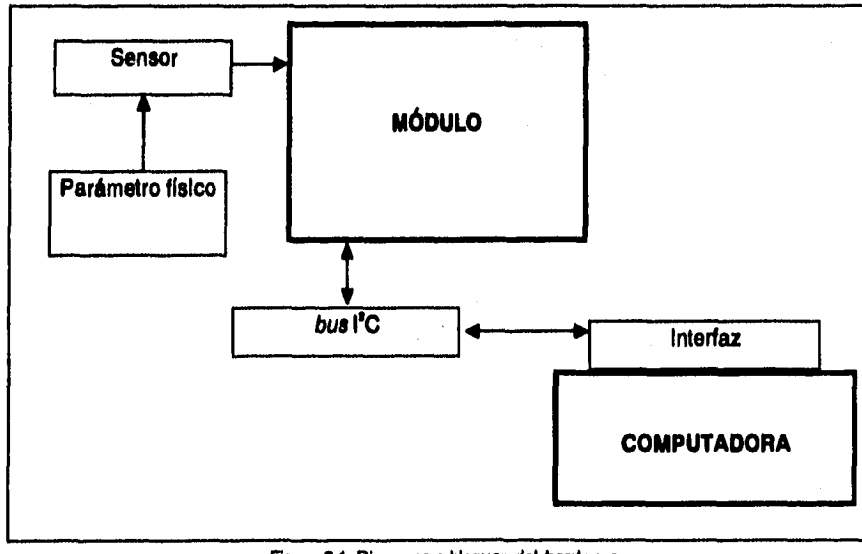


Figura 2.1. Diagrama a bloques del *hardware*.

2.1.1. Arquitectura del sistema de adquisición de datos

- TOPOLOGÍA

La topología del sistema es inherente a su función principal: la concentración de datos provenientes de módulos autónomos.

Dado que cada módulo opera de manera independiente, y que el sistema debe permanecer en operación constante, la topología más adecuada es la de un bus lineal. Esta configuración permite el funcionamiento correcto del sistema aún en ausencia de varios de sus elementos.

- MEDIO DE TRANSMISIÓN

Otra consideración que lleva a la elección de un bus lineal es el costo del medio de transmisión. La utilización de un bus lineal de número reducido de hilos permite emplear conductores tan simples como par trenzado para distancias relativamente pequeñas (menores a 500 metros).

Para la elaboración del prototipo se utilizará cable telefónico de cuatro hilos como conductor, puesto que las condiciones de operación serán perfectamente conocidas y controladas. Sin embargo, en una aplicación de mayor escala pueden utilizarse convertidores electro-ópticos para emplear comunicaciones por fibra.

- FORMATO Y CARACTERÍSTICAS ELÉCTRICAS

Como ya se expuso anteriormente, la opción de enviar señales en forma paralela no resulta muy eficiente por la distancia de transmisión reducida y un gran costo de cableado. Por estas razones, queda descartada la idea de tener un bus paralelo.

Por otro lado, el uso de un bus serie presenta la ventaja de la distancia, aún cuando la velocidad de transmisión no llega a ser extremadamente alta. Dentro de las opciones de transmisión serie, la interfaz RS232C también queda descartada, puesto que la distancia que se puede alcanzar es sumamente corta.

La transmisión serie se adapta mejor al sistema requerido, donde las características eléctricas de la interfaz RS485 permiten suficiente distancia y

flexibilidad para integrar la red. Sin embargo, la enorme ventaja que presenta el bus I^2C con el protocolo integrado en los circuitos es determinante. Gracias a esta característica, el sistema puede enlazar módulos de diferentes naturalezas, como pueden ser microprocesadores, unidades de respaldo de información y convertidores A/D entre otros. Adicionalmente, el protocolo I^2C permite la operación de varios elementos inteligentes en el bus, como se detalla en páginas subsecuentes. Esto brinda una gran versatilidad al diseño de la red, incluyendo los diseños de interfaz simples gracias a la ausencia de lógicas de decodificación complejas.

2.1.2. Características físicas del *bus* I^2C

La mayoría de los sistemas electrónicos, sin importar su área de aplicación, cuentan con algunos bloques comunes. Tales son: un control inteligente - generalmente un microcontrolador -, circuitos de propósito general - como memoria, puertos o convertidores - y circuitos de propósito específico - como sintonizadores, procesadores de señales y decodificadores DTMF -. Para explotar estas características comunes, Philips desarrolló un bus bidireccional de 2 líneas para controlar eficientemente las comunicaciones entre circuitos integrados. Al simplificar grandemente los diseños, el bus de Interconexión entre Circuitos Integrados (I^2C) permite comunicación bidireccional entre más de 150 circuitos compatibles de cualquiera de las tres categorías mencionadas anteriormente. Todos los circuitos integrados compatibles con I^2C incorporan una interfaz que les permite comunicarse entre sí utilizando el protocolo.

Algunas de las características del bus son:

- Se requieren solamente dos líneas para la transmisión: una línea de datos seriales (SDA) y una de reloj (SCL).

- Cada dispositivo conectado al bus tiene una dirección única. Ésta permite el acceso por *software* al dispositivo estableciendo relaciones maestro/esclavo en todo momento.
- En las comunicaciones maestro/esclavo, el maestro puede actuar como maestro-receptor o como maestro-transmisor.
- El protocolo detecta colisiones y evita la corrupción de la información debido a accesos simultáneos por varios maestros.
- La transferencia de datos en forma serie, enfocada a *bytes* de 8 *bits*, puede alcanzar velocidades de hasta 100 kbps en condiciones normales ó 400 kbps si se utiliza el modo de alta velocidad.
- El número de dispositivos que pueden conectarse al bus es infinito. Solo es necesario mantener la capacitancia del bus por debajo de 400 pF.

Éstas son solo algunas de las características del bus. La mayoría de ellas permiten al diseñador una mayor flexibilidad y simplicidad derivadas de que la interfaz está integrada en cada uno de los circuitos compatibles con I^2C. Además, la presencia o ausencia de un dispositivo en el bus no afecta en absoluto la operación de la red, haciendo al sistema un modelo de gran generalidad y versatilidad.

Las características del protocolo son el resultado de satisfacer las necesidades de funcionamiento del bus. Se desea un sistema que conlleve el mínimo costo de conexión entre circuitos. Este sistema no requiere velocidades de transmisión muy elevadas, y su eficiencia global dependerá mayoritariamente de las características de los circuitos interconectados.

Todas estas condiciones llevan a la idea de un bus serie. A pesar de no tener una velocidad tan grande como los paralelos, estos buses requieren menor número de conexiones y , desde el punto de vista de un fabricante de circuitos integrados, menos terminales en los dispositivos.

2.1.3. Características de los módulos

Los módulos deben presentar una serie de características que los hagan adecuados para la operación con el bus I^2C . Evidentemente se prefiere utilizar dispositivos que ya tengan la interfaz como parte de su diseño original, evitando así la fabricación de equipo adicional.

Philips produce una gran variedad de dispositivos que tienen integrada la interfaz I^2C, lo que permite la construcción de cualquier sistema aprovechando esta tecnología. Debido a que los módulos que registrarán la información deben ser autónomos, una característica esencial debe ser la capacidad de memoria. Además, puesto que éste será el punto de transición entre el mundo real analógico y la transmisión digital, el módulo deberá contar con convertidores A/D.

Existen tanto módulos de memoria como convertidores A/D equipados con interfaces I^2C, bloques en los que se podría basar la construcción del equipo. Sin embargo, resulta mucho más simple y versátil utilizar un microcontrolador que incluya dentro de sus características a estos equipos. La principal ventaja de configurar así el sistema es la capacidad ampliada de dispositivos en el bus. Puesto que el número máximo de terminales conectadas al bus está en función de la capacitancia de los equipos, el uso de un microcontrolador reduce a la mitad el número de dispositivos necesarios para el registro de información, como se muestra en la figura 2.2.

Otra ventaja de construir el sistema de esta forma es la eficiencia en el manejo del *bus*. Al utilizar un microcontrolador, el intercambio de información entre convertidor y memoria no tiene que utilizar el *bus*, permitiendo una operación más ágil.

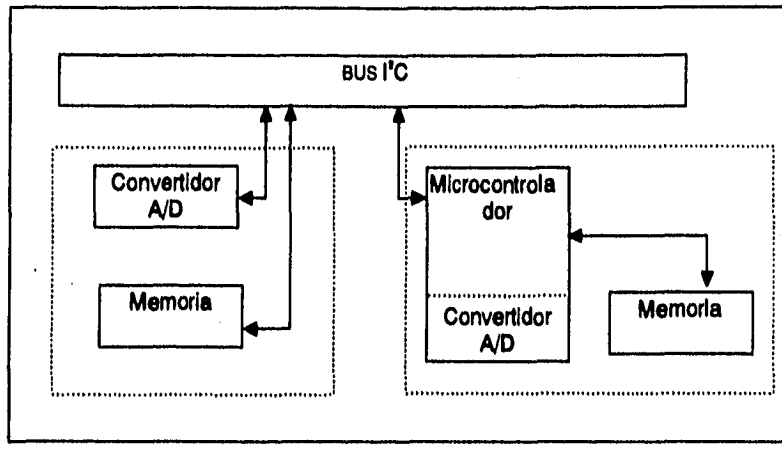


Figura 2.2. Comparación entre dos configuraciones de módulos.

Adicionalmente, muchos de los microcontroladores Philips tienen otro tipo de puertos de salida que permitirían un sistema con redundancia y en general mayor flexibilidad. Tal es el caso de la salida de modulación por ancho de pulso del 80C552, microcontrolador que además de tener convertidores A/D e interfaz I²C cuenta con salidas serie para formato RS232C. Al contar con todas estas funciones, en el supuesto caso de una falla en el bus, la información podría ser recuperada a través de un medio alternativo, o bien almacenarse en la memoria indefinidamente.

Indudablemente, la precisión de la información obtenida está en función de la resolución de los convertidores que se utilicen. Los convertidores más comunes utilizan 8 *bits* para la conversión, permitiendo 256 lecturas distintas en un intervalo definido. Si se deseara mayor precisión, podría utilizarse un convertidor de 16 *bits*, considerando las modificaciones pertinentes en el *software* que controla la conversión y la transmisión I²C. Para este fin se armaría una palabra de 16 *bits* a partir de dos *bytes* transmitidos, logrando el resultado deseado.

La programación del microcontrolador resulta simple, pues las funciones que realizará no son elaboradas. Necesitará la señal de inicio de conversión, una de fin de conversión, una de almacenamiento en memoria, etc. y todas forman parte del programa que ejecuta el 80C552.

Estas son algunas de las ventajas más evidentes de utilizar una configuración de microcontrolador como elemento central del módulo de adquisición de datos. De este modo, el diagrama presentado en la figura 2.1 se convierte en el mostrado en la figura 2.3 Sin embargo, pueden realizarse estudios posteriores para comparar su desempeño con el de otro tipo de arquitectura, como podrían ser los convertidores independientes.

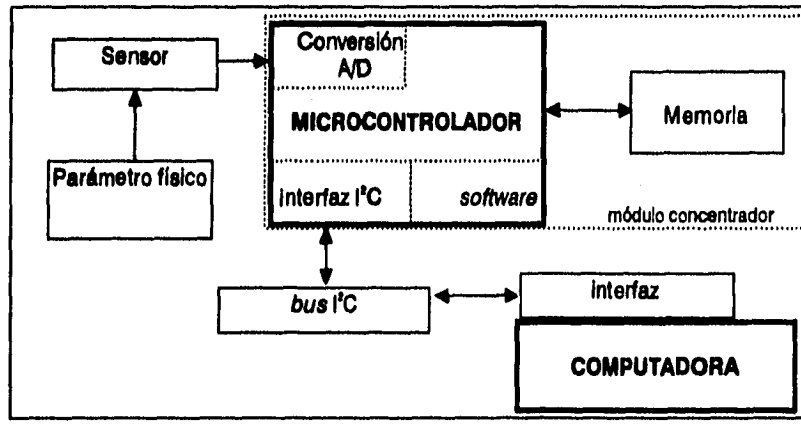


Figura 2.3. Diagrama a bloques del *hardware*.

Para la medición de temperatura se utilizará el sensor LM35, por presentar las características más adecuadas al proyecto, incluyendo simplicidad de configuración. El intervalo de temperaturas que se pretenden medir está limitado a las ambientales, por lo que la precisión y el intervalo de operación de este sensor son ideales. Además, es un dispositivo económico y de operación simple.

La medición de humedad se simulará utilizando un potenciómetro, pues los sensores comerciales tienen un costo elevado y poca disponibilidad. Esto es adecuado a los fines ilustrativos del proyecto a un costo moderado.

De los sensores discutidos para llevar a cabo la medición de la iluminación, la opción de la fotocelda resultó idónea. Esto es debido a que su disponibilidad es excelente, por lo que además resulta mucho más económico. Las lecturas que se llevaran a cabo son de iluminación relativa. Éstas permitirán la determinación de niveles adecuados de iluminación con la precisión requerida.

2.1.4. Características de la interfaz con la computadora

Esta es posiblemente la etapa más importante del sistema. Es aquí donde los datos adquiridos por módulos independientes llegarán en formato I^2C para ser almacenados en la computadora personal. Es por esto que se exploraron varias opciones, siendo las siguientes las más relevantes:

- MÓDULO INTERNO

La construcción de la interfaz como módulo interno, así como las demás opciones, presenta numerosas ventajas y desventajas. La mayoría de las ventajas están enfocadas a la velocidad de operación del sistema, y su principal desventaja radica en el diseño específico de la tarjeta para cada computadora. A continuación se considera el caso de una tarjeta para computadora compatible con IBM-PC.

La inserción de una tarjeta en las ranuras de expansión de la computadora brinda la gran oportunidad de utilizar enormes velocidades al poder emplear el acceso directo del DMA. Para estos fines, solamente es necesario diseñar una simple lógica de decodificación para asignar una dirección de memoria a la tarjeta. En esta tarjeta se colocará un circuito que proporcione el formato I^2C a partir de la información disponible

en el bus de datos de la computadora. De esta forma se tiene un elemento maestro/esclavo en el interior de la computadora, mismo que permite un elaborado sistema multimaestro a velocidades estupendas, pues la conversión de protocolo se realiza por *hardware*, no por *software*.

Otra de las grandes ventajas de esta configuración es que se dispone de la fuente de alimentación de la computadora, la cual puede energizar todos los circuitos que se requieren, así como el propio bus. En la figura 2.4 se muestra esta configuración.

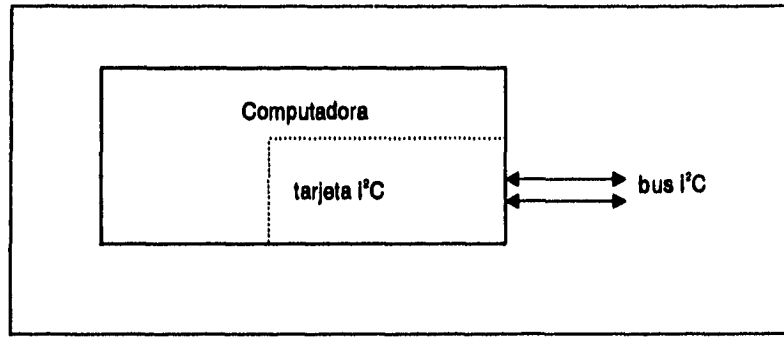


Figura 2.4. Interfaz como módulo interno.

La tarjeta que se debe elaborar queda perfectamente determinada solamente si se conoce la arquitectura de la computadora a utilizar. Esto significa que el sistema sería uniplataforma, característica no deseable. Sin embargo, puede diseñarse un equipo para cada una de las plataformas computacionales más comunes, donde sin duda debe comenzarse por las IBM-compatibles.

- UTILIZACIÓN DE LA COMPUTADORA COMO ELEMENTO MAESTRO

Para evitar la molesta y posiblemente complicada tarea de insertar una tarjeta más en el interior de la computadora, puede utilizarse la computadora como parte integral del sistema I²C, donde a través del puerto paralelo se realizará el intercambio de información. No sólo eso, sino que al emplear el puerto paralelo como salida puede utilizarse cualquier tipo de computadora o microprocesador como elemento del sistema.

Utilizando un par de entradas configuradas con colector abierto del puerto paralelo, es posible tener control sobre las líneas SDL y SCL del bus, permitiendo perfectamente la operación del sistema como se muestra en la figura 2.5. Otra gran ventaja es, nuevamente, la disponibilidad de polarización directamente de la fuente de la computadora.

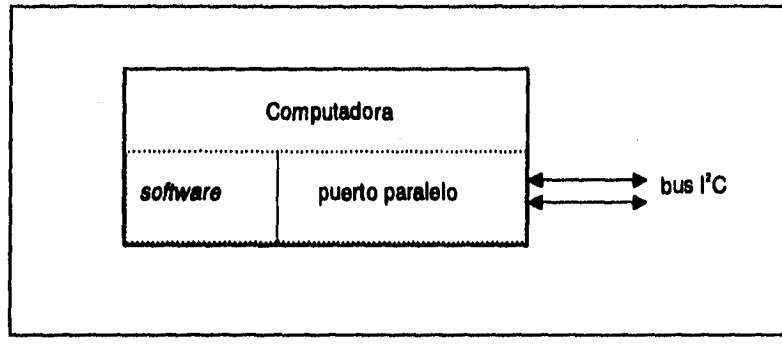


Figura 2.5. Computadora como elemento maestro.

Sin embargo, una desventaja importante está en la implementación del protocolo. Puesto que el puerto paralelo pasa a formar parte del sistema I²C, la instrumentación del protocolo deberá hacerse por *software*, donde la información que llegue al puerto ya tenga todas las condiciones necesarias para su correcta transmisión, recepción e interpretación por otros dispositivos. Esta tarea, además de

representar una carga importante para la capacidad de procesamiento de la computadora, implica el diseño de todas las rutinas para transferencia de datos, contemplando las posibles colisiones, etc.

- MÓDULO EXTERNO

Hasta este momento las características más útiles que se han obtenido son la independencia de la plataforma computacional y el protocolo en *hardware*. Puesto que la primera característica se obtiene solamente al tener el equipo fuera de la computadora, debe considerarse la opción de un módulo externo. Además, la utilización de circuitería que de formato a la información es una alternativa digna de seria consideración.

Haciendo uso de los circuitos que trabajan en el módulo interno, puede extraerse la información de la computadora a través del puerto paralelo, en forma paralela, y alimentarse posteriormente a la etapa de conversión, de donde saldrá con formato I^2C. Esto permitirá que el sistema sea compatible con cualquier computadora capaz de controlar una impresora común. Indudablemente, el diseño que permita la extracción de la información, y en particular la recepción, requerirá más atención, misma que se le dará posteriormente.

Al utilizar comunicación en serie, es indispensable contar con una referencia de tiempo, misma que hasta el momento podría ser provista por el oscilador de la computadora. Sin embargo, al construir un módulo independiente se pierde esta ventaja, teniéndose que construir un oscilador independiente. Cabe destacar que no es importante la sincronía entre éste y el de la computadora, pues la velocidad de transmisión en el bus no tiene relación alguna con la operación de sus terminales.

La fuente de energía es otra ventaja que se sacrifica al utilizar un módulo externo, pero es fácilmente subsanada al construir un simple circuito de rectificación y regulación que permita conectar el equipo al suministro comercial. Esta solución se representa en la figura 2.6.

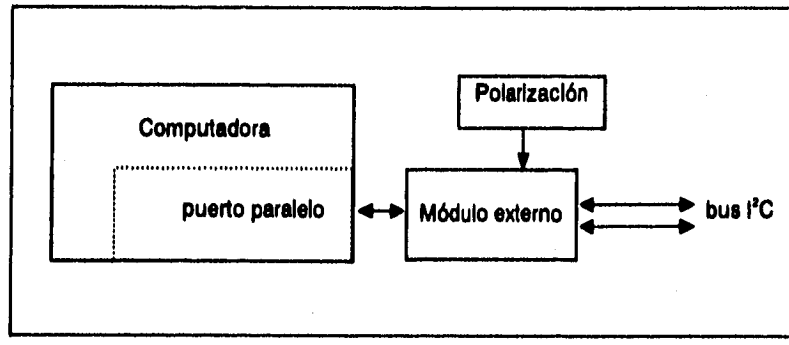


Figura 2.6. Interfaz con módulo externo.

Este es el último de los parámetros de *hardware* por definir. A partir de estas alternativas se propondrá posteriormente el diseño detallado del sistema. La constitución, en forma general, queda definida como se muestra en la figura 2.7.

2.2. *Software*

Una vez establecida la infraestructura necesaria para lograr los objetivos propuestos, es necesario describir la forma en que interactuarán los subsistemas que componen el proyecto. Para este fin se utiliza el *software*, un conjunto de instrucciones que permiten el procesamiento de la información así como su intercambio con subsistemas distintos.

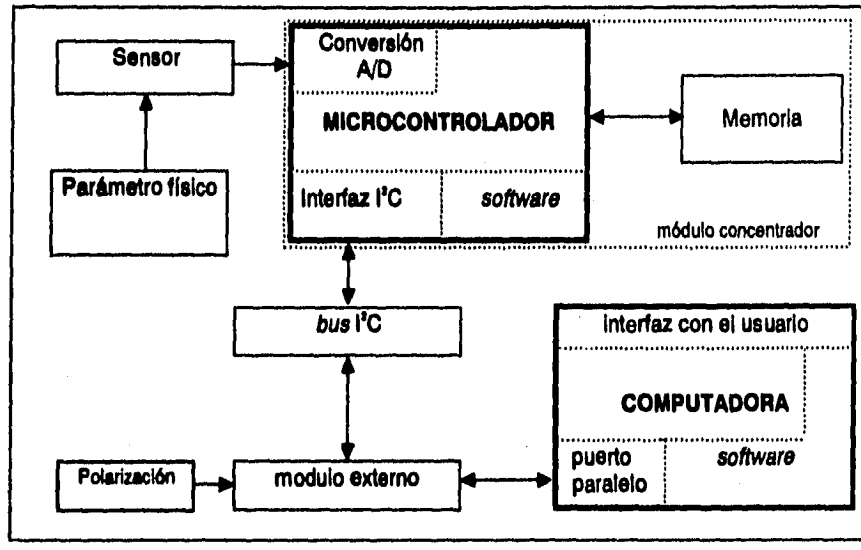


Figura 2.7. Diagrama a bloques del *hardware*.

Para el caso aquí presentado, existen distintos puntos donde el *software* desempeña un papel importante. Uno de ellos, posiblemente el principal, es el encargado de la correcta transmisión de los datos adquiridos en alguno de los módulos. Otro elemento de *software* es el que ordena al microcontrolador que registre la información del convertidor A/D proveniente del mundo real, y aún otro elemento es el encargado del almacenamiento e interpretación de la información en la computadora terminal. A continuación se describen algunos de estos puntos.

2.2.1. Protocolo del *bus* I^2C

Se ha descrito la gran ventaja de un bus serie sobre un paralelo en términos de la complejidad de los circuitos, así como del espacio que ocupan. Sin embargo, el bus está constituido por la capa física y la capa de transporte, esto es, las rutinas y formatos que regulan la comunicación son también parte del bus.

Los dispositivos que se comunican entre sí en un bus serie deben tener un protocolo que elimine toda posibilidad de confusión, pérdida o bloqueo de información. Por si esto fuera poco, el protocolo debe garantizar la comunicación entre dispositivos de distintas velocidades, y la calidad de la transmisión debe ser independiente de los dispositivos que la realicen. Debe emplearse un criterio que ceda el control del bus a un solo dispositivo a la vez. Todo esto se considera al diseñar el protocolo utilizado en el bus I^2C.

El bus I^2C soporta cualquier tipo de fabricación de circuitos: CMOS, bipolar, etc. Se utilizan dos líneas - SDA y SCL - para transmitir información entre los dispositivos conectados al bus. Cada dispositivo, sin importar su función, reconoce su dirección única y opera como transmisor o receptor, dependiendo del dispositivo y del maestro de la comunicación. Esto significa que el maestro puede solicitar a una memoria el proceso de escritura o de lectura, pero sólo podrá, evidentemente, escribir a un monitor de cristal líquido.

El maestro de una comunicación es aquel dispositivo que inicia la transferencia de datos en el bus y genera el reloj que permite la transmisión. Durante esta transferencia, todos los demás dispositivos en el bus son esclavos.

Otra característica del bus I²C es su posibilidad de operación con múltiples elementos maestros. Esto significa que hay más de un dispositivo conectado al bus capaz de iniciar y controlar la transferencia de información. Generalmente se utilizan microcontroladores como elementos maestros, pudiéndose configurar como esclavos de ser necesario.

La posibilidad de conectar más de un microcontrolador al bus significa que dos o más podrían iniciar la transferencia simultáneamente. Puesto que los dispositivos monitorean la información al momento de transmitirla, el protocolo identifica la colisión cuando alguno de los maestros transmite un "uno" y detecta un "cero". La conexión de AND-alambrada al bus permite la existencia de esta situación, misma que provoca la interrupción de la transmisión de todos los elementos en conflicto.

Como puede verse en la figura 2.8, si alguno de los elementos conectados al *bus* transmitiera un nivel lógico alto mientras un segundo elemento transmite un nivel bajo, el primero recibirá un *bit* distinto del que envió, detectando la colisión y suspendiendo su transmisión.

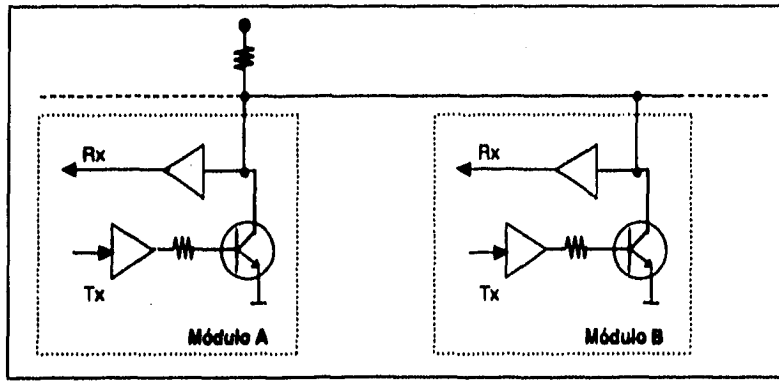


Figura 2.8. El *bus* I²C.

Si, por ejemplo, el transmisor del módulo A emite un nivel lógico alto, el transistor correspondiente se saturaría, llevando el *bus* a un nivel eléctrico bajo. Si en el mismo instante el transmisor del módulo B emite un nivel lógico bajo, el transistor correspondiente permanece en corte, y el *bus* debería permanecer en "1". Sin embargo, el *bus* ya manifiesta un "0" debido a las acciones del módulo A. Cuando el módulo B recibe un *bit* distinto del que transmitió, interrumpe la transmisión.

Tanto la línea de datos como la de reloj son bidireccionales, y se conectan a un voltaje positivo a través de un resistor de pull-up. Cuando el bus está libre, ambas líneas están en un nivel alto. Las etapas de salida de los dispositivos deben tener salida de colector abierto para conformar la AND-alambrada en el bus. Los niveles de voltaje del bus son totalmente flexibles, pues contemplan la posibilidad de interconectar familias distintas.

Para que los datos se consideren válidos, la señal de datos (SDA) debe permanecer constante durante el nivel alto del reloj (SCL). Si ocurre una transición de alto a bajo, el bus la considera una señal de arranque (*Start*); si la transición es al revés, considera una de paro (*Stop*). Estas dos condiciones son generadas exclusivamente por los maestros. Después de un *Start*, el bus se considera ocupado hasta después de presentarse un *Stop*.

La transferencia de datos contempla que cada *byte* transmitido sea de 8 *bits*. El número de *bytes* que puede transmitirse en cada transferencia es ilimitado, aunque cada uno debe ser seguido de un *bit* de reconocimiento (*Acknowledge*). La información se transmite del *bit* más significativo (MSB) al menos significativo (LSB). Si el receptor no logra obtener un *byte* completo, puede mantener la línea de reloj (SCL) en nivel bajo, provocando un estado de pausa (*Wait*).

La señal de reconocimiento, cuyo pulso de reloj asociado es producido por el maestro, es indispensable. Durante este pulso, el receptor debe llevar la línea SDA a un nivel bajo. Si el receptor no envía la señal de *Acknowledge* (ACK), el transmisor puede enviar un *Stop* para abortar la transferencia. Un caso particular es cuando el maestro actúa como receptor, donde él mismo debe generar el *bit* de ACK.

La transferencia de datos se describe a través de los siguientes diagramas de bloques, identificados como figuras 2.9 y 2.10. Para transmitir un mensaje entre dispositivos compatibles con I^2C, es necesario transmitir una señal de comienzo (*Start*) así como generar la señal de sincronía o reloj durante toda la transmisión. Se transmiten *bits* que conforman la dirección propia del destinatario del mensaje, y si se recibe la señal adecuada (*Acknowledge*) se continúa la transmisión de bytes de información coordinados con la señal de reloj. Cuando se desea suspender la transmisión, se genera la condición de paro (*Stop*).

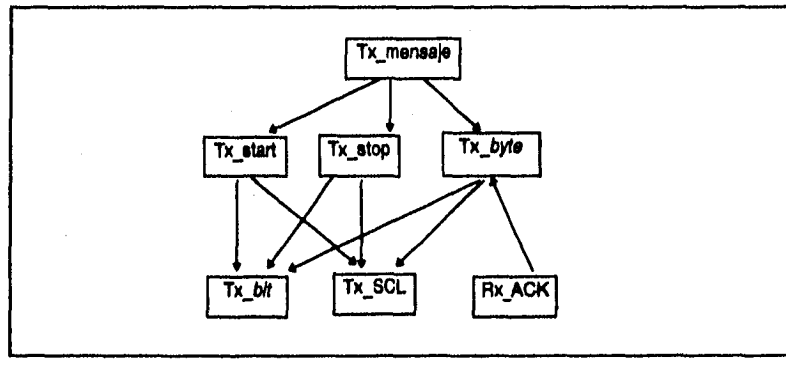


Figura 2.9. Esquema de transmisión.

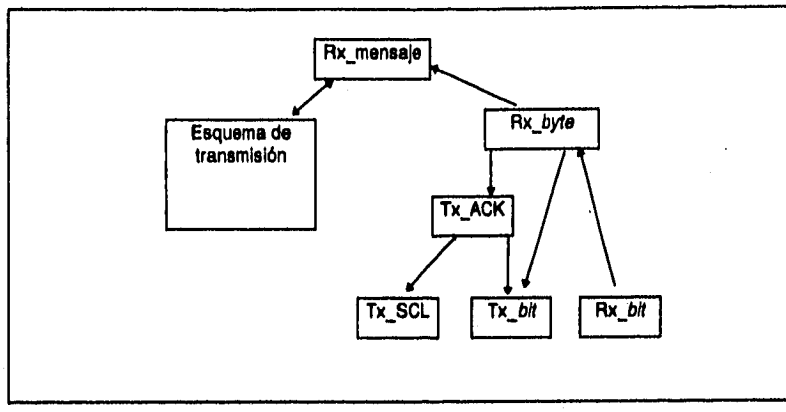


Figura 2.10. Esquema de recepción de información.

La recepción de información sigue un esquema muy similar. Aprovecha las funciones de transmisión para establecer la comunicación - esto es, transmitir la dirección I^2C, definir la operación, etc. Hecho esto, el dispositivo maestro se encarga de generar el reloj, transmitir señales de *Acknowledge* y recibir *bytes*.

Ambos procedimientos se explican con más detenimiento posteriormente, y en el capítulo cuatro se detalla la implementación específica con *Turbo Pascal 6.0* de algunas de las rutinas mencionadas.

En la transmisión de la primera palabra se definen las características de la transmisión. Este primer *byte* contiene: un *bit* de *Start*, la dirección I^2C del esclavo en cuestión, y un *bit* que define si la operación es de lectura o escritura (R/W). Después de la transmisión de esta palabra, la transferencia de información puede verificarse hasta la presencia de una condición de *Stop* o una nueva condición de *Start* dirigida a otro esclavo.

A continuación, en la figura 2.11, se presenta un ejemplo de transmisión entre dos dispositivos, contemplando algunas de las situaciones anteriormente mencionadas. Obviamente existen muchas más especificaciones que definen completamente al bus, pero que van más allá de los objetivos de este trabajo. La especificación completa se incluye como apéndice de este trabajo.

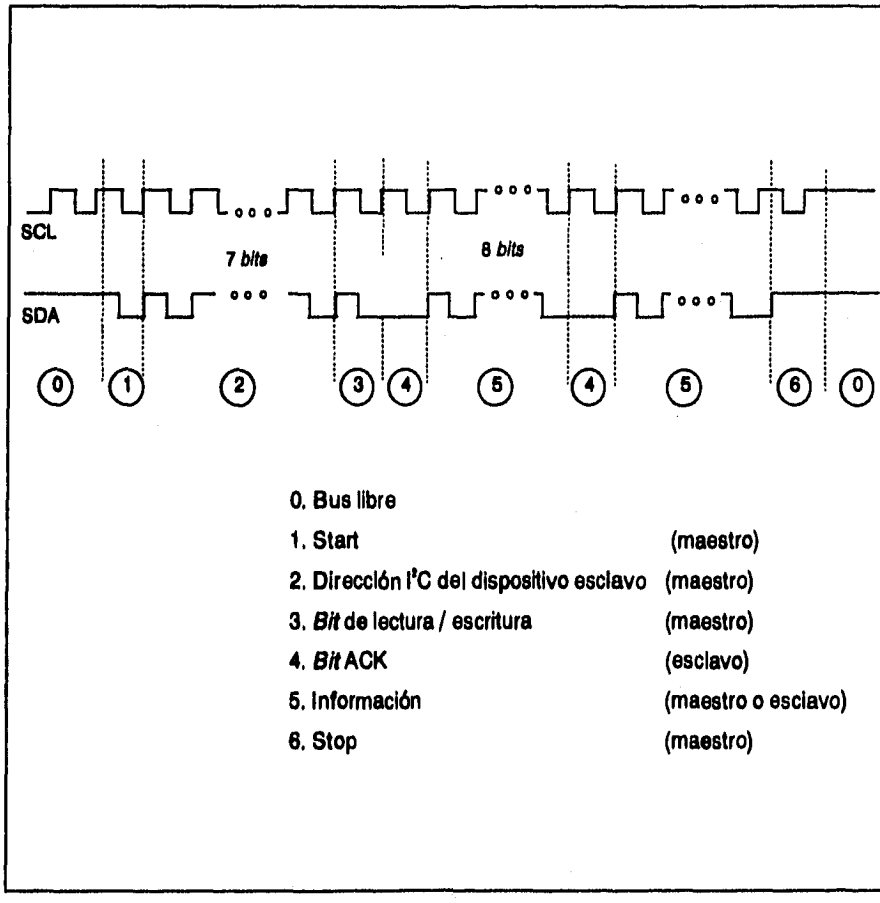


Figura 2.11. Ejemplo de intercambio de información.

En el ejemplo anterior, el maestro genera una condición de *Start* a partir del bus libre. Posteriormente, transmite la dirección I^2C del dispositivo esclavo en cuestión, seguido por la instrucción de lectura / escritura. La comunicación se considera establecida cuando el esclavo transmite (ACK), momento en que el maestro inicia la transmisión de información en palabras de 8 *bits* seguidas por la recepción de (ACK). La transmisión de este ejemplo termina cuando el maestro genera *Stop*, pero podría terminar cuando cesa de recibir (ACK) tras un *byte* de información o bien al generar una nueva condición de *Start* con otra dirección o instrucción.

2.2.3. Estructura de los programas

La programación que se encarga de concentrar e interpretar la información para que el usuario pueda acceder a ella cómodamente se divide en dos grandes grupos: uno, que está enfocado a la adquisición de los datos provenientes del *bus* I^2C en archivos de computadora personal, y otro que está encargado de darle una presentación accesible, así como brindar herramientas para producir reportes y gráficas. La programación para realizar las mediciones se realiza en lenguaje ensamblador para el microcontrolador 80C552.

- DESCRIPCIÓN DE LA MICROPROGRAMACIÓN

El propósito del programa concentrador es tomar las lecturas analógicas y digitalizarlas para almacenarlas en RAM. Sin embargo, las lecturas analógicas deben convertirse a digitales para ser manejadas por el microcontrolador. Esto se logra a través del convertidor analógico-digital de ocho canales.

Para leer todos los canales , el microcontrolador multiplexa en el tiempo la operación del convertidor. De esta forma el usuario puede determinar qué canal será convertido.

Los datos ya digitalizados se almacenan en memoria para poder guardar la información un tiempo limitado - hasta que se solicita la transmisión por el *bus* I^2C. Una vez llena la memoria, las lecturas nuevas se pierden hasta que toda la información ha sido transmitida y la memoria está lista para recibir más información.

Si se le pide al módulo más información de la que tiene almacenada, procederá a funcionar como "monitor", enviando las lecturas más recientemente adquiridas.

Para subsanar medianamente la falta de detección de errores, es necesaria la transmisión de bits de sincronía. Estos sirven para indicar a la computadora el inicio de un bloque de cinco lecturas, presentándose al inicio de la transmisión I^2C y posteriormente entre los "bloques" de cinco lecturas. En caso de utilizar más canales, sería necesario redimensionar estos "bloques".

Con el fin de no confundir los *bits* de sincronía con *bits* de información, se definió arbitrariamente el dato $99 como *bit* de control. Si el resultado de alguna conversión A / D diera por resultado $99, el programa lo transforma en $98 para evitar confusiones.

Las lecturas se obtienen según las especificaciones del microprocesador 80C552, como se detalla en capítulos posteriores. A través del uso de registros establecidos en la memoria interna del *chip*, se controla la conversión, se almacena la información y, de ser requerido, se transmite por el *bus* I^2C.

El procedimiento se decribe a grandes razgos a continuación. Se inicia la conversión del primer canal, direccionado a través del registro de control (ADCON) y estableciendo la condición de inicio. Se revisa contínuamente el registro de estado (ADSTA) para verificar que la conversión haya concluído. El resultado de la conversión puede leerse del registro designado a la lectura de datos (ADCH).

El dato binario se extrae del registro de conversión y se almacena en la memoria externa. Cuando este procedimiento se ha repetido cinco veces, se revisa el estado de la bandera de interrupción del bus I²C. En caso de que no esté activada, se prosigue a hacer un nuevo juego de cinco lecturas y se guardan en memoria. Si la bandera se encuentra activada, se procede a una rutina de transferencia de información.

La rutina de transferencia se encarga de llevar uno por uno los datos guardados en la memoria al registro de datos del *bus* I²C (S1DAT). Posteriormente se revisa la presencia de la señal de ACK, proveniente del PCF8584. Una vez transmitido y recibido el dato, se sigue el mismo procedimiento para todos los subsecuentes.

Al alcanzar el tope de la memoria (la última localidad de memoria RAM direccionable) se continúa con el procedimiento de monitoreo. Con éste se envían directamente las lecturas del convertidor analógico digital al *bus* I²C. Para esto, se realizan las cinco conversiones e inmediatamente se transmiten por el bus. Los bytes de sincronía también se envían al término de las cinco lecturas.

Cualquiera de los procedimientos anteriores se interrumpe si se detecta una condición de STOP en el *bus* I²C. Dicho evento es únicamente posible cuando el PCF8584 la genera, pues siempre tendrá el papel de maestro en las comunicaciones. Al darse esta condición, el microcontrolador regresa al inicio de su programa para guardar nuevamente los datos en la memoria. De esta forma se consigue que el 80C552 se mantenga en espera de una nueva llamada para transmisión por el *bus* I²C.

- ADQUISICIÓN DE DATOS

Los datos que llegan al puerto paralelo pasan por una etapa de adecuación, donde se contemplan las inversiones de *bits* debido al *hardware*, escalas específicas de cada parámetro, etc. Ya adecuados, los datos se almacenan en dos archivos: uno, para revisión posterior o histórico; otro, para monitoreo. La diferencia entre éstos es que en la historia se agrega información, mientras que en el otro se actualiza. Al final de 24 horas, el archivo histórico se guarda permanente y automáticamente para consultas posteriores. Hecho esto, la historia inicia de nuevo.

Además de la función de adecuación de los datos, este programa se encarga de la comunicación con la interfaz, coordinando la escritura y lectura de datos, así como el puerto a que se conecta el equipo. Es a través de este programa que se recibe la información I²C convertida a formato paralelo. Adicionalmente, es este programa el que permite configurar el microcontrolador PCF8584, corazón de la interfaz.

En la figura 2.12 se muestra la estructura del programa de adquisición de datos, módulo básico para la operación del sistema. La utilización de un archivo de monitoreo independiente al histórico permite que se mantenga la integridad de la información, pues el monitoreo puede realizarse a través de un archivo temporal y efímero, mientras que el archivo histórico debe registrar cada una de las lecturas recibidas.

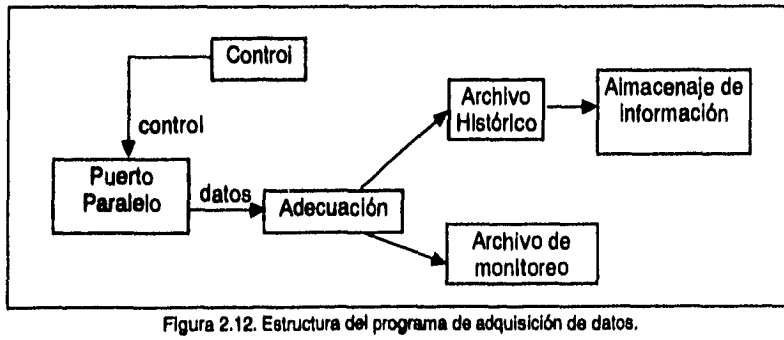


Figura 2.12. Estructura del programa de adquisición de datos.

Este módulo fue desarrollado en TurboPascal 6.0. Se eligió este lenguaje por ofrecer las ventajas propias de una alto nivel de programación y por ser más rápido que, por ejemplo, QuickBasic, el más simple de los lenguajes descritos anteriormente. La versión del programa que utiliza el sistema está compilada en un archivo ejecutable, obteniéndose mayor velocidad.

- INTERPRETACIÓN DE DATOS

Las características de este módulo están enfocadas a resolver las necesidades de usuarios que no necesariamente tienen conocimiento de programación. Esto implica que el *software* con que interactúen debe ser amigable, reduciendo la posibilidad de errores y problemas al mínimo. Su estructura se muestra en la figura 2.13.

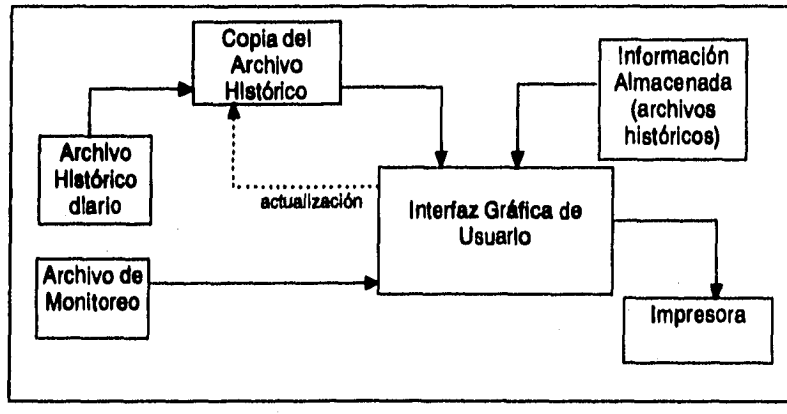


Figura 2.13. Estructura del programa de interpretación de datos.

La información almacenada en los dos archivos por el módulo de adquisición de datos es extraída por la interfaz gráfica, representándola en forma de tablas o gráficas según el usuario lo seleccione. Esta interfaz es capaz de manejar el monitoreo de la información, extrayéndola del archivo de monitoreo, analizar el comportamiento de parámetros a lo largo del día, utilizando el archivo histórico, o consultar archivos generados en fechas previas y que han sido almacenados en el disco duro de la computadora. Sobra mencionar que el programa puede imprimir todas las representaciones que genera en pantalla, así como administrar los archivos que él mismo genera, editando, abriendo o eliminándolos.

Este módulo se desarrolló con el lenguaje *Visual Basic 3.0*, y está específicamente diseñado para trabajar bajo Windows 3.X y posteriores. Se eligió este lenguaje por su gran versatilidad para diseño y construcción de interfaces gráficas, así como su simplicidad de programación. Fue necesario considerar la plataforma a utilizar, pero es claro que la enorme mayoría de las computadoras personales compatibles con IBM utilizan el ambiente Windows.

Es importante destacar que la adquisición de los datos no requiere Windows, sino que aprovecha las características multitarea del ambiente para permanecer residente en memoria y continuar su ejecución en segundo plano. Esto es aún más importante porque la adquisición de datos puede realizarse en una computadora XT y posteriormente analizarse en una máquina que ejecute Windows 3.11 ó superior.

Podría pensarse en el desarrollo de la adquisición de datos con Visual Basic. Sin embargo, este lenguaje, en la versión disponible (3.0 Professional) complica considerablemente el manejo de puertos. Si bien Visual C++ permite estos manejos, se consideró como parámetro prioritario el tiempo de aprendizaje del lenguaje.

2.3. Características globales del sistema

Una vez conociendo todas las partes involucradas en el diseño y construcción del sistema, podemos integrar un modelo que lo represente en forma general esquemáticamente. La figura 2.14 muestra de nuevo dicho esquema.

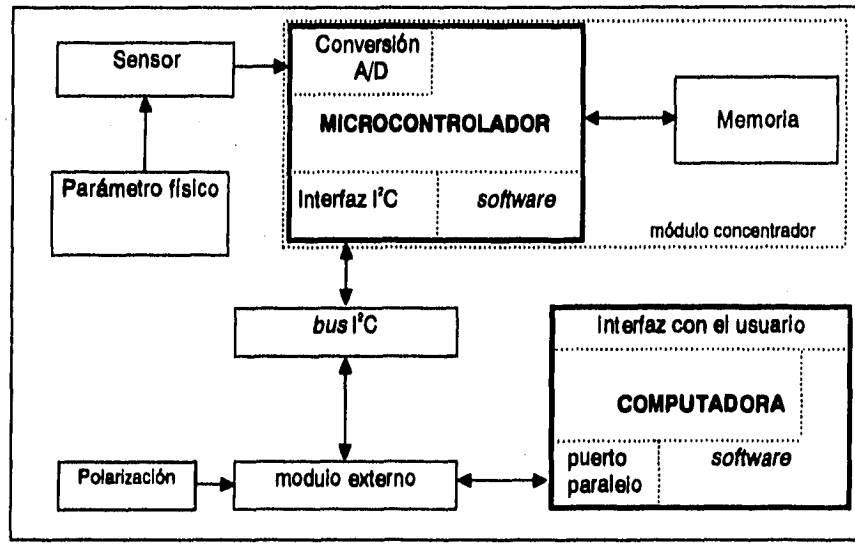


Figura 2.14. Diagrama de bloques del sistema

A través de la interacción de este *hardware*, regida por el *software* descrito, podrá obtenerse mediciones, almacenarlas en memoria, transmitirlas, recibirlas e interpretarlas en una computadora personal. Las características de cada uno de los módulos quedaron esbozadas para sentar las bases del diseño que a continuación se presenta.

Capítulo 3
DISEÑO DEL SISTEMA

A lo largo del capítulo anterior, el diseño quedó determinado a nivel de diagramas de bloques. A continuación se presentarán los diseños detallados de cada uno de los elementos del sistema.

3.1. *Hardware*

En este apartado se definirá detalladamente la composición de cada uno de los subsistemas del proyecto mostrados en la figura 2.14. Para tal efecto se discutirán independientemente los módulos de adquisición de datos, incluyendo los sensores, y la interfaz con la computadora.

3.1.1. DISEÑO DE LOS "MÓDULOS CONCENTRADORES"

De acuerdo con las características deseadas de los "módulos concentradores", éstos se diseñaron alrededor de un microcontrolador Philips con puerto I²C : el 80C552, cuyas especificaciones se encuentran en el apéndice A.

La configuración básica de estos módulos se muestra en la figura 3.1.

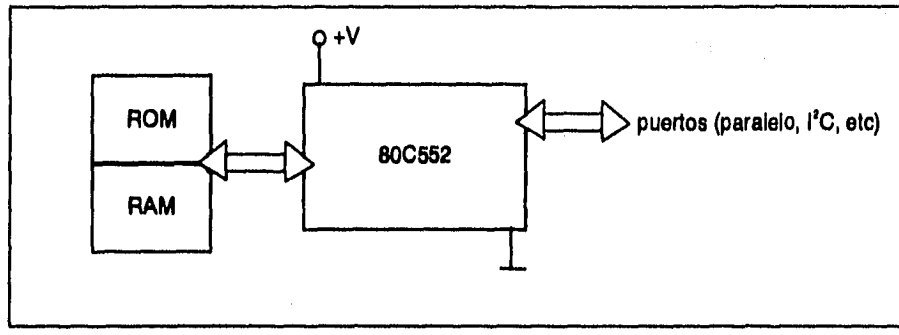


Figura 3.1. Configuración básica con el 80552.

El 80C552 es un microcontrolador de 8 *bits* de la familia del 80C51. Además de las funciones propias del 8051, este *chip* tiene 256 *bytes* de RAM, cinco puertos bidireccionales de 8 *bits*, un convertidor A / D de 10 *bits* y una salida de modulación por ancho de pulso (PWM). Los registros involucrados en la transmisión a través de la interfaz I²C , así como en la conversión A / D se describen en detalle como parte del *software*.

Para incrementar las capacidades del dispositivo se conectarán dos memorias, la Intel 2716 y la Hyundai 6264, cuyas especificaciones se incluyen en el apéndice A. Esto permitirá almacenar permanentemente el programa del microcontrolador, así como

tener una mayor capacidad de almacenamiento de información proveniente de los sensores.

Todos estos dispositivos tienen polarización de 5 volts, compatible con los voltajes TTL manejados en el resto del sistema y en el mismo *bus*. El direccionamiento extendido se realiza a través de los puertos 0 y 2 , mientras que los datos llegan a través del puerto 0. El puerto 5 contiene a los 8 canales multiplexados del convertidor A / D, cuya referencia se define en dos terminales más del *chip*. El oscilador que permite la operación del microcontrolador requiere la inclusión de un cristal de cuarzo, y la frecuencia de operación se determina a través de uno de los registros. Los diagramas electrónicos de esta configuración forman parte del apéndice B.

La medición de la temperatura se realiza a través de un sensor monolítico para grados Celsius, el LM35 de National Semiconductors, cuyas especificaciones se incluyen en el apéndice A. Este dispositivo se polariza con 5 volts, al igual que el resto de los componentes de este proyecto, y entrega a su salida un voltaje de 10 mV /°C. Dicha salida se alimenta directamente a uno de los canales de conversión del 80C552.

Para la medición de iluminación se utilizará un fotorresistor, como se muestra en la figura 3.2. Al utilizar este dispositivo como parte de un divisor de voltaje, se tendrá a la salida un voltaje proporcional a la luz incidente sobre la celda. Este voltaje se alimenta a un segundo canal del convertidor A/D, brindando la medición deseada.

La medición de los demás parámetros será simulada con el fin de efectuar pruebas en el sistema. La detección de presencia se simulará con un simple interruptor, y la medición de humedad y calidad del aire se simularán generando una señal de voltaje variable a través de potenciómetros.

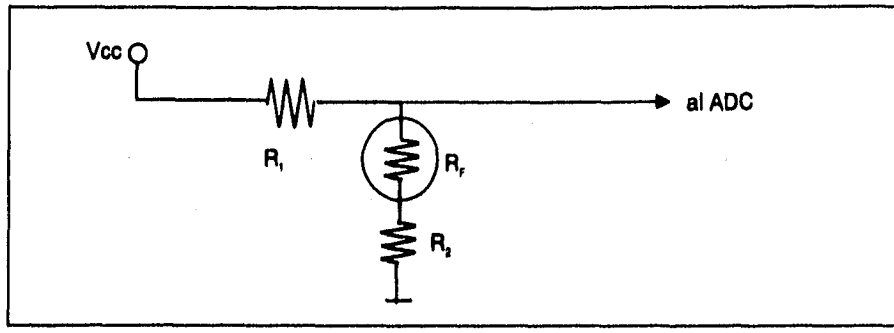


Figura 3.2. Fotocelda.

Con el fin de concentrar todos los transductores se construye una tarjeta de simulación / medición, cuyo diagrama se muestra en la figura 3.3 y cuyos diagramas electrónicos se incluyen en el apéndice B.

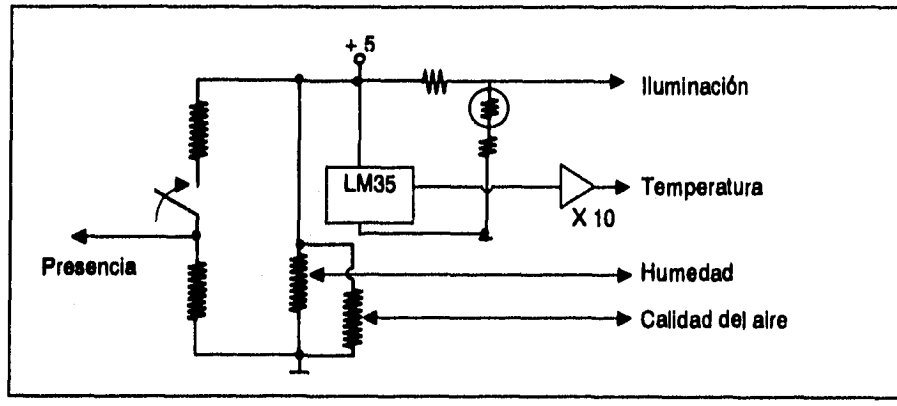


Figura 3.3. Módulo de simulación / medición.

Otro de los aspectos importantes de estas mediciones es la adecuación de las señales a una misma escala. Esto es necesario debido a que el convertidor A / D tiene solamente una referencia y no puede variar arbitrariamente. Considerando que el voltaje a través del sistema es de 5 volts, es lógico utilizar esa misma diferencia de potencial como plena escala de conversión. Al amplificar con un factor de 10 la señal proveniente del LM35, se tendrán lecturas de hasta 50 grados centígrados. Las demás señales pueden adecuarse fácilmente a esos niveles de voltaje.

La configuración del hardware de los módulos no es compleja debido a que la mayoría de las operaciones se realizan dentro del microcontrolador elegido. Dependiendo de la aplicación, deberán tomarse las consideraciones pertinentes al agregar sensores a los módulos.

Con esto se tiene diseñada una parte importante del sistema global, como se puede ver en la figura 3.4.

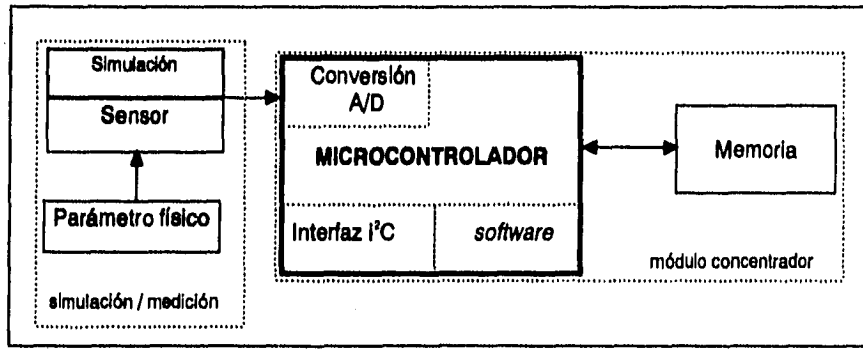


Figura 3.4. Diagrama a bloques.

3.1.2. DISEÑO DE LA INTERFAZ CON LA COMPUTADORA

El conjunto de ventajas y desventajas ofrecido por el módulo externo permiten la construcción de un sistema interplataforma de buena velocidad y complejidad razonable que enlaza un número grande de módulos para la transmisión e interpretación de datos. Para la etapa de diseño es necesario analizar cada una de las partes involucradas en el proceso.

- PUERTO PARALELO

El puerto paralelo es utilizado por la mayoría de las computadoras personales para enlazarse con una impresora a través de un cable Centronics paralelo. El uso de este puerto permite que el diseño no sea restrictivo de una sola plataforma. A través de este puerto se transmitirá la información a la interfaz para su posterior emisión con formato I^2C. Esto significa que se requieren ocho líneas que extraigan información de la computadora, como se muestra en la figura 3.5.

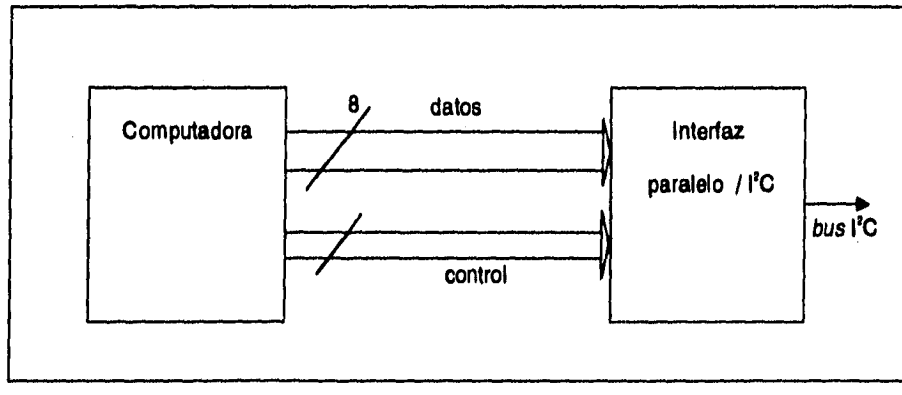


Figura 3.5. Transmisión.

Puesto que el bus I^2C es bidireccional, también se requerirán ocho líneas que introduzcan información a la computadora, como se muestra en la figura 3.6. Adicionalmente, debe contemplarse la necesidad de líneas que controlen el intercambio de información, regulando la interacción de la interfaz con la computadora.

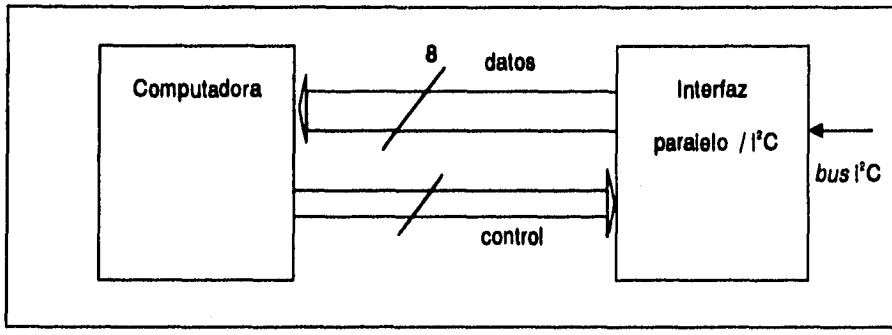


Figura 3.6. Recepción.

La manera más intuitiva para lograr estas dos condiciones es utilizar un puerto paralelo bidireccional que permita la transmisión y recepción de datos por las mismas líneas, simplificando el diseño del módulo enormemente (figura 3.7).

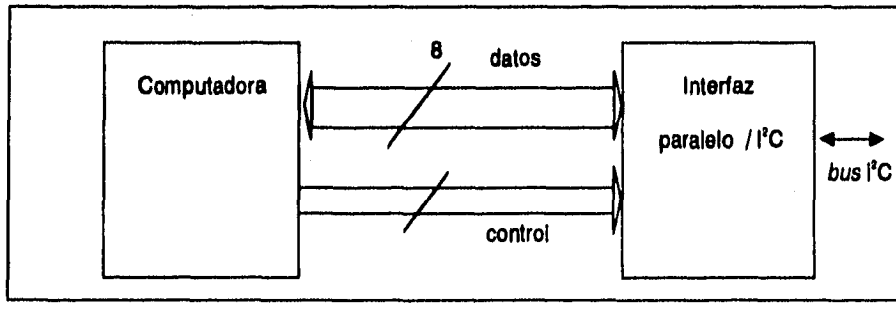


Figura 3.7. Conexión a través de un puerto paralelo bidireccional.

Lamentablemente, este tipo de puerto paralelo no es muy comercial. Fue creado originalmente por IBM para sus computadoras PS/2, y durante algún tiempo pretendió reemplazar a los puertos unidireccionales. Sin embargo, la enorme difusión de estos últimos hacen del puerto bidireccional un equipo extraordinariamente poco común. Es por esto que es necesario explorar la conexión a través de un puerto unidireccional.

El uso generalizado del puerto paralelo es el control de una impresora. Debido a esto, las líneas tienen denominaciones que se refieren a operaciones de impresión. En el caso de una IBM-PC, estas líneas salen a través de un conector DB-25, como el mostrado en la figura 3.8.

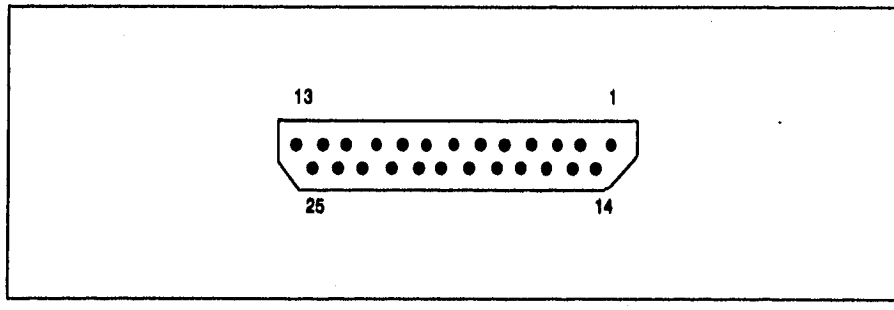


Figura 3.8. Conector DB-25.

Las funciones de cada una de las líneas disponibles en este conector se detalla en la tabla 3.1.

Número de terminal	Nombre	Tipo	Dirección
1	Strobe	Control	Salida-Colector Abierto
2	Data *Bit* 0	Datos	Salida
3	Data *Bit* 1	Datos	Salida
4	Data *Bit* 2	Datos	Salida
5	Data *Bit* 3	Datos	Salida
6	Data *Bit* 4	Datos	Salida
7	Data *Bit* 5	Datos	Salida
8	Data *Bit* 6	Datos	Salida
9	Data *Bit* 7	Datos	Salida
10	Acknowledge	Control	Entrada-Colector Abierto
11	Busy	Control	Entrada-Colector Abierto
12	Paper End	Control	Entrada-Colector Abierto
13	Select	Control	Entrada-Colector Abierto
14	Auto Feed	Control	Salida-Colector Abierto
15	Error	Control	Entrada-Colector Abierto
16	Initialize Printer	Control	Salida-Colector Abierto
17	Select Input	Control	Salida-Colector Abierto
18-25	Ground		

Tabla 3.1 Descripción del puerto paralelo de una IBM-PC. (continuación)

Un somero análisis de las características de este puerto permite observar que solamente existen cinco líneas de entrada. Sin embargo, gracias a que varias de las salidas tienen características de colector abierto, pueden utilizarse como entradas.

El procedimiento de transmitir un *byte* es muy simple. Se escribe el dato a una dirección de memoria que está asociada con las terminales 2-9 del puerto paralelo.

Inmediatamente se presenta la información en el conector DB-25 a través de las líneas de datos. El procedimiento de lectura de las terminales de entrada es muy similar, pues solamente es necesario consultar una dirección predeterminada de memoria. Sin embargo, la consulta de la información que entra por las salidas colector abierto es ligeramente más compleja. Es necesario colocar todas esas salidas en un nivel eléctrico alto, y posteriormente consultar su estado. Si la terminal recibe un nivel eléctrico bajo, su estado cambiará debido a su comportamiento de AND-alambrada

La tabla 3.2 muestra las direcciones de memoria asociadas con la transmisión y recepción de información a través del puerto paralelo. El símbolo "$" denota un número hexadecimal.

Dirección		b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
$378	nombre	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
	pin	9	8	7	6	5	4	3	2
$379	nombre	BUSY	ACK	P. END	SEL	ERR			
	pin	11	10	12	13	15			
$37A	nombre					S.INPUT	INIT	A.F.	STRB
	pin					17	16	14	1

Tabla 3.2. Direcciones de memoria asociadas al puerto paralelo de la IBM-PC.

- INTERFAZ BIDIRECCIONAL DE 8 *BITS*

Para conjuntar estas características de transmisión, y en vista de que la velocidad de operación no es muy grande, se propone el diseño mostrado en la figura 3.9.

Si la línea de $\overline{\text{STROBE}}$ se coloca en un nivel alto, puede transmitirse del puerto paralelo al exterior. Los *bits* de datos pasan a través del *latch* "A" directamente a la salida de datos. El *latch* "B" presenta una alta impedancia, evitando que los datos regresen. Los *latches* "C" y "D" se encargan de dirigir las líneas de control provenientes de las líneas $\overline{\text{INIT}}$, $\overline{\text{AUTO FEED}}$ y $\overline{\text{SELECT INPUT}}$ a las tres salidas de control. El *latch* "E" se encarga de evitar conflictos entre los datos y las líneas de control. Dependiendo de las operaciones requeridas, estas salidas pueden utilizarse como líneas de selección de un decodificador, como se verá posteriormente.

Para la recepción, la señal de $\overline{\text{STROBE}}$ será baja, con lo que los *latches* "A" y "D" quedarán en estado de alta impedancia. Los datos pasarán entonces a través de los *latches* "B" y "C", llegando al puerto paralelo. Para las señales de control se utilizan directamente tres líneas de datos del puerto paralelo. Estos *bits* quedan aislados de los datos entrantes por los *latches* A y D.

Para la transmisión se utilizan aislamientos en once líneas, mientras que para la recepción se utilizan quince. Para esto será necesario utilizar *latches* capaces de manejar 16 líneas simultáneamente, efecto que se logra al operar dos *latches* octales sincronizadamente. Los dispositivos empleados son los 74LS373 de Motorola, dispositivos que incorporan en un solo *chip* ocho *latches* con una sola línea de selección. De esta forma, las líneas que pasan por "A" y "D" se concentran en dos chips, mientras que las que pasan por "B", "C" y "E" pasan por otros dos.

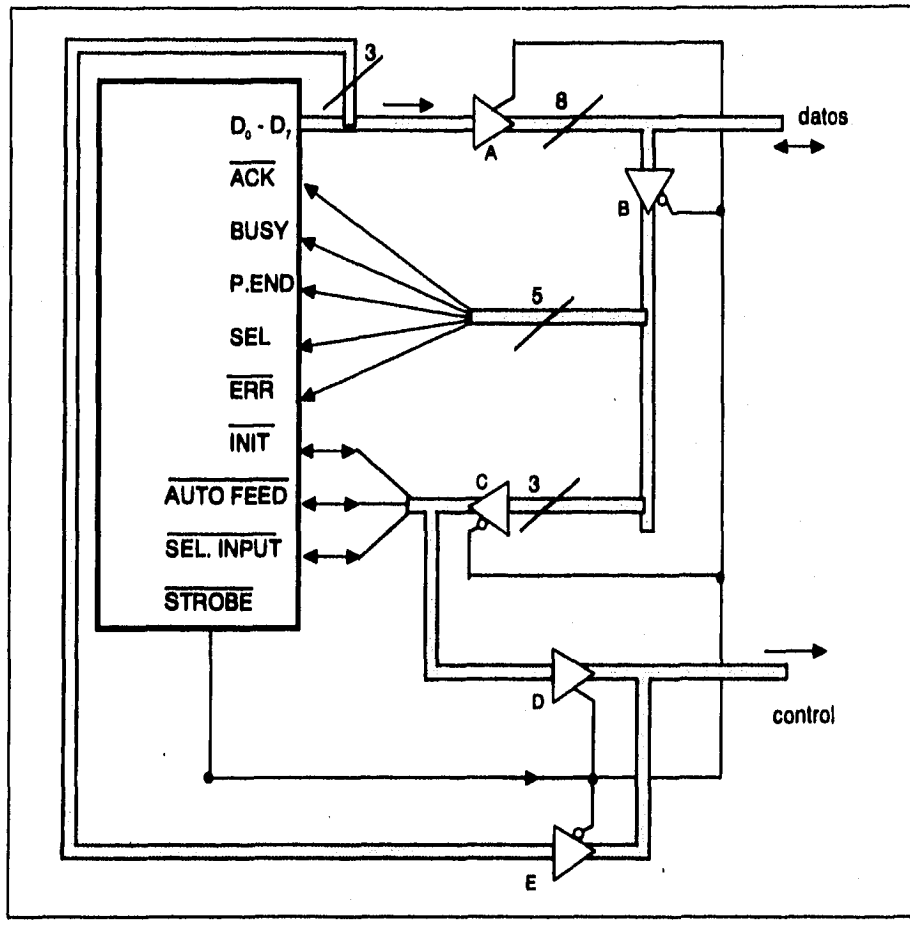


Figura 3.9. Diseño de interfaz bidireccional de 8 *bits* para puerto paralelo.

Para habilitar los circuitos integrados deseados al mismo tiempo que se desactivan los demás, es necesario contar con la señal de $\overline{\text{STROBE}}$ tanto directamente como negada. Esto se puede lograr con cualquier inversor TTL comercial. En la construcción del prototipo se utilizó, debido a su fácil consecución, una compuerta NAND 74132 con las entradas cortocircuitadas.

Gracias a que el puerto paralelo utiliza niveles compatibles con TTL, los circuitos de implementación no son costosos. La familia específica de TTL no es importante dado que el módulo se energizará con la alimentación comercial y la velocidad de operación no es muy elevada. En la construcción del prototipo se utilizaron componentes de la familia LS (Low Power Schottky), mas puede emplearse cualquier familia pues el consumo de energía no es una consideración primordial.

Para hacer esta interfaz más versátil, se construyó en una tarjeta independiente de la interfaz I^2C, de modo que cuenta con un bus hacia el puerto paralelo y otro hacia la aplicación, como muestra la figura 3.10. Esto permite su operación independiente como sistema interplataforma para enlace bidireccional de 8 *bits*.

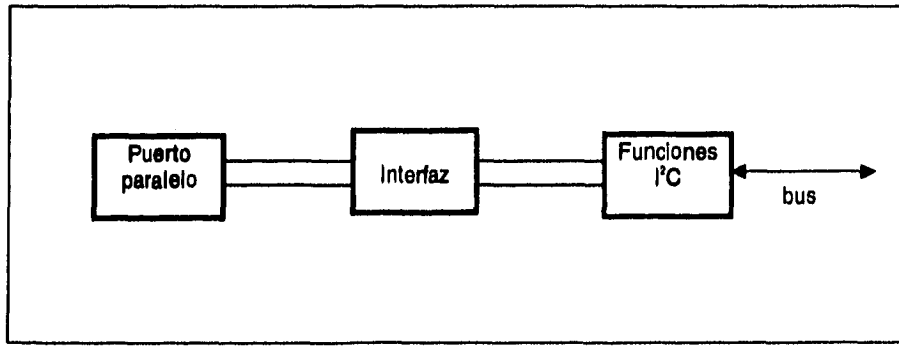


Figura 3.10. Interfaz bidireccional de 8 *bits*.

- INTERFAZ I²C

La función de esta interfaz es recibir los datos del puerto paralelo a través del subsistema antes descrito y codificarla en formato I²C para su transmisión, así como el proceso inverso cuando la información llega por el *bus* a la computadora.

Philips diseñó para esta operación el microcontrolador de propósito específico PCF8584, denominado "controlador de *bus* I²C". Este *chip* recibe información en formato de 8 *bits* y, a través de programación, codifica, envía y recibe datos con formato y protocolo I²C. Tiene la capacidad de actuar en un sistema multimaestro, puede trabajar directamente en el *bus* de datos de un procesador 80X86 (PC), 680X0 (Macintosh) o bien con microprocesadores como el 8051, Z80, etc. Sus especificaciones completas se incluyen en el apéndice A.

Para su funcionamiento adecuado, este sistema requiere polarización, una base de tiempo y varias líneas de control, como se muestra en la figura 3.11.

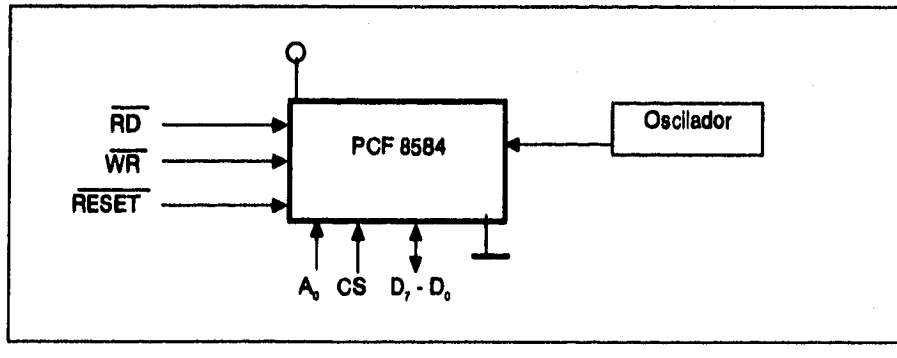


Figura 3.11. Configuración básica del PCF8584.

La polarización de este microcontrolador se realiza con 5 volts de corriente directa. Puesto que es la misma polarización que requiere la interfaz bidireccional, el diseño emplea una sola fuente de alimentación para ambas tarjetas. A continuación se detalla su diseño.

La primera consideración en el diseño de la fuente es la potencia que el circuito ha de consumir. Debido a que todos los dispositivos que integran el presente proyecto operan con 5 volts, resulta más simple considerar las conrrientes como parámetro de diseño. Diseñando para un caso crítico, pueden tomarse los valores máximos de consumo de corriente proporcionadas en las especificaciones de los circuitos inegrados.

En el caso de la interfaz bidireccional, se tiene un requerimiento máximo de 244 mA, considerando todos los componentes TTL y el microcontrolador PCF8584. Por su parte, el circuito del microcontrolador 80C552 consume 310 mA, además de la corriente requerida por los sensores. Al utilizar el regulador LM7805, se puede alimentar una carga de 1 A con una regulación bastante aceptable para nuestros propósitos, ya que proporciona un voltaje de rizo de 50 mV máximo.

Una vez planteados los lineamientos, puede procederse al diseño propiamente. La primera etapa está constituida por un fusible rápido de 500 mA y un transformador de voltaje. La relación de transformación utilizada es de 110 / 12, lo cual arroja una corriente máxima en el primario de 110 mA_{rms}. El fusible deberá soportar 154 mA como máximo, por lo que deberá utilizarse un fusible comercial de valor igual o inferior al especificado. El utilizar un fusible de mayor calibre permitiría que el equipo consumiera más de 1 A, resultando en daños posiblemente permanentes.

Para convertir la señal de AC en una señal pulsante de DC, se rectifica la señal utilizando un puente de diodos. Se emplea un puente cuyas especificaciones indiquen un máximo de corriente de 1 A, o bien puede construirse a partir de diodos de propósito general como el 1N4001, resultando en una rectificación más homogénea.

La siguiente etapa realiza la labor de filtrado, donde se eliminan altas frecuencias (ruido) y se filtran también los armónicos propios de la señal rectificada. Para esto se utilizó un capacitor de 0.1 μF, recomendado para el uso de reguladores, y uno de 2200 μF en paralelo. El cálculo del rizo en función de la capacitancia se desprecia debido a lo expuesto en el siguiente párrafo.

Se utilizó un regulador de voltaje LM7805, ya que proporciona la regulación de voltaje adecuada conduciendo corrientes de hasta 1 A. Debido a que el voltaje a su entrada es de 19 V_{cc} con un rizo del orden de milivolts, el regulador opera dentro del intervalo de condiciones recomendado para su correcta operación. Para asegurar aún más la continuidad del voltaje, puede colocarse otro capacitor de 2200 μF a la salida del regulador y uno de 0.1 μF para eliminar el ruido de alta frecuencia que pueda presentarse. El diseño completo de la alimentación se muestra en la figura 3.12.

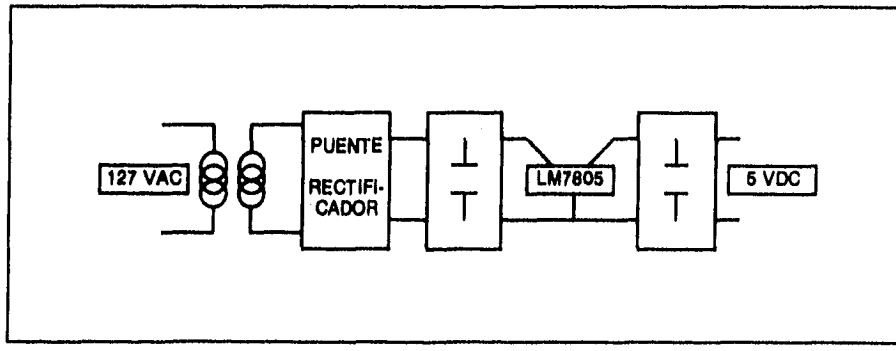


Figura 3.12. Diseño de la alimentación.

A pesar de la precisión obtenida con este diseño, resulta más práctico utilizar un eliminador de baterías comercial como etapa preliminar. De esta forma se obtiene un sistema modular de tamaño y peso reducidos. Adicionalmente, la disponibilidad de un eliminador de baterías puede considerarse universal. Debido a que estos dispositivos entregan voltajes de directa no muy estables, su salida se hace pasar por una etapa de regulación como se muestra en la figura 3.13 a continuación.

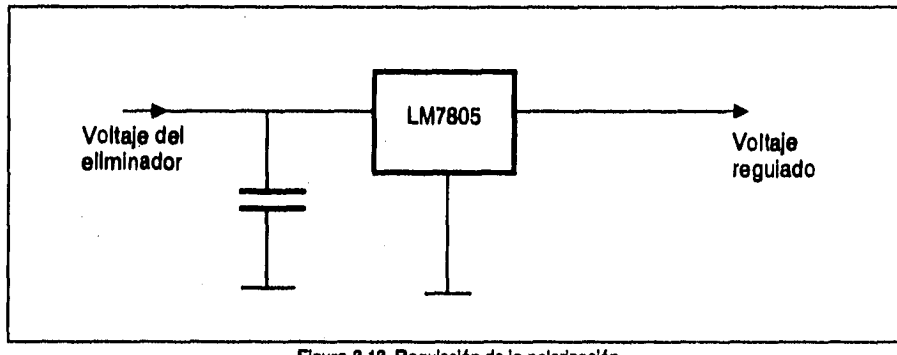


Figura 3.13. Regulación de la polarización.

El voltaje que entrega el eliminador pasa por un capacitor que lo mantiene aproximadamente constante debido a su gran capacitancia. Una vez logrado este voltaje continuo se alimenta al regulador integrado LM7805, dispositivo que entrega un voltaje continuo regulado de 5 volts. Pueden incluirse capacitores de 0.1 μF a la salida, reduciendo aún más el rizo de voltaje según las especificaciones correspondientes.

Debido a que el *bus* I²C opera con información en serie, el PCF8584 requiere una base de tiempo estable. Dicha base, según sus especificaciones, puede ser desde 3 hasta 12 MHz, dependiendo de la velocidad de operación deseada en el *bus*. Es importante destacar que la frecuencia del oscilador no es igual a la del *bus*, pues esta última es la división de la primera para maximizar su estabilidad.

Para obtener la máxima velocidad de transmisión en el *bus* se utiliza una base de tiempo de 12 MHz. Esta se consigue con un oscilador de cuarzo, que puede armarse a base de un cristal y una compuerta CMOS, o comprarse en un dispositivo integrado. Estos integrados son muy comerciales y de bajo costo, por lo que representan la solución ideal. Se polarizan con 5 V y ofrecen la estabilidad de oscilación de un cristal. La figura 3.14 muestra su diagrama de conexión.

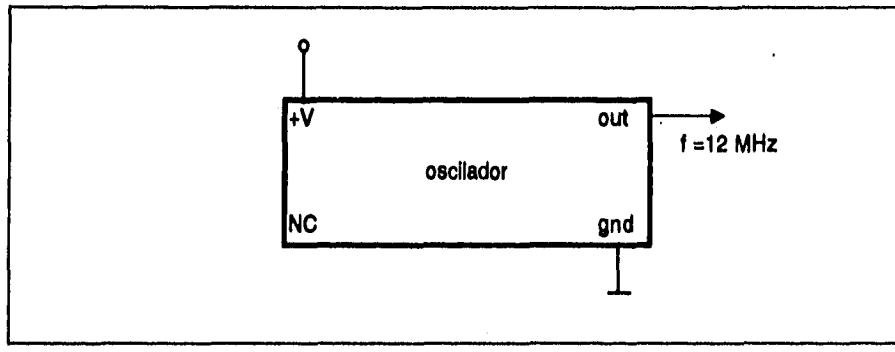


Figura 3.14. Oscilador de cuarzo integrado.

Como se ve, la construcción de esta tarjeta es realmente simple, pues el PCF8584 tiene en su interior todo lo necesario para la conversión a I²C. La complejidad de operación se refleja en el *software*.

- INTEGRACIÓN DE LA INTERFAZ CON LA COMPUTADORA

Una vez establecida la comunicación bidireccional a través del puerto paralelo, y contando con la interfaz I²C en una tarjeta adicional, es necesario establecer cómo llegarán los datos y las instrucciones de control de un módulo a otro. La manera en que se realizará este intercambio estará dictada por los requerimientos de operación del PCF8584.

Debido a su velocidad de operación relativamente baja, el sistema operará por encuesta, es decir sin interrupciones. Además, puesto que el único elemento maestro será la computadora (a través del equipo necesario) toda actividad en el *bus* será coordinada por el PCF8584.

Las líneas de control que se requieren para esta aplicación son $\overline{RD}$ (lectura), $\overline{WR}$ (escritura), A_0 (utilizada para programación) y $\overline{RESET}$. Aquí se presenta una dificultad: es necesario controlar 4 líneas distintas cuando la interfaz bidireccional brinda solo tres. La solución a este problema es utilizar un decodificador.

Un decodificador como el 74LS138 es capaz de encender cuatro líneas en su salida a partir de dos en su entrada. Sin embargo, no es capaz de activar más de una a la vez. Debido a que la línea A_0 se utiliza para la programación del microcontrolador, será frecuentemente utilizada en conjunción con operaciones de lectura y escritura. Sin embargo, nunca se harán operaciones de lectura y escritura simultáneas, ni combinaciones que incluyan la línea de *RESET*. Considerando lo anterior, dos de las líneas de la interfaz se llevarán a un decodificador de 2 X 4 y la tercera controlará directamente la línea A_0. Esto se muestra en la figura 3.15.

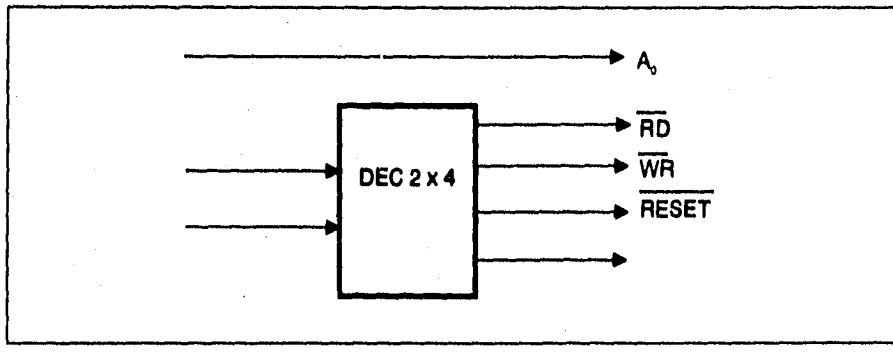


Figura 3.15. Configuración de las líneas de control.

Al lograr las conexiones descritas a lo largo de estos apartados se obtiene total control de la operación del PCF8584 para controlar el *bus* I²C, como se muestra en la figura 3.16. En la tabla 3.3 se resumen las direcciones de memoria que actúan directamente sobre el microcontrolador. Los detalles de estas direcciones, su obtención y procedimiento de lectura se detallan bajo el análisis de *software*. El signo "$" denota números hexadecimales.

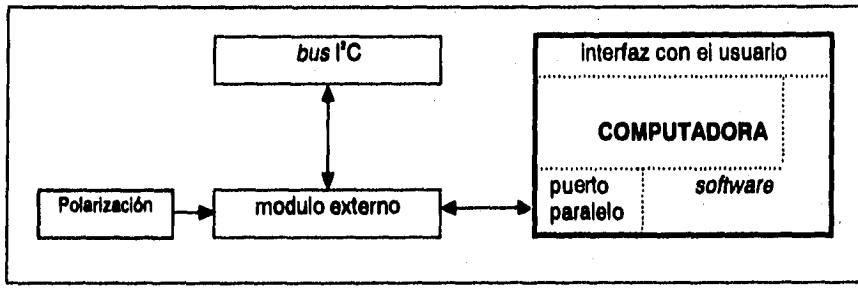


Figura 3.16. Diagrama de bloques.

	Transmisión		Recepción		
	Datos	Control	Datos		Control
Dirección	$378	$37A	$379	$37A	$378
Bits	$b_7 - b_0$	$b_3 - b_1$	$b_7 - b_3$	$b_3 - b_1$	$b_3 - b_1$

Tabla 3.3. Direcciones de memoria relacionadas con el PCF8584.

3.2. *Software*

A través del conjunto de instrucciones que se detalla a continuación, el sistema recabará la información, la convertirá al formato I^2C y la transmitirá hasta la computadora personal donde será interpretada. Cada una de estas tareas se lleva a cabo con una metodología específicamente diseñada para el *hardware* directamente involucrado

3.2.1. MÓDULOS

El microcontrolador 80C552 constituye la única parte inteligente de los módulos de adquisición de datos, por lo que es la única susceptible de programación por software. El proceso que realiza al recibir la información transducida, convertirla a información digital y posteriormente transmitirla a través del *bus* se consigue con una serie de pasos predefinidos que se discuten a continuación.

CONTROL DE CONVERSIÓN A/D

La conversión realizada por el microcontrolador se regula a través de dos registros de función específica, ubicados en las direcciones $C6 y $C7 de la memoria RAM interna. El registro ADCON, mostrado en la tabla 3.4, contiene los *bits* de control necesarios para la correcta conversión de los datos.

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
ADC.1	ADC.0	ADEX	ADCI	ADCS	ADDR2	ADDR1	ADDR0

Tabla 3.4. Registro ADCON.

Los tres *bits* menos significativos determinan el canal que será utilizado en la conversión A/D. El *bit* 3, ADCS, marca el inicio de una nueva conversión si el *bit* 4, ADCI, está en "0". La interrupción del convertidor, ADCI, indica que se terminó una conversión cuyo resultado puede leerse del registro ADCH y los dos *bits* más significativos del registro ADCON. En caso de desear la conversión de 8 *bits*, como en el caso de la presente aplicación, simplemente se utiliza el resultado del registro ADCH. Por último, el *bit* ADEX indica al microcontrolador que el inicio de conversión será determinado por software y nunca de forma externa. El registro ADCH se muestra en la tabla 3.5.

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
ADC.9	ADC.8	ADC.7	ADC.6	ADC.5	ADC.4	ADC.3	ADC.2

Tabla 3.4. Registro ADCON

El programa del microcontrolador, cuyo listado se incluye en el apéndice C, tiene la siguiente secuencia:

1. Indicar inicio de conversión interna (ADEX=0).
2. Seleccionar el canal de conversión. (ADDR 0-2) .El programa realiza la conversión de cada uno de los cinco canales empleados en el prototipo para posteriormente transmitir los datos por el *bus*.
3. Iniciar la conversión del canal seleccionado. (ADCS=1).
4. Para saber si la conversión terminó, después de 50 ciclos de máquina, se consulta el *bit* de interrupción. (ADCI)
5. Cuando ADCI es "1", se extrae el *byte* del registro ADCH y se escribe en RAM para su posterior transmisión.

6. Hecho esto, se llevan los *bytes* ADCS y ADCI a un nivel bajo para permitir una nueva conversión.

ALMACENAMIENTO Y TRANSMISIÓN DE INFORMACIÓN

Los datos digitalizados con el convertidor se almacenan en el instante oportuno en la memoria RAM. Con el objeto de minimizar las transmisones de la dirección I²C, dicho instante es después de cada serie de conversiones y no después de cada una de ellas.

El resultado de la conversión se almacena en el registro ADCH, de donde se extrae a la memoria RAM. Después se traslada de la RAM al registro S1DAT, ubicado en la dirección $DA y punto de partida para la transmisión I²C.

El formato de la transmisión, así como otras características relacionadas con el *bus*, se define a través de los registros descritos en la tabla 3.5.

Registro	Dirección	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
S1ADDR	$DB	Dirección esclava I²C							GC
S1DAT	$DA	Datos recibidos / Transmitidos por el *bus* i2c							
S1STA	$D9	SC4	SC3	SC2	SC1	SC0	0	0	0
S1CON	$D8	CR2	ENS1	STA	STO	SI	AA	CR1	CR0

Tabla 3.5. Registros relacionados con el *bus* I²C.

El registro S1STA reporta condiciones cuyas claves se encuentran en la tabla 3.6.

CLAVE	CONDICIÓN
$A8 ó $B0	Se recibió la dirección esclava, el *bit* de dirección indica transmisión y se envió el *bit* de *Acknowledge*. Puede procederse con la transmisión si el *bit* AA en S1CON está en nivel alto.
$B8	Se transmitió el *byte* contenido en S1DAT, se recibió *Acknowledge* y puede continuarse con la transmisión si AA=1
$C0	Se transmitió el *byte* contenido en S1DAT, pero NO se recibió *Acknowledge*. Se retransmitirá si STA y AA están en nivel alto.
$C8	Se transmitió el último *byte* desde S1DAT, y AA=0, por lo que la transmisión termina.

Tabla 3.6. Claves de estado de la interfaz I²C.

El registro S1ADDR aloja la dirección esclava I^2C del microcontrolador. Al recibir este *byte* a través del *bus*, el 80C552 entra al modo transmisor esclavo, pues éste será el único útil para el desarrollo del proyecto. Después de establecida la comunicación, cada *byte* de información es seguido de la transmisión de *Acknowledge*.
El *bit* menos significativo, GC, habilita la respuesta del microcontrolador a un "llamado general" en el *bus* I^2C, es decir, la transmisión de la dirección $00.

El registro S1DAT es la dirección de memoria de donde partirán los datos hacia la computadora. El microcontrolador puede leer y escribir directamente a este registro, cuya información corre siempre de derecha a izquierda. Esto significa que el primer *bit* recibido o transmitido será el más significativo del *byte*.

Los *bits* que componen el registro de control S1CON, ubicado en la dirección $D8 , tienen las funciones descritas en la tabla 3.7.

S I:	Cuando este *bit* está en "1", se solicita una interrupción en el puerto serie, es decir, la suspensión de la transmisión en la línea SCL.
ENS1:	Habilita la operación de la interfaz I²C.
STO:	Este *bit* se coloca en "0" cuando se verifica una condición de *Stop* en el *bus* I²C. También se coloca en "0" cuando la interfaz no está habilitada (ENS1=0).
STA	Este *bit*, en modo esclavo, registra la presencia de una condición de *Start* en el *bus* I²C.
AA	Al colocar un "1" en este registro, la generación del *bit* de *Acknowledge* es automática al recibir la dirección propia o un *byte* de datos completo. Para interrumpir la transmisión puede colocarse en "0".
CR2 CR1 CR0	Estos *bits* determinan la frecuencia de operación del *bus*. La relación entre los *bits* y la frecuencia se muestra en la tabla 3.7.

Tabla 3.7. Registro S1CON.

La tabla 3.8 muestra los *bits* que determinan la velocidad de transmisión en el *bus* en función de la frecuencia del reloj del microcontrolador. Es importante destacar que la frecuencia determinada por estos registros no tiene injerencia alguna sobre la velocidad del *bus* cuando el dispositivo opera como esclavo.

Los pasos a seguir para la transmisión de los datos desde el módulo hasta la computadora son los siguientes:

1. Cargar la dirección esclava en el registro S1ADDR. Para esta aplicación se utilizará la dirección $AA. El estado del *bit* GC no es importante para este proyecto.
2. Habilitar la operación de la interfaz I²C con el *bit* ENS1 en nivel alto. Los *bits* AA, STA, STO y SI deben se colocan en "0". Esto termina la programación inicial de la interfaz.

CR2	CR1	CR0	Frecuencia del *bus* (kHz) (en función de la frecuencia del μC)		
			6 MHz	12 MHz	16 MHz
0	0	0	23	47	63
0	0	1	27	54	71
0	1	0	31	63	83
0	1	1	37	75	100
1	0	0	6.25	12.5	17
1	0	1	50	100	-
1	1	0	100	-	-

Tabla 3.8. Frecuencias de operación.

3. Cuando el microcontrolador recibe su dirección esclava seguida del *bit* de dirección ("1" para transmisión) entra al modo esclavo transmisor y envía el *bit* de *Acknowledge* al maestro de la transmisión. Posteriormente reporta los resultados en el registro S1STA.
4. Cuando recibe el código $A8, $B0 ó $B8, el sistema está listo para cargar un nuevo *byte* en el registro S1DAT. Si se desea continuar la transmisión, se coloca el *bit* AA en "1"; si se desea interrumpirla, en "0".

3.2.2. SISTEMA DE ADQUISICIÓN DE DATOS

El *software* del sistema controlará los elementos inteligentes que lo conforman. Puesto que la interfaz bidireccional del puerto paralelo no es más que circuitos lógicos combinacionales, no requiere *software* de ninguna clase. Aquellos que sí lo requieren son el microcontrolador del *bus* y la computadora personal.

- CONTROLADOR DEL *BUS* I^2C

Este controlador, constituido por el PCF8584, es programado a través de la interfaz bidireccional de 8 *bits* para el puerto paralelo. Todas las instrucciones de programación forman parte del programa de adquisición de datos desarrollado para la computadora personal en lenguaje *Turbo Pascal*. En realidad no existe *software* específico para este microcontrolador puesto que no tiene memoria. Su programación se reduce a modificar valores de *bits* determinados en un registro específico. Esta característica es el resultado de que la conversión al protocolo I^2C no depende del usuario, sino que es inherente a la función del *chip*.

El registro de control "S1" del PCF8584 consta de 8 *bits* que tienen funciones distintas en la lectura y escritura, como se describe en las tablas 3.9 y 3.10.

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Escritura (control)	PIN	ESO	ES1	ES2	ENI	STA	STO	ACK
Lectura (estado)	PIN	0	STS	BER	AD0/ LRB	AAS	LAB	BB

Tabla 3.9. Registro de control "S1".

La descripción detallada de cada una de estas funciones se incluye en las especificaciones del apéndice A.

Función	Descripción
PIN	Indica que se ha recibido un *byte* completo
ESO	Habilita la salida serie I^2C
ES1,ES2	Direccionan otros registros
ENI	Habilita la operación con interrupciones
STA	Controla la generación del *bit* de *Start*
STO	Controla la generación del *bit* de *Stop*
ACK	Controla la generación del *bit* de *Acknowledge*
STS	Útil en recepción (esclavo) exclusivamente
BER	Error en el bus
AD0 / LRB	AD0 se utiliza en modo esclavo. LRB recibe el *bit* de *Acknowledge*
AAS	Direccionado como esclavo
LAB	Indica colisiones en el *bus*
BB	Indica que el *bus* está ocupado

Tabla 3.10. *Bits* de control del registro S1.

Los demás registros, cuyo acceso se logra a través de la programación de los *bits* ES1 y ES2 en conjunción con la línea A_0 en nivel bajo, alojan un vector de interrupciones, la dirección esclava del microcontrolador, y la frecuencia de operación del *bus*, como se describe en la tabla 3.11.

Tanto el registro de interrupciones como el de la dirección esclava resultan innecesarios para el desarrollo de esta aplicación. Sin embargo, el microcontrolador no puede funcionar sin dirección propia, por lo que en su momento se le programará una dirección arbitraria distinta a la de los módulos.

ES1	ES2	Registro direccionado
0	0	dirección esclava
0	1	interrupciones
1	0	registro de reloj

Tabla 3.11. Registros adicionales.

El registro de reloj aloja información de la frecuencia del oscilador y la frecuencia del *bus* I²C de la forma mostrada en la tabla 3.12.

b_4	b_3	b_2	Frecuencia del oscilador	b_1	b_0	Frecuencia del *bus*
0	X	X	3 MHz	0	0	90 kHz
1	0	0	4.43 MHz	0	1	45 kHz
1	0	1	6 MHz	1	0	11 kHz
1	1	0	8 MHz	1	1	1.5 kHz
1	1	1	12 MHz			

Tabla 3.12. Registro del reloj "S2".

Como se puede ver, la frecuencia del oscilador se registra en los *bits* 2 a 4 del registro, mientras que el divisor de frecuencia utilizado para el *bus* se establece con los *bits* 0 y 1.

La programación inicial del microcontrolador se lleva a cabo a través de *software* desarrollado para la computadora personal, mismo que se describe detalladamente a continuación.

- COMPUTADORA PERSONAL

Como se explicó anteriormente, el *software* involucrado en la adquisición y tratamiento de la información es distinto. El encargado de la adquisición de datos controla el PCF8584, la transmisión a través del *bus* I^2C y el almacenamiento de los datos en diversos archivos. El programa que interpreta la información se basa en dichos archivos para elaborar gráficas, tablas y pantallas de monitoreo. A continuación se describe cada uno de estos módulos de programación.

ADQUISICIÓN DE DATOS

Para recibir la información proveniente del *bus* I^2C es necesario coordinar las acciones del PCF8584 con la computadora, efecto conseguido a través de las líneas de control del dispositivo.

La interfaz bidireccional requiere una señal que indique si funcionará como transmisor o receptor, pues esto definirá el conjunto de *latches* que conducirá la información. Dicha señal se obtiene a través de la línea de $\overline{STROBE}$ del puerto paralelo, y forma parte medular de todas las rutinas de programación, por lo que se programó un procedimiento llamado **STROBE** que recibe como parámetro un "1" si se desea usar el *hardware* de transmisión o un "0" si se desea el de recepción. Con base en este parámetro, el procedimiento determina el estado del *bit* 0 en la palabra de

control ubicada en $37A. Para no alterar el resto de la palabra de control, se lee el contenido, se ajusta el *bit* de interés y se reemplaza la información en la dirección $37A.

El procedimiento **A0** se encarga de activar o desactivar la línea A_0, requerida para la programación del microcontrolador. Como con todos los procedimientos, es necesario indicar si la operación se realizará como parte de una transmisión o de una recepción de información. Con base en esta información se llama al procedimiento STROBE y posteriormente se activa la línea de control correspondiente a A_0 a través de los *bits* adecuados según la información concentrada en la tabla 3.13.

Puesto que la palabra de control tiene injerencia sobre varias líneas simultáneamente, es indispensable garantizar que solamente el *bit* deseado cambiará. Como se describió en el caso de STROBE, la dirección involucrada se leerá, se modificará el *bit* requerido y posteriormente se regresará a la dirección de memoria.

	Transmisión			Recepción		
Dirección	$37A			$378		
	b_2	b_1	b_0	b_2	b_1	b_0
A_0	1	X	X	1	X	X
RD	X	0	0	X	0	1
WR	X	1	1	X	1	0
RESET	X	0	1	X	0	0

Tabla 3.13. *Bits* de control.

Un tercer procedimiento es el encargado de la operación de las líneas $\overline{RD}$, $\overline{WR}$ y $\overline{RESET}$ del microcontrolador. Por su naturaleza, estas líneas nunca serán activadas simultáneamente, cuestión que es inherente a la operación del decodificador. La

operación de este procedimiento es totalmente análoga al procedimiento A0, con la sola excepción del número de parámetros que recibe. El procedimiento **RWRES** recibe como información la dirección (Tx ó Rx), y el estado de cada línea que puede controlar en el orden sugerido por el nombre del procedimiento.

Con estos tres procedimientos se tiene control total de las líneas que regulan las operaciones del PCF8584. Sin embargo, aún falta desarrollar procedimientos que se encarguen de la transferencia de información.

El procedimiento ***TXBYTE*** comienza por configurar la interfaz bidireccional para la transmisión llamando al procedimiento STROBE. Posteriormente activa la línea de escritura con el procedimiento RWRES, y tras dar un pequeño tiempo de espera envía a través de la dirección **$378** el *byte* de información. Dependiendo del instante en que se envíe, este *byte* puede ser parte de la información transmitida a un módulo o simplemente parámetros de programación para el PCF8584. Al terminar la operación se desactiva la línea de escritura.

El procedimiento ***RXBYTE*** es el encargado de recibir los datos provenientes del microcontrolador. Tras llamar al procedimiento STROBE se activa la línea de lectura, transfiriendo un *byte* de información hasta el puerto paralelo de la computadora. Debido a la construcción de la interfaz bidireccional, cinco de los *bits* se reciben en la dirección de memoria **$379** y los tres restantes en la $37A. Esta estructura se describió en la tabla 3.3, y el armado del *byte* original se logra haciendo operaciones lógicas con las partes. A través de estas operaciones se conservan los *bits* 7 a 3 de la dirección $379 y los *bits* 3 a 1 de la dirección $37A, conformando los 8 *bits* del *byte* deseado como se muestra en la tabla 3.14.

Bits recibidos	b_7 - b_3 @ \$379	b_3 - b_1 @ \$37A
Byte final	b_7 - b_3	b_2 - b_0

Tabla 3.14. Byte recibido.

Éstos son los procedimientos encargados de la transferencia de información a través del puerto paralelo a la interfaz I^2C. El siguiente paso es aprovechar estos procedimientos para la transmisión en el *bus*.

Las comunicaciones por el *bus* I^2C consta de dos etapas: la inicialización o programación del microcontrolador y la transferencia como tal, misma que se considera en dos casos independientes: recepción y transmisión. Debido a la operación con un solo maestro del sistema prototipo, se desarrolló el *software* necesario para la recepción (maestro) en la computadora. Philips proporciona lineamientos para el desarrollo de todos los modo de operación: recepción (como maestro o esclavo) y transmisión (maestro o esclavo). Dichas recomendaciones forman parte de las especificaciones del PCF8584 y se incluyen en el apéndice A.

La programación inicial del microcontrolador se realiza a través de un procedimiento de Turbo Pascal. Inicia activando la línea de *RESET*, para garantizar un estado inicial conocido del dispositivo. Una vez en este estado, el PCF8584 debe ser programado con una dirección esclava, aún cuando se emplee exclusivamente como maestro en las comunicaciones. Para este fin se activa la línea A_0, permitiendo el acceso al registro de control "S1". Se activa exclusivamente el *bit* 7 del registro, dirigiendo el apuntador al registro de dirección " S0' " y desactivando las interrupciones y demás funciones innecesarias en esta etapa. Posteriormente se libera A_0 y se envía la dirección esclava a través del *bus*, quedando almacenada en el registro adecuado gracias a la programación previa del registro S1. Para esta aplicación se utiliza la dirección \$AA, elegida arbitrariamente.

El siguiente paso en la inicialización del dispositivo consiste en determinar las frecuencias de operación. Puesto que se utiliza un oscilador de 12 MHz y se desea la máxima frecuencia en el *bus*, el *byte* que se almacena en el registro de reloj es $1C, que indica estas dos condiciones. Para la programación del registro se eleva A_0 y se modifican los *bits* ES1 y ES2 para apuntar hacia el registro S2. Hecho esto se libera A_0 y se transmite la palabra de programación ($1C).

Por último, es necesario habilitar las operaciones de formato I^2C en el microcontrolador. Dichas operaciones son la generación de *Acknowledge, Start* y *Stop*, así como encender el *bit* ESO, encargado de permitir la operación I^2C. La palabra de control requerida, $C1, se transmite con la línea A_0 en nivel alto, acción que deja al PCF8584 listo para recibir o transmitir en modo maestro o esclavo.

El algoritmo para la recepción en modo maestro se describirá a continuación. Su diagrama de flujo es parte de las recomendaciones de Philips incluidas en el apéndice A.

Una vez inicializado el microcontrolador se procede a la recepción en modo maestro. Ésta es la opción utilizada en un sistema por encuesta como es el conformado por los módulos y la computadora. Podría desarrollarse de manera muy similar un sistema coordinado por interrupciones en el que sea necesario programar todos los modos de transferencia de información entre el *bus* y la computadora.

Gracias a que el sistema propuesto es unimaestro, no es necesario revisar si el *bus* está libre, por lo que se procede directamente a la transmisión de la dirección esclava de interés, que en este caso es la del microcontrolador 80C552 como elemento central de algún módulo.

Esta transmisión se realiza al enviar, con A_0 en nivel bajo, la dirección de siete *bits* al PCF8584, donde será convertida a formato I^2C. Acto seguido se eleva la línea A_0 para acceder al registro de control, donde se encienden los *bits* PIN, ESO y STA. Dichos *bits* encienden la interfaz I^2C, habilitan la generación de las condiciones de *Start* y Acknowledge, y PIN indica que no ha recibido información. Puesto que se desea una operación de lectura, el último *bit* transmitido en este *byte* debe ser "1" (R).

El procedimiento anterior establece la comunicación maestro / esclavo entre la computadora y alguno de los módulos. La información se recibe a través de un proceso reiterativo que se describe a continuación.

La recepción puede subdividirse en dos procesos: uno de consulta en el registro de estado, y otro que constituye la recepción misma. Para la etapa de consulta la línea A_0 permanece en nivel alto mientras se consulta el *bit* PIN. Esto permite determinar si ha llegado un *byte* nuevo de información, procedimiento alternativo a la operación con interrupciones. En el momento en que PIN está en un estado bajo se procede a revisar la presencia de un nivel alto en LRB, indicador de una transmisión exitosa seguida de la condición de *Acknowledge*. Sabiendo que existe un *byte* nuevo para leer, la línea A_0 se coloca en nivel bajo para leer el *byte* del registro S0 utilizando el procedimiento ***RXBYTE***. Finalmente se almacena este dato en un archivo, a través del procedimiento **ALARCHIVO** que será descrito posteriormente.

El proceso de recepción se repetirá tantas veces como sea necesario,siguiendo los pasos del párrafo anterior cada vez. Es importante observar que el primer *byte* recibido en una transmisión corresponde a la dirección esclava del dispositivo, y puede ser indeseable.

Para suspender la comunicación después del último *byte* deseado se accede al registro de control y se coloca el *bit* ACK en cero, con lo cual se suspende la generación de *Acknowledge* y la transmisión se detiene. Para liberar el *bus* en el caso de un sistema multimaestro es necesario generar una condición de *Stop*, modificando el *bit* correspondiente en el registro de control.

La coordinación de las operaciones de módulo están a cargo del software específico, descrito previamente. Es por ello que el inicio de conversión, la extracción de los bytes hacia la memoria y su posterior conversión y transmisión en formato I^2C no dependen en absoluto del *software* aquí descrito. La interacción entre la computadora y los módulos se realiza siempre a nivel del protocolo I^2C.

El último procedimiento que conforma este programa es el encargado de almacenar la información en un archivo secuencial cuyos datos serán interpretados posteriormente. Este procedimiento llamado **ALARCHIVO** genera en el disco de la computadora un archivo en el que se registra el módulo y parámetro de medición, la fecha y hora en que llega la información a la computadora, y por último el valor de la lectura. Cada grupo se almacena en un renglón, haciendo de los archivos un registro comprensible aún sin la ayuda del programa de interfaz que se describirá posteriormente.

La fecha y hora se obtienen del sistema operativo de la computadora. El módulo y parámetro de medición están claramente definidos por la dirección I^2C que se utilizó para recabar la información, y la lectura llega a través del *bus*. Dicha lectura sufre una adecuación antes de almacenarse en el archivo debido a la escala de la conversión A / D.

El conjunto de procedimientos anteriormente descrito conforma el programa de adquisición de datos elaborado en *Turbo Pascal 6.0* para MS-DOS. Esto permite que la adquisición de datos sea realizada por cualquier computadora basada en el microprocesador 80X86 o hasta un 8088.

Para la constante adquisición de datos es necesario que el programa se ejecute continuamente, es decir, que permanezca residente en memoria. Con el fin de simplificar esta labor, y aprovechando las características de las computadoras empleadas, el programa se ejecutará desde el ambiente Windows en modo 386 extendido, lo que permite desempeño multitarea. De esta forma puede recibirse información mientras se utiliza el programa de monitoreo o cualquier otra aplicación.

INTERFAZ CON EL USUARIO

El programa para interpretación de los datos fue desarrollado en *Visual Basic 3.0 Professional*, constituyendo una solución amigable al procesamiento de la información. La elaboración del programa consiste en crear formas o ventanas donde se incluyen elementos de control y despliegue, como menús, botones, gráficas, tablas, etc. Una vez construidas las ventanas se establecen las relaciones entre los distintos controles y los demás objetos, cuyos parámetros se verán afectados por las acciones del usuario.

La primera ventana es la de propósito general. A través de la barra de menú de esta ventana, mostrada en la figura 3.17 , se tiene acceso al manejo de archivos, la elección del módulo y parámetro que se desea observar y hasta el tipo de interpretación de resultados que se desea. La opción de ayuda se incluye como muestra, aún cuando se consideró que el sistema es suficientemente intuitivo.

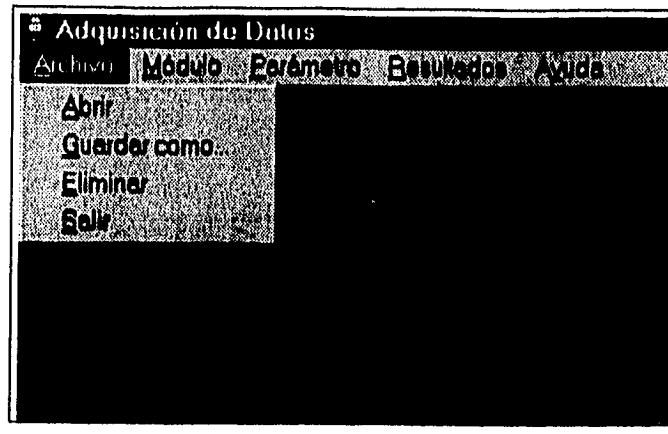


Figura 3.17. Ventana principal.

Cuando el usuario inicia la ejecución del programa, el archivo de trabajo es el último que fue elaborado a partir de la adquisición constante. Sin embargo, es posible que se desee analizar información de algún archivo previamente almacenado en el disco duro de la computadora. Ésta es la función del comando *Abrir*, el cual funciona a través del recuadro de diálogo mostrado en la figura 3.18.

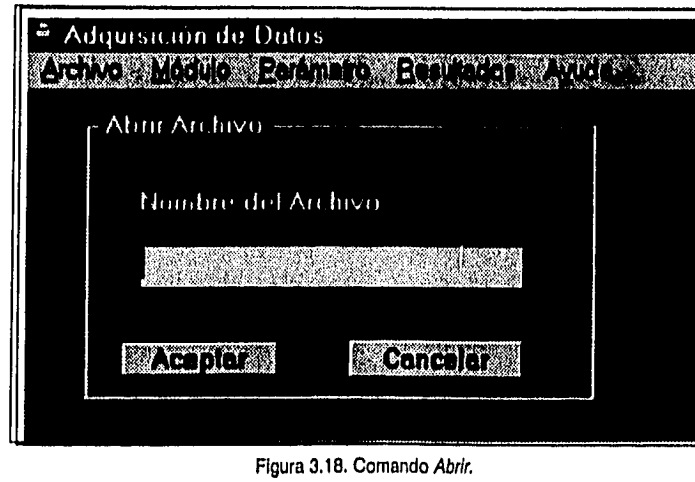


Figura 3.18. Comando *Abrir*.

De manera análoga, el comando *Guardar como* permite que la información siendo analizada se almacene en un archivo con nombre y ubicación arbitrariamente elegidos por el usuario. El recuadro de diálogo se muestra en la figura 3.19.

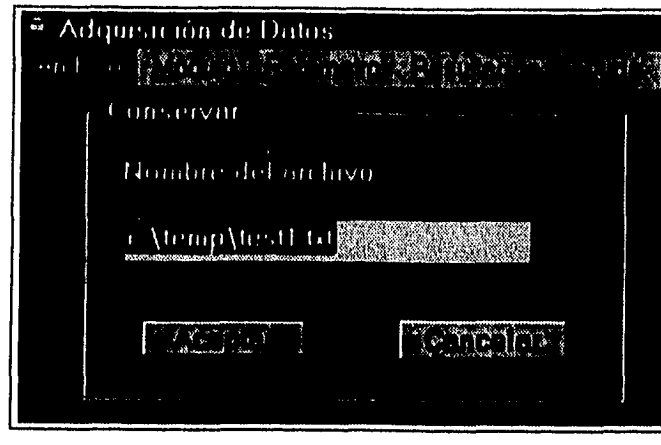


Figura 3.19. Comando *Guardar Como.*

Si por algún motivo el usuario deseara eliminar el archivo que está revisando, tanto de la memoria como del disco, deberá acceder a la ventana mostrada en la figura 3.20, correspondiente al comando *Eliminar.*

Como última opción dentro de la sección de *Archivo* se tiene el comando *Salir,* cuyo significado es evidente

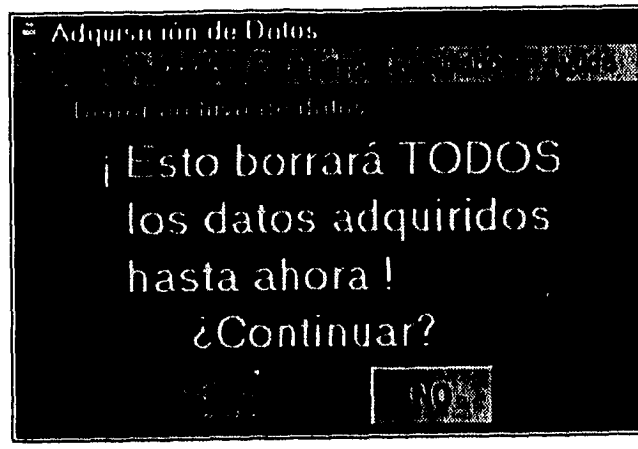


Figura 3.20. Comando *Eliminar.*

Las secciones de *Módulo* y *Parámetro* determinan a través de pequeñas marcas (✓) la información que se desea analizar, como se muestra en las figuras 3.21 y 3.22.

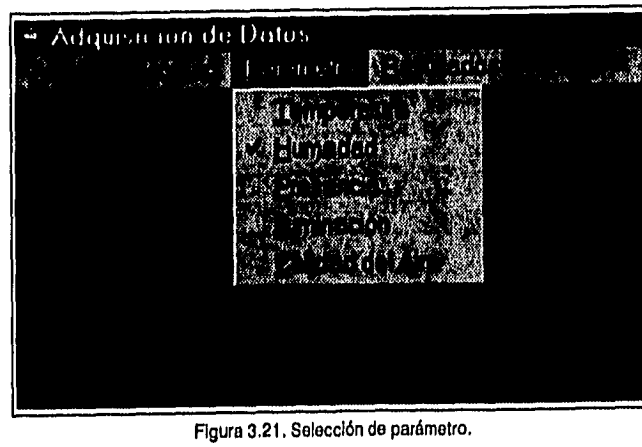


Figura 3.21. Selección de parámetro.

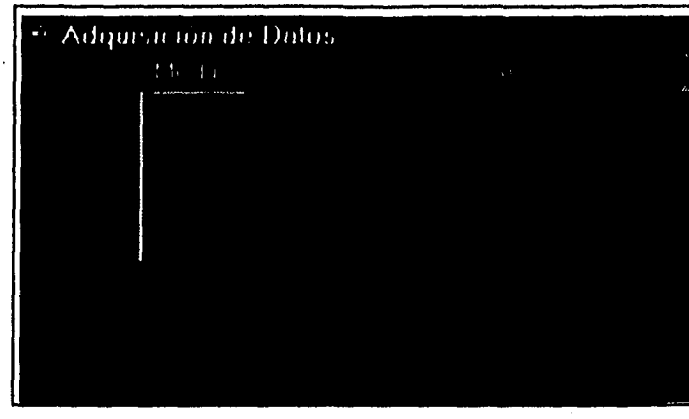


Figura 3.22. Selección de módulo.

Una vez seleccionados el archivo a utilizar y la información específica a analizar, puede accederse a la sección de *Resultados*, cuya pantalla se muestra en la figura 3.23. En esta sección del menú puede elegirse utilizar un reporte o una gráfica como medio de interpretación de los datos referentes a los parámetros seleccionados anteriormente. A través de este menú también se tiene acceso al módulo de monitoreo continuo, cuya pantalla se muestra en la figura 3.24.

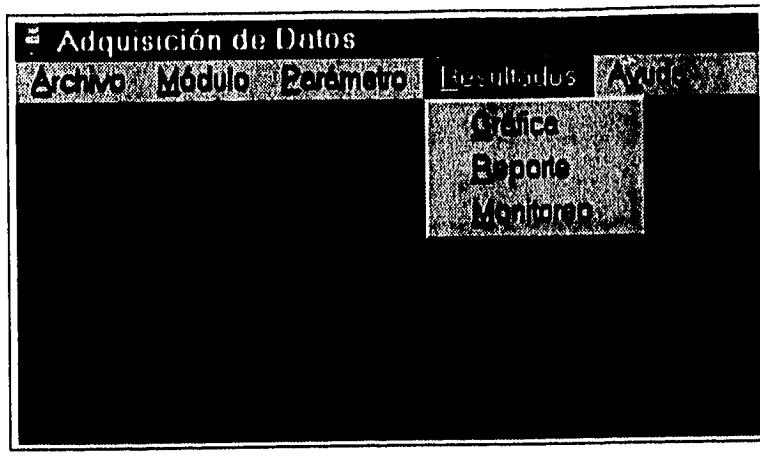


Figura 3.23. Pantalla de selección de resultados.

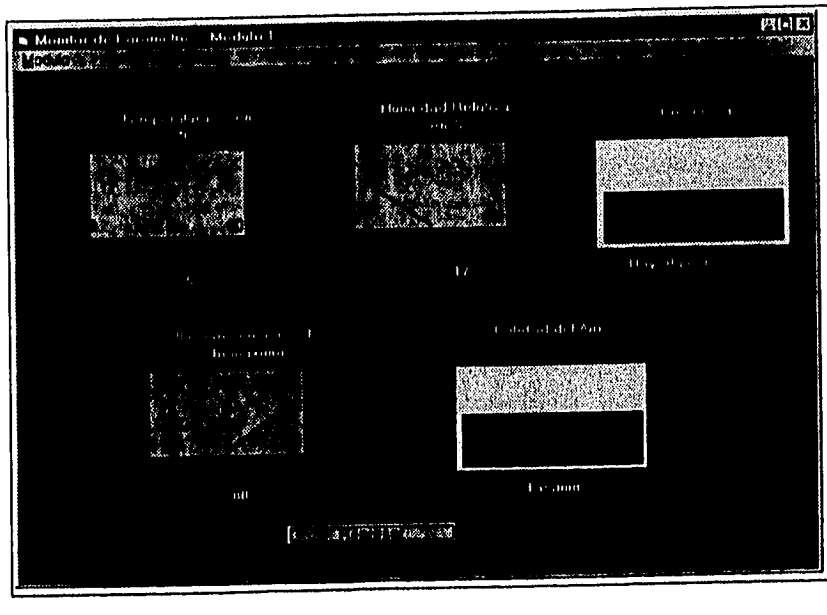


Figura 3.24. Pantalla de monitoreo.

Al interpretar los datos, ya sea con una gráfica o a través de un reporte, se muestra una pantalla con el número de módulo en cuestión, el parámetro bajo análisis y los valores relevantes. Debido a que el sistema continuará adquiriendo datos, es necesaria la opción de actualizar la información, lograda a través del botón adecuado. Adicionalmente puede imprimirse el resultado obtenido al oprimir el icono de impresión. Las pantallas de resultados se muestran en las figuras 3.25. y 3.26.

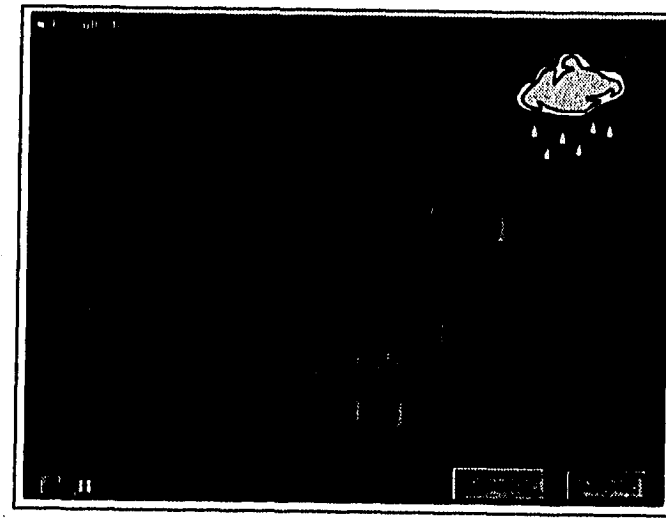


Figura 3.25. Resultados gráficos.

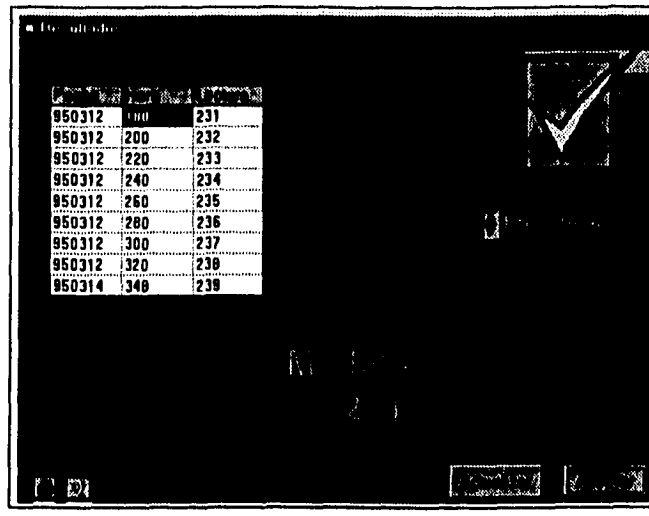


Figura 3.26. Reporte de resultados.

Éstas son todas las ventanas que componen el programa de interpretación de datos. La mayoría de las funciones que realizan son parte constituyente del lenguaje de programación, y únicamente es necesario relacionarlas con los controles deseados.

Las ventanas de resultados hacen una búsqueda en el archivo actual y, aprovechando la estructura de los archivos, discriminan la información relevante al número de módulo y parámetro establecidos por el usuario.

La ventana de monitoreo lee continuamente una copia del archivo generado para este fin por el procedimiento ALARCHIVO discutido en la sección anterior.

El resto de las acciones que realiza el programa son inherentes a *Visual Basic* o bien directamente a Windows, como es el control de impresión. De cualquier forma, este programa logra una integración ideal para los propósitos de análisis e interpretación de datos adquiridos a través del *bus* I^2C.

Capítulo 4

RESULTADOS

Este capítulo trata de la construcción y pruebas de operación de un prototipo. Con este equipo podrá demostrarse la operación de la computadora como elemento concentrador de información, así como sus capacidades de monitoreo continuo.

Para efectos de pruebas, sólamente se utilizará un módulo. Es importante recordar que los principios de funcionamiento son completamente generales y aplicables a un número indefinido de dispositivos conectados al *bus* I^2C.

4.1. Construcción

Debido a la naturaleza eventual de este prototipo, su construcción se desarrolló sobre tarjetas perforadas con pequeñas huellas de cobre al reverso, como la mostrada en la figura 4.1.

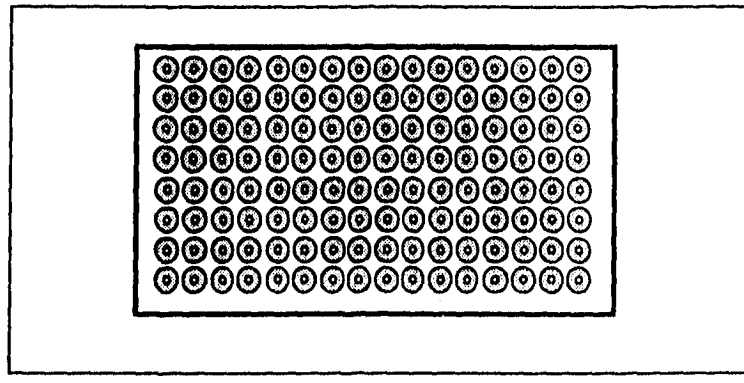


Figura 4.1. Tableta perforada.

El primer subsistema en construirse fue la interfaz bidireccional de 8 *bits*. En esta tarjeta se colocaron los cuatro *latches* octales, el inversor y el decodificador, así como dos hileras de postes que permiten la conexión al resto del sistema. Por un extremo, los postes llegan a través de un cable plano de 20 hilos a un conector DB-25 que se conecta al puerto paralelo de la computadora personal. La otra hilera de conectores conduce los datos y señales de control a la interfaz I^2C a través de cable plano. Por este último cable viaja también la polarización, proveniente de la tableta I^2C. Todos los circuitos integrados se montaron sobre bases para permitir su fácil extracción en caso de algún problema grave. Ninguno de ellos es sensible a la estática, por lo que no fue necesario tomar precauciones especiales. Esta primera tarjeta se muestra en la figura 4.2.

Figura 4.2. Interfaz bidireccional.

La siguiente sección en construirse fue la interfaz I²C: el PCF8584 y accesorios, como se muestra en la figura 4.3. Se instalaron el regulador de voltaje, el oscilador integrado y el microcontrolador en otra tableta de características y tamaño muy similares a la anterior. La entrada del eliminador de baterías se hace a través de un conector común, y tanto el controlador del *bus* como el oscilador se instalaron sobre bases. Además, su extracción permite hacer pruebas de la interfaz bidireccional y las líneas de control.

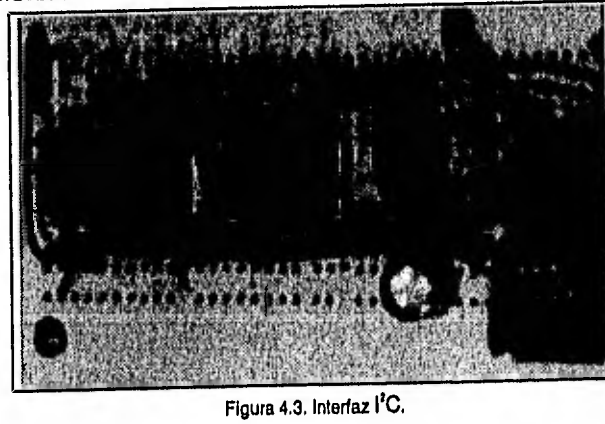


Figura 4.3. Interfaz I²C.

Una vez más debido a que éste es un prototipo, no se tomaron en cuenta las trayectorias de las conexiones ni la colocación de los dispositivos sobre la tableta. El diseño de circuitos impresos debe contemplar estos parámetros con el fin de reducir los efectos indeseables de capacitancia e inductancia. Para resolver este dilema, se utilizó el programa OrCad PCB II para la generación de los circuitos impresos que se proponen en el apéndice D. Las impresiones están al doble del tamaño real para reducir errores al elaborar los negativos.

Estos dos subsistemas, que permiten la conexión de la computadora personal al *bus* I²C , se encapsulan en un solo gabinete como se muestra en la figura 4.4.

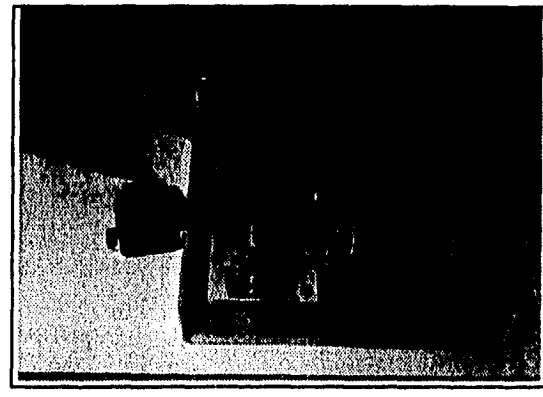


Figura 4.4. Interfaz completa.

La construcción del módulo de medición y simulación siguió un procedimiento totalmente análogo. Como puede observarse en la figura 4.5, el módulo autónomo consiste en un microprocesador y su respectiva memoria (RAM y ROM). Todos los circuitos integrados se colocaron cuidadosamente en bases, pues esto permite la extracción de elementos selectos para pruebas independientes.

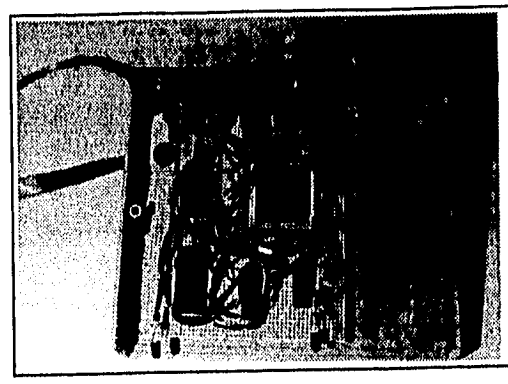


Fig. 4.5. Módulo de medición.

El módulo de simulación contiene algunos sensores y otros dispositivos que representan los resultados obtenidos por sensores reales. Los sensores miden temperatura e iluminación. Para simular las lecturas de calidad de aire y de humedad se utilizan potenciómetros, y para simular el parámetro de presencia se utiliza un interruptor. El módulo se ilustra en la figura 4.6.

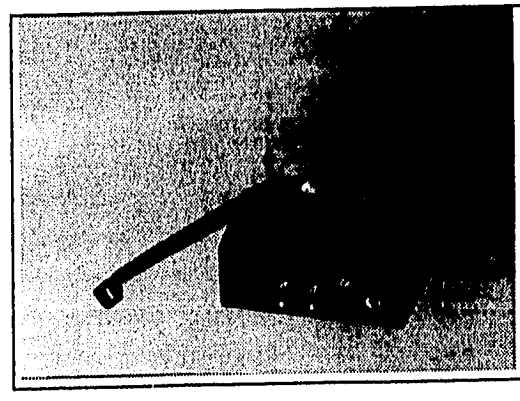


Figura 4.6. Módulo de simulación.

Como se muestra en la figura 4.7, el módulo de simulación es independiente del módulo autónomo, por lo que este último puede alimentarse directamente con transductores reales. La información viaja entre los módulos a través de un cable plano de 10 hilos (aunque en esta aplicación únicamente se utilizan siete de ellos) con conectores para postes. Esto permite utilizar los ocho canales del convertidor analógico digital del microprocesador, si bien este prototipo únicamente emplea cinco de ellos.

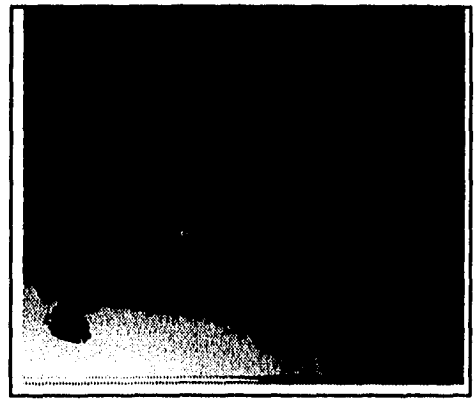


Figura 4.7. Módulos de simulación y medición.

4.2 Pruebas de Operación y Resultados

Una vez terminada la implementación del sistema, se realizaron una serie de pruebas para verificar su correcto desempeño. El primer paso fue probar individualmente todas las componentes del sistema. Fue aquí donde se descubrieron problemas como soldaduras en frío, conectores mal configurados y

falsos contactos. Una vez asegurada la correcta interconexión de estos elementos, se procedió a probar los módulos individualmente.

El primer módulo en probarse fue el sistema de medición y simulación. Esto fue debido a su enorme simplicidad, tanto de construcción como de operación. Los voltajes analógicos que entrega están acotados entre 0 y 5 volts, y se comportaron exactamente como esperado. La única discrepancia con el diseño radica en la no linealidad del fotorresistor, misma que se traduce en lecturas ligeramente no lineales. Posiblemente convendría utilizar potenciómetros de precisión para una simulación mejorada, mas los resultados no justifican el incremento en costo. Finalmente, se decidió amplificar la señal proveniente del LM35 con el fin de aprovechar la resolución del convertidor A/D.

Las pruebas de este módulo consistieron en polarizar el equipo y medir los voltajes entregados a través del cable y los conectores. La limpieza de los potenciómetros eliminó el comportamiento ocasionalmente errático de las simulaciones, y la adición de un capacitor al interruptor suprimió los picos de voltaje.

Una vez que la etapa de transducción funcionó adecuadamente, se procedió a verificar la operación del módulo autónomo. Con este fin se elaboraron diversos programas para el microcontrolador 80C552 que utilizaran varias de las funciones requeridas para la operación final.

Las pruebas se iniciaron con un programa que exclusivamente direccionara una localidad en ROM. Esto permitió verificar que el direccionamiento fuera correcto, pues las lecturas en el *bus* de direcciones se mantuvieron estables. Acto seguido se hizo una prueba de escritura a RAM para verificar que ésta también estuviera funcionando adecuadamente. El programa escribía una cuenta progresiva de 8 *bits* en RAM, lo recuperaba al acumulador

y posteriormente colocaba el dato leído en el puerto 4. De esta manera se pudo corroborar la operación del microcontrolador.

Una vez probada la operación básica del módulo, se prosiguió a las pruebas de los canales de conversión A/D. Para esto se hizo una rutina que tomara las lecturas del módulo de simulación y colocara el resultado de la conversión directamente en el puerto 4. El siguiente paso, una vez logrado esto, fue guardar los datos en memoria y verificar su permanencia.

Finalmente, fue necesario probar la operación del *bus* I²C. Para comprobar su operación como transmisor-esclavo, se configuró la interfaz interna y se revisó el reconocimiento de la dirección propia mediante el registro de estado (S1STA). El primer paso fue conseguir que el módulo contestara por medio de un ACK al llamado de su direción esclava. Además se verificó que el módulo contestara a su dirección y no a cualquier llamada que se hiciera en el *bus*. Acto seguido se transmitió información arbitraria (únicamente ceros, únicamente $FF y posteriormente una cuenta progresiva) para observar determinísticamente los resultados de la transferencia.

Paralelamente, se comprobó la correcta interacción del módulo I²C con la interfaz bidireccional. Fue necesario hacer varios ajustes al *software* de modo que se evitaran conflictos al cambiar la dirección de memoria encargada de controlar el PCF8584. Finalmente se conectaron las líneas del *bus* I²C al osciloscopio, donde se pudo verificar la generación del reloj en la línea SCL, así como el ACK requerido para una recepción de maestro.

Como pruebas de transmisión, se efectuaron adquisiciones con longitudes del *bus* de hasta 10 metros. A pesar de que las lecturas presentaron errores ocasionales, se alcanzó la velocidad aproximada de 90 kbps. Sin

embargo, se presentó un problema enorme: el desfasamiento de los datos con su identificador de parámetro.

Debido a que sólamente se utilizaron cinco canales, se elaboró una rutina que transmite un *bit* de sincronía como sexto dato en cada ciclo. De esta forma los datos y sus identificadores quedan perfecta y constantemente correlacionados y, en caso de error, el sistema se vuelve a sincronizar automáticamente.

La interpretación de datos con el *software* desarrollado para Windows se probó generando archivos con datos aleatorios que representaran las lecturas de parámetros. Tanto la modalidad de interpretación como la de monitor se comportaron a la perfección, con lo que concluyeron las pruebas al equipo.

CONCLUSIONES

La experimentación con este sistema podría continuarse utilizando diversos métodos de transmisión, combinando las bondades de formatos como RS-485 con el *bus* I^2C, o bien utilizando los extensores de línea propuestos por Philips, P82B715P.

Si se considera el proyecto como su estructura generalizada, es decir como una red de propósito general, es importante experimentar con sistemas multimaestro para determinar la eficiencia de un sistema con colisiones, un número mayor de dispositivos, una mayor variedad de direcciones y software notablemente más elaborado.

Si bien el sistema funciona adecuadamente para el monitoreo de parámetros a cortas distancias, la presencia de errores impide su aplicación inmediata a situaciones

que requieran gran precisión. El protocolo es capaz de detectar errores en el bit de Acknowledge, pero esto es insuficiente como esquema de detección y corrección. Para la aplicación a gran escala de esta red es indispensable utilizar métodos como CRC o Hamming que garanticen una transmisión confiable a distancia. Esto es particularmente crítico si se desea aplicar la red al control o monitoreo de motores, cuya interferencia electromagnética es particularmente problemática.

En el desarrollo del prototipo fue necesaria la inclusión de un bit para sincronía, permitiendo que las mediciones siempre correspondan al parámetro adecuado y el ruido no provocase defasamientos.

Uno de los méritos más grandes del diseño del sistema es la interfaz bidireccional de 8 bits. Utilizando una computadora suficientemente rápida, así como elementos discretos de familias más veloces, pueden lograrse conexiones de muy alta velocidad (150 kbytes por segundo) que permitan enlaces directos entre computadoras, dispositivos de respaldo, etc. Para esta aplicación, esta velocidad es claramente excesiva.

Una de las enormes limitantes del software que interpreta la información es su incapacidad de exportar directamente a paquetería comercial de bases de datos u hojas de cálculo. Esto puede hacerse a través de las opciones de OLE (Object Linking and Embedding) y DDE (Dynamic Data Exchange) incluidas en Visual Basic como herramientas de desarrollo. Sin embargo, sólo sería lógico hacerlo si la aplicación lo justifica y el tratamiento de los datos requiere más que la simple interpretación brindada por el programa elaborado.

En conclusión, el protocolo I^2C, sus interfaces y el software elaborados en este trabajo presentan una solución de bajo costo para enlaces de comunicaciones a corta distancia y velocidades medias. Cada vez más fabricantes ofrecen interfaces I^2C como parte de sus microcontroladores y otros dispositivos, por lo que la versatilidad de diseño es muy grande. Sin embargo, su falta de compatibilidad con otros protocolos, así como la ausencia de un esquema de corrección de errores, hacen de su utilización algo poco realista más allá de aplicaciones a microescala.

BIBLIOGRAFÍA

1. The Waite Group's Tricks of the MS-DOS Masters,Second Edition
   Angermeyer, J. & Fahringer, R. & Jaeger, K.
   SAMS
   E.U.A., 1991

2. Transducer Handbook
   Boyle, H.B.
   Butterworth-Heinemann
   Inglaterra, 1992

3. "Busman's guide to I^2C"
   Button, M.
   Electronics World + Wireless World
   Enero, 1994

4. "I^2C via the PC"
   Davies, J.
   Electronics World + Wireless World
   Diciembre, 1994

5. PC Magazine Guide to Connectivity
   Derfler, F.J.
   ZD Press
   E.U.A., 1991

6. La Biblia del Turbo Pascal
   Duntemann, J.
   ANAYA MULTIMEDIA
   Madrid, 1991.

7. Interfacing With the IBM Personal Computer,Second Edition
   Eggebrecht, L.C.
   SAMS
   E.U.A., 1994

8. Experimental Methods for Engineers
   Hollman, J.P.
   McGraw-Hill
   E.U.A., 1989

9. Electronic Projects to Control your Home Environment
Horn, D.
McGraw-Hill
E.U.A., 1994

10. Upgrading and Repairing Pcs.Second edition
Mueler, S.
Q.U.E.
E.U.A., 1992

11. The alarm, sensor and security cookbook.1st Edition
Petruzzellis, T.
TAB Books
E.U.A.

12. Application Notes and Development Tools for 80C51 Microcontrollers
Data Handbook
Philips Semiconductors
E.U.A., 1995

13. 80C51-Based 8-Bit Microcontrollers
Data handbook IC20
Philips Semiconductors
E.U.A., 1995

14. "I^2C Interface for Pcs"
Ruffel, J.
Elektor Electronics
Febrero, 1992

15. Electronic Circuits: Discrete and Integrated.Third Edition
Schilling, D. & Belove, C.
McGraw-Hill
E.U.A., 1989

16. Handbook of QuickBASIC
Schnider, D.
BRADY
E.U.A., 1991

17. "LPT Frequently Asked Questions"
Stewart, Z
zstewart@nyx.cs.du.edu
Julio, 1994

Apéndice A: Hojas de especificaciones

DATA SHEET

PCF8584
I²C-bus controller

Product specification
Supersedes data of May 1994
File under Integrated Circuits, IC12

1995 Aug 29

Philips Semiconductors

PHILIPS

CONTENTS

1 FEATURES

- Parallel-bus to I²C-bus protocol converter and interface
- Compatible with most parallel-bus microcontrollers/microprocessors including 8049, 8051, 6800, 68000, and Z80
- Both master and slave functions
- Automatic detection and adaption to bus interface type
- Programmable interrupt vector
- Multi-master capability
- I²C-bus monitor mode
- Long-distance mode (4-wire)
- Operating supply voltage 4.5 to 5.5 V
- Operating temperature range: −40 to +85 °C.

2 GENERAL DESCRIPTION

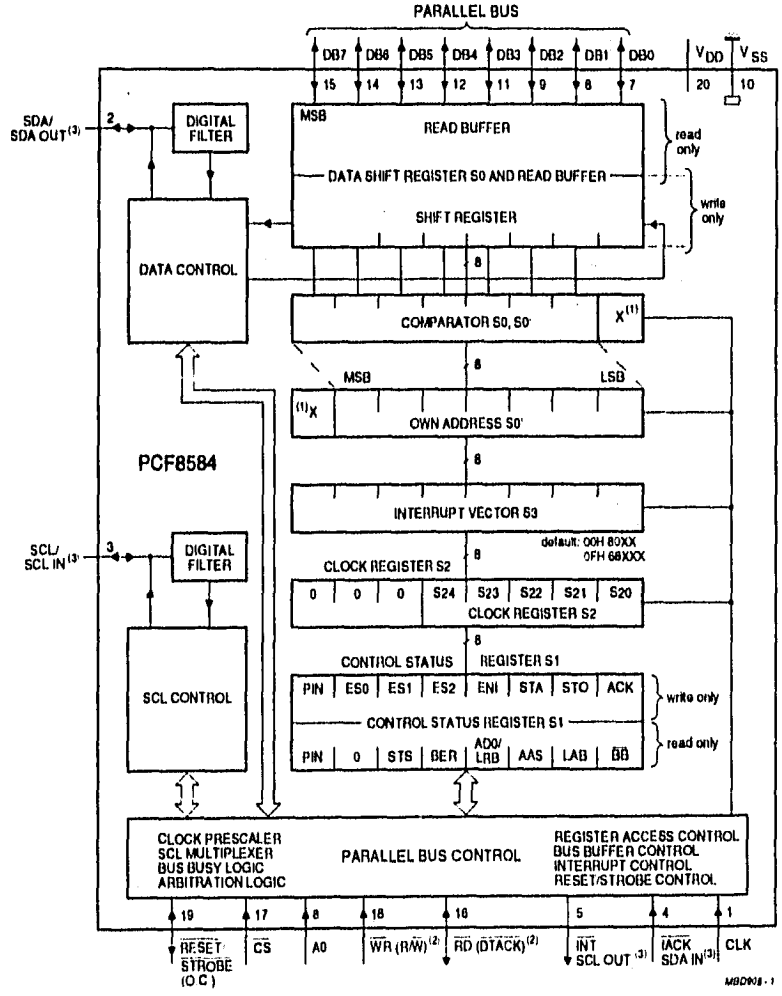
The PCF8584 is an integrated circuit designed in CMOS technology which serves as an interface between most standard parallel-bus microcontrollers/microprocessors and the serial I²C-bus. The PCF8584 provides both master and slave functions.

Communication with the I²C-bus is carried out on a byte-wise basis using interrupt or polled handshake. It controls all the I²C-bus specific sequences, protocol, arbitration and timing. The PCF8584 allows parallel-bus systems to communicate bidirectionally with the I²C-bus.

3 ORDERING INFORMATION

TYPE NUMBER	PACKAGE		
	NAME	DESCRIPTION	VERSION
PCF8584P	DIP20	plastic dual in-line package; 20 leads (300 mil)	SOT146-1
PCF8584T	SO20	plastic small outline package; 20 leads; body width 7.5 mm	SOT163-1

4 BLOCK DIAGRAM

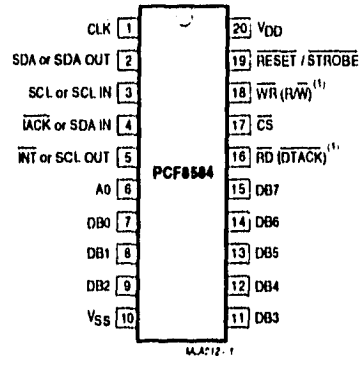


(1) X = don't care.
(2) Pin mnemonics between brackets indicate the 68000 mode pin designations.
(3) These pin mnemonics represent the long-distance mode pin designations.

Fig.1 Block diagram.

5 PINNING

SYMBOL	PIN	I/O	DESCRIPTION
CLK	1	I	clock input from microcontroller clock generator (internal pull-up)
SDA or SDA OUT	2	I/O	I²C-bus serial data input/output (open-drain). Serial data output in long-distance mode.
SCL or SCL IN	3	I/O	I²C-sarial clock input/output (open-drain). Serial clock input in long-distance mode.
$\overline{IACK}$ or SDA IN	4	I	Interrupt acknowledge input (internal pull-up); when this signal is asserted the interrupt vector in register S3 will be available at the bus Port if the ENI flag is set. Serial data input in long-distance mode.
$\overline{INT}$ or SCL OUT	5	O	Interrupt output (open-drain); this signal is enabled by the ENI flag in register S1. It is asserted when the PIN flag is reset. (PIN is reset after 1 byte is transmitted or received over the I²C-bus). Serial clock output in long-distance mode.
A0	6	I	Register select input (internal pull-up); this input selects between the control/status register and the other registers. Logic 1 selects register S1, logic 0 selects one of the other registers depending on bits loaded in ES0, ES1, and ES2 of register S1.
DB0	7	I/O	bidirectional 8-bit bus Port 0
DB1	8	I/O	bidirectional 8-bit bus Port 1
DB2	9	I/O	bidirectional 8-bit bus Port 2
V_{SS}	10	–	ground
DB3	11	I/O	bidirectional 8-bit bus Port 3
DB4	12	I/O	bidirectional 8-bit bus Port 4
DB5	13	I/O	bidirectional 8-bit bus Port 5
DB6	14	I/O	bidirectional 8-bit bus Port 6
DB7	15	I/O	bidirectional 8-bit bus Port 7
$\overline{RD}$ ($\overline{DTACK}$)	16	I/(O)	$\overline{RD}$ is the read control input for MAB8049, MAB8051 or Z80-types. $\overline{DTACK}$ is the data transfer control output for 68000-types (open-drain).
$\overline{CS}$	17	I	chip select input (internal pull-up)
$\overline{WR}$ (R/$\overline{W}$)	18	I	$\overline{WR}$ is the write control input for MAB8048, MAB8051, or Z80-types (internal pull-up). R/$\overline{W}$ control input for 68000-types.
$\overline{RESET}$/ $\overline{STROBE}$	19	I/O	Reset input (open-drain); this input forces the I²C-bus controller into a predefined state; all flags are reset, except PIN, which is set. Also functions as strobe output.
V_{DD}	20	–	supply voltage



(1) Pin mnemonics between brackets indicate the 68000 mode pin designations.

Fig.2 Pin configuration.

6 FUNCTIONAL DESCRIPTION

6.1 General

The PCF8584 acts as an interface device between standard high-speed parallel buses and the serial I²C-bus. On the I²C-bus, it can act either as master or slave. Bidirectional data transfer between the I²C-bus and the parallel-bus microcontroller is carried out on a byte-wise basis, using either an interrupt or polled handshake. Interface to either 80XX-type (e.g. MAB8048, MAB8051, Z80) or 68000-type buses is possible. Selection of bus type is automatically performed (see Section 6.2).

Table 1 Control signals utilized by the PCF8584 for microcontroller/microprocessor interfacing

TYPE	R/$\overline{W}$	$\overline{WR}$	$\overline{R}$	$\overline{DTACK}$	$\overline{IACK}$
8048/ 8051	no	yes	yes	no	no
68000	yes	no	no	yes	yes
Z80	no	yes	yes	no	yes

The structure of the PCF8584 is similar to that of the I²C-bus interface section of the MABXXXX/PCF84(C)XX-series of microcontrollers, but with a modified control structure. The PCF8584 has five internal register locations. Three of these (own address register S0', clock register S2 and interrupt vector S3) are used for initialization of the PCF8584. Normally they are only written once directly after resetting of the PCF8584.

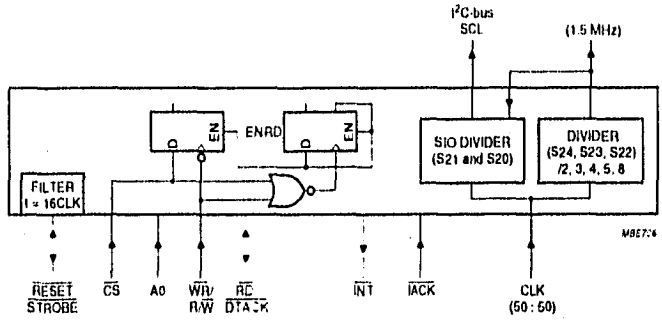
The remaining two registers function as double registers (data buffer/shift register S0, and control/status register S1) which are used during actual data transmission/reception. By using these double registers, which are separately write and read accessible, overhead for register access is reduced. Register S0 is a combination of a shift register and data buffer.

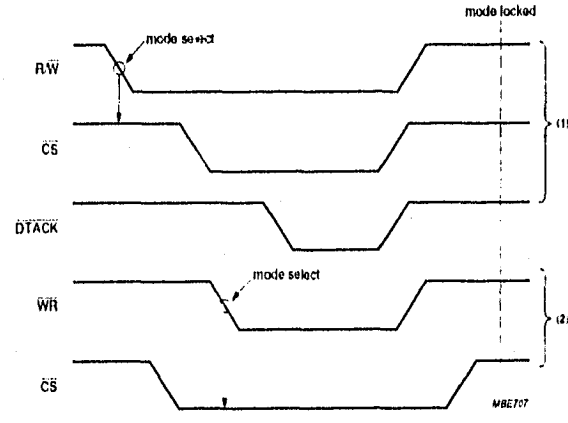
Register S0 performs all serial-to-parallel interfacing with the I²C-bus.

Register S1 contains I²C-bus status information required for bus access and/or monitoring.

6.2 Interface Mode Control (IMC)

Selection of either an 80XX mode or 68000 mode interface is achieved by detection of the first $\overline{WR}$-$\overline{CS}$ signal sequence. The concept takes advantage of the fact that the write control input is common for both types of interfaces. An 80XX-type interface is default. If a HIGH-to-LOW transition of $\overline{WR}$ (R/$\overline{W}$) is detected while $\overline{CS}$ is HIGH, the 68000-type interface mode is selected and the $\overline{DTACK}$ output is enabled. Care must be taken that $\overline{WR}$ and $\overline{CS}$ are stable after reset.





(1) Bus timing: 68000 mode write cycle.
(2) Bus timing: 80XX mode.

Fig.3 68000/80XX timing sequence utilized by the Interface Mode Control (IMC).

6.3 Set-up registers S0', S2 and S3

Registers S0', S2 and S3 are used for initialization of the PCF8584 (see Fig.5 'Initialization sequence' flowchart).

6.4 Own address register S0'

When the PCF8584 is addressed as slave, this register must be loaded with the 7-bit I²C-bus address to which the PCF8584 is to respond. During initialization, the own address register S0' must be written to, regardless whether it is later used. The Addressed As Slave (AAS) bit in status register S1 is set when this address is received (the value in S0 is compared with the value in S0'). Note that the S0 and S0' registers are offset by one bit; hence, programming the own address register S0' with a value of 55H will result in the value AAH being recognized as the PCF8584's slave address (see Fig.1).

Programming of S0' is accomplished via the parallel-bus when A0 is LOW, with the appropriate bit combinations set in control status register S1 (S1 is written when pin A0 = HIGH). Bit combinations for accessing all registers are given in Table 5. After reset, S0' has default address 00H (PCF8584 is thus initially in monitor mode, see Section 6.12).

6.5 Clock register S2

Register S2 provides control over chip clock frequency and SCL clock frequency. S20 and S21 provide a selection of 4 different I²C-bus SCL frequencies which are shown in Table 2. Note that these SCL frequencies are only obtained when bits S24, S23 and S22 are programmed to the correct input clock frequency (f_{clk}).

Table 2 Register S2 selection of SCL frequency

BIT		APPROXIMATE SCL FREQUENCY f_{SCL} (kHz)
S21	S20	
0	0	90
0	1	45
1	0	11
1	1	1.5

S22, S23 and S24 are used for control of the internal clock prescaler. Due to the possibility of varying microcontroller clock signals, the prescaler can be programmed to adapt to 5 different clock rates, thus providing a constant internal clock. This is required to provide a stable time base for the SCL generator and the digital filters associated with the I²C-bus signals SCL and SDA. Selection for adaption to external clock rates is shown in Table 3.

Programming of S2 is accomplished via the parallel-bus when A0 = LOW, with the appropriate bit combinations set in control status register S1 (S1 is written when A0 = HIGH). Bit combinations for accessing all registers are given in Table 5.

Table 3 Register S2 selection of clock frequency

INTERNAL CLOCK FREQUENCY			
S24	S23	S22	f_{clk} (MHz)
0	X[1]	X[1]	3
1	0	0	4.43
1	0	1	6
1	1	0	8
1	1	1	12

Note

1. X = don't care.

6.6 Interrupt vector S3

The interrupt vector register provides an 8-bit user-programmable vector for vectored-interrupt microcontrollers. The vector is sent to the bus port (DB7 to DB0) when an interrupt acknowledge signal is asserted and the ENI (enable interrupt) flag is set. Default vector values are:

- Vector is '00H' in 80XX mode
- Vector is '0FH' in 68000 mode.

On reset the PCF8584 is in the 80XX mode, thus the default interrupt vector is '00H'.

6.7 Data shift register/read buffer S0

Register S0 acts as serial shift register and read buffer interfacing to the I²C-bus. All read and write operations to/from the I²C-bus are done via this register. S0 is a combination of a shift register and a data buffer; parallel data is always written to the shift register, and read from the data buffer. I²C-bus data is always shifted in or out of shift register S0.

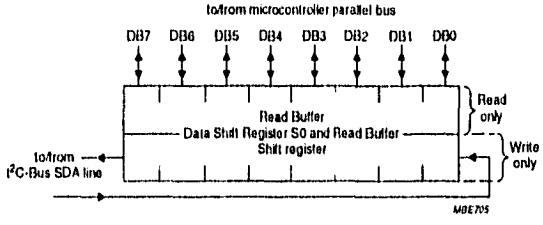


Fig.4 Data shift register/bus buffer S0.

In receiver mode the data from the shift register is copied to the read buffer during the acknowledge phase. Further reception of data is inhibited (SCL held LOW) until the S0 read buffer is read (see Section 6.8.1.1).

In the transmitter mode data is transmitted to the I²C-bus as soon as it is written to the S0 shift register if the serial I O is enabled (ESO = 1).

Remarks:

1. A minimum of 6 clock cycles must elepse between consecutive parallel-bus accesses to the PCF8584 when the I²C-bus controller operates at 8 or 12 MHz. This may be reduced to 3 clock cycles for lower operating frequencies.
2. To start a read operation immediataly after a write, it is necessary to read the S0 read buffer in order to invoke reception of the first byte ('dummy read' of the address). Immediately after the acknowledgement, this first byte will be transferred from the shift register to the read buffer. The **next** raed will then transfer the correct value of the first byte to the microcontroller bus (see Fig.7).

6.8 Control/status register S1

Register S1 controls I²C-bus operation and provides I²C-bus status information. Register S1 is accessed by a HIGH signal on register select input A0. For more efficient communication between microcontroller/processor and the I²C-bus, register S1 has separate read and write functions for all bit positions (see Fig.3). The write-only section provides register access control and control over I²C-bus signals, while the read-only section provides I²C-bus status information.

Table 4 Control/status register S1

CONTROL/STATUS	BITS								MODE
Control[1]	PIN	ESO	ES1	ES2	ENI	STA	STO	ACK	write only
Status[2]	PIN	0[3]	STS	BER	AD0/ LRB	AAS	LAB	$\overline{BB}$	read only

Notes

1. For further information see Section 6.8.1.
2. For further information see Section 6.8.2.
3. Logic 1 if not-initialized.

6.8.1 REGISTER S1 CONTROL SECTION

The write-only section of S1 enables access to registers S0, S0', S1, S2 and S3, and controls I²C-bus operation; see Table 4.

6.8.1.1 PIN (Pending Interrupt Not)

When the PIN bit is written with a logic 1, all status bits are reset to logic 0. This may serve as a software reset function (see Figs 5 to 8). PIN is the only bit in S1 which may be both read and written to. PIN is mostly used as a status bit for synchronizing serial communication, see Section 6.8.2.

6.8.1.2 ESO (Enable Serial Output)

ESO enables or disables the serial I²C-bus I/O. When ESO is LOW, register access for initialization is possible. When ESO is HIGH, I²C-bus communication is enabled; communication with serial shift register S0 is enabled and the S1 bus status bits are made available for reading.

Table 5 Register access control; ESO = 0 (serial interface off) and ESO = 1 (serial interface on)

INTERNAL REGISTER ADDRESSING 2-WIRE MODE				
A0	**ES1**	**ES2**	**$\overline{IACK}$**	**FUNCTION**
ESO = 0; serial interface off[1]				
1	0	X	1[2]	R/W S1: control
0	0	0	1[2]	R/W S0': (own address)
0	0	1	1[2]	R/W S3: (interrupt vector)
0	1	0	1[2]	R/W S2: (clock register)
ESO = 0; serial interface on				
1	0	X	1	W S1: control
1	0	X	1	R S1; status
0	0	0	1	R/W S0: (data)
0	0	1	1	R/W S3: (interrupt vector)
X	0	X	0	R S3: (interrupt vector ACK cycle))

Notes

1. With ESO = 0, bits ENI, STA, STO and ACK of S1 can be read for test purposes.
2. 'X' if ENI = 0.

6.8.1.3 ES1 and ES2

ES1 and ES2 control selection of other registers for initialization and control of normal operation. After these bits are programmed for access to the desired register (shown in Table 5), the register is selected by a logic LOW level on register select pin A0.

6.8.1.4 ENI

This bit enables the external interrupt output $\overline{INT}$, which is generated when the PIN bit is active (logic 0).

This bit must be set to logic 0 before entering the long-distance mode, and remain at logic 0 during operation in long-distance mode.

6.8.1.5 STA and STO

These bits control the generation of the I²C-bus START condition and transmission of slave address and R/$\overline{W}$ bit, generation of repeated START condition, and generation of the STOP condition (see Table 7).

Table 6 Register access control; ESO = 1 (serial interface on) and ES1 = 1; long-distance (4-wire) mode; note 1

INTERNAL REGISTER ADDRESSING; LONG-DISTANCE (4-WIRE) MODE				
A0	**ES1**	**ES2**	**$\overline{IACK}$**	**FUNCTION**
1	1	X	1	W S1; control
1	1	X	X	R S1; status
0	1	X	X	R/W S0; (data)

Note

1. Trying to read from or write to registers other than S0 and S1 (setting ESO = 0) brings the PCF8584 out of the long-distance mode.

Table 7 Instruction table for serial bus control

STA	STO	PRESENT MODE	FUNCTION	OPERATION
1	0	SLV/REC	START	transmit START + address, remain MST/TRM if R/$\overline{W}$ = 0; go to MST/REC if R/$\overline{W}$ = 1
1	0	MST/TRM	REPEAT START	same as for SLV/REC
0	1	MST/REC; MST/TRM	STOP READ; STOP WRITE	transmit STOP go to SLV/REC mode; note 1
1	1	MST	DATA CHAINING	send STOP, START and address after last master frame without STOP sent; note 2
0	0	ANY	NOP	no operation; note 3

Notes

1. In master receiver mode, the last byte must be terminated with ACK bit HIGH ('negative acknowledge').
2. If both STA and STO are set HIGH simultaneously in master mode, a STOP condition followed by a START condition + address will be generated. This allows 'chaining' of transmissions without relinquishing bus control.
3. All other STA and STO mode combinations not mentioned in Table 7 are NOPs.

6.8.1.6 ACK

This bit must be set normally to a logic 1. This causes the I²C-bus controller to send an acknowledge automatically after each byte (this occurs during the 9th clock pulse). The bit must be reset (to logic 0) when the I²C-bus controller is operating in master/receiver mode and requires no further data to be sent from the slave transmitter. This causes a negative acknowledge on the I²C-bus, which halts further transmission from the slave device.

6.8.2 REGISTER S1 STATUS SECTION

The read-only section of S1 enables access to I²C-bus status information; see Table 4.

6.8.2.1 PIN bit

'Pending Interrupt Not' (MSB of register S1) is a status flag which is used to synchronize serial communication and is set to logic 0 whenever the PCF8584 requires servicing. The PIN bit is normally read in polled applications to determine when an I²C-bus byte transmission/reception is completed. The PIN bit may also be written, see Section 6.8.1.

Each time a serial data transmission is initiated (by setting the STA bit in the same register), the PIN bit will be set to logic 1 automatically (inactive). When acting as transmitter, PIN is also set to logic 1 (inactive) each time S0 is written. In receiver mode, the PIN bit is automatically set to logic 1 (inactive) each time the data register S0 is read.

After transmission or reception of one byte on the I²C-bus (9 clock pulses, including acknowledge), the PIN bit will be automatically reset to logic 0 (active) indicating a complete byte transmission/reception. When the PIN bit is subsequently set to logic 1 (inactive), all status bits will be reset to logic 0. PIN is also set to zero on a BER (bus error) condition.

In polled applications, the PIN bit is tested to determine when a serial transmission/reception has been completed. When the ENI bit (bit 4 of write-only section of register S1) is also set to logic 1 the hardware interrupt is enabled. In this case, the PIN flag also triggers an external interrupt (active LOW) via the $\overline{INT}$ output each time PIN is reset to logic 0 (active).

When acting as slave transmitter or slave receiver, while PIN = 0, the PCF8584 will suspend I²C-bus transmission by holding the SCL line LOW until the PIN bit is set to logic 1 (inactive). This prevents further data from being transmitted or received until the current data byte in S0 has been read (when acting as slave receiver) or the next data byte is written to S0 (when acting as slave transmitter).

PIN bit summary:

- The PIN bit can be used in polled applications to test when a serial transmission has been completed. When the ENI bit is also set, the PIN flag sets the external interrupt via the $\overline{INT}$ output.
- Setting the STA bit (start bit) will set PIN = 1 (inactive).
- In transmitter mode, after successful transmission of one byte on the I²C-bus the PIN bit will be automatically reset to logic 0 (active) indicating a complete byte transmission.
- In transmitter mode, PIN is set to logic 1 (inactive) each time register S0 is written.
- In receiver mode, PIN is set to logic 0 (active) on completion of each received byte. Subsequently, the SCL line will be held LOW until PIN is set to logic 1.
- In receiver mode, when register S0 is read, PIN is set to logic 1 (inactive).
- In slave receiver mode, an I²C-bus STOP condition will set PIN = 0 (active).
- PIN = 0 if a bus error (BER) occurs.

6.8.2.2 STS

When in slave receiver mode, this flag is asserted when an externally generated STOP condition is detected (used only in slave receiver mode).

6.8.2.3 BER

Bus error; a misplaced START or STOP condition has been detected. Resets $\overline{BB}$ (to logic 1; inactive), sets PIN = 0 (active).

6.8.2.4 LRB/AD0

'Last Received Bit' or 'Address 0 (General Call) bit'. This status bit serves a dual function, and is valid only while PIN = 0:

1. LRB holds the value of the last received bit over the I²C-bus while AAS = 0 (not addressed as slave). Normally this will be the value of the slave acknowledgement; thus checking for slave acknowledgement is done via testing of the LRB.
2. AD0; when AAS = 1 ('Addressed As Slave' condition), the I²C-bus controller has been addressed as a slave. Under this condition, this bit becomes the 'AD0' bit and will be set to logic 1 if the slave address received was the 'general call' (00H) address, or logic 0 if it was the I²C-bus controller's own slave address.

6.8.2.5 AAS

'Addressed As Slave' bit. Valid only when PIN = 0. When acting as slave receiver, this flag is set when an incoming address over the I²C-bus matches the value in own address register S0' (shifted by one bit, see Section 6.4), or if the I²C-bus 'General Call' address (00H) has been received ('General Call' is indicated when AD0 status bit is also set to logic 1, see Section 6.8.2.4).

6.8.2.6 LAB

'Lost Arbitration' Bit. This bit is set when, in multi-master operation, arbitration is lost to another master on the I²C-bus.

6.8.2.7 $\overline{BB}$

'Bus Busy' bit. This is a read-only flag indicating when the I²C-bus is in use. A zero indicates that the bus is busy, and access is not possible. This bit is set/reset (logic 1/logic 0) by STOP/START conditions.

6.9 Multi-master operation

To avoid conflict between data and repeated START and STOP operations, multi-master systems have some limitations:

- When powering up multiple PCF8584s in multi-master systems, the possibility exists that one node may power up slightly after another node has already begun an I²C-bus transmission; the Bus Busy condition will thus not have been detected. To avoid this condition, a delay should be introduced in the initialization sequence of each PCF8584 equal to the longest I²C-bus transmission, see flowchart 'PCF8584 initialization' (Fig.5).

6.10 Reset

A LOW level pulse on the $\overline{RESET}$ input forces the I²C-bus controller into a well-defined state. All flags in S1 are reset to logic 0, except the PIN flag, which is set to logic 1. S0' and S3 are set to 00H.

The $\overline{RESET}$ pin is also used for the $\overline{STROBE}$ output signal. Both functions are separated on-chip by a digital filter. The reset input signal has to be sufficiently long (minimum 30 clock cycles) to pass through the filter. The $\overline{STROBE}$ output signal is sufficiently short (8 clock cycles) to be blocked by the filter. For more detailed information on the strobe function see Section 6.12.

6.11 Comparison to the MAB8400 I²C-bus interface

The structure of the PCF8584 is similar to that of the MAB8400 series of microcontrollers, but with a modified control structure. Access to all I²C-bus control and status registers is done via the parallel-bus port in conjunction with register select input A0, and control bits ES0, ES1 and ES2.

6.11.1 DELETED FUNCTIONS

The following functions are not available in the PCF8584:

- Always selected (ALS flag)
- Access to the bit counter (BC0 to BC2)
- Full SCL frequency selection (2 bits instead of 5 bits)
- The non-acknowledge mode (ACK flag)
- Asymmetrical clock (ASC flag).

6.11.2 ADDED FUNCTIONS

The following functions either replace the deleted functions or are completely new:

- Chip clock prescaler
- Assert acknowledge bit (ACK flag)
- Register selection bits (ES1 and ES2 flags)
- Additional status flags (BER, 'bus error')
- Automatic interface control between 80XX and 68000-type microcontrollers
- Programmable interrupt vector
- Strobe generator
- Bus monitor function
- Long-distance mode [non-I²C-bus mode (4-wire): only for communication between parallel-bus processors using the PCF8584 at each interface point].

6.12 Special function modes

6.12.1 STROBE

When the I²C-bus controller receives its own address (or the '00H' general call address) followed immediately by a STOP condition (i.e. no further data transmitted after the address), a strobe output signal is generated at the $\overline{RESET}/\overline{STROBE}$ pin (pin 19). The $\overline{STROBE}$ signal consists of a monostable output pulse (active LOW), 8 clock cycles long (see Fig.9). It is generated after the STOP condition is received, preceded by the correct slave address. This output can be used as a bus access controller for multi-master parallel-bus systems.

6.12.2 LONG-DISTANCE MODE

The long-distance mode provides the possibility of longer-distence serial communication between parallel processors via two I²C-bus controllers. This mode is selected by setting ES1 to logic 1 while the serial interface is enabled (ESO = 1).

In this mode the I²C-bus protocol is transmitted over 4 unidirectional lines, SDA OUT, SCL IN, SDA IN and SCL IN (pins 2, 3, 4 and 5). These communication lines should be connected to line drivers/receivers (example: RS422) for long-distance applications. Hardware characteristics for long-distance transmission are then given by the chosen standard. Control of data transmission is the same as in normal I²C-bus mode. After reading or writing data to shift register S0, long-distance mode must be initialized by setting $\overline{ESO}$ and ES1 to logic 1. Because the interrupt output $\overline{INT}$ is not available in this operating mode, synchronization of data transmission/reception must be polled via the PIN bit.

Remarks:

Before entering the long-distance mode, ENI must be set to logic 0.

When powering up an PCF8584-node in long-distance mode, the PCF8584 must be isolated from the 4-wire bus via 3-state line drivers/receivers until the PCF8584 is properly initialized for long-distance mode. Failure to implement this precaution will result in system malfunction.

6.12.3 MONITOR MODE

When the 7-bit own address register S0' is loaded with all zeros, the I²C-bus controller acts as a passive I²C monitor. The main features of the monitor mode are:

- The controller is always selected.
- The controller is always in the slave receiver mode.
- The controller never generates an acknowledge.
- The controller never generates an interrupt request.
- A pending interrupt condition does not force SCL LOW.
- $\overline{BB}$ is set to logic 0 after detection of a START condition, and reset to logic 1 after a STOP condition.
- Received data is automatically transferred to the read buffer.
- Bus traffic is monitored by the PIN bit, which is reset to logic 0 after the acknowledge bit of an incoming byte has been received, and is set to logic 1 as soon as the first bit of the next incoming byte is detected. Reading the data buffer S0 sets the PIN bit to logic 1. Data in the read buffer is valid from PIN = 0 and during the next 8 clock pulses (until next acknowledge).
- AAS is set to logic 1 at every START condition, and reset at every 9th clock pulse.

7 SOFTWARE FLOWCHART EXAMPLES

7.1 Initialization

The flowchart of Fig.5 gives an example of a proper initialization sequence of the PCF8584. When implemented in 808X-type bus systems with peripherals sharing a common $\overline{WR}$ signal, this procedure should be executed first before communicating with other bus peripherals to avoid false programming of the PCF8584 (see Section 6.2)

7.2 Implementation

The flowcharts (Figs 6 to 9) illustrate proper programming sequences for implementing master transmitter, master receive, and master transmitter, repeated start and master receiver modes in polled applications.

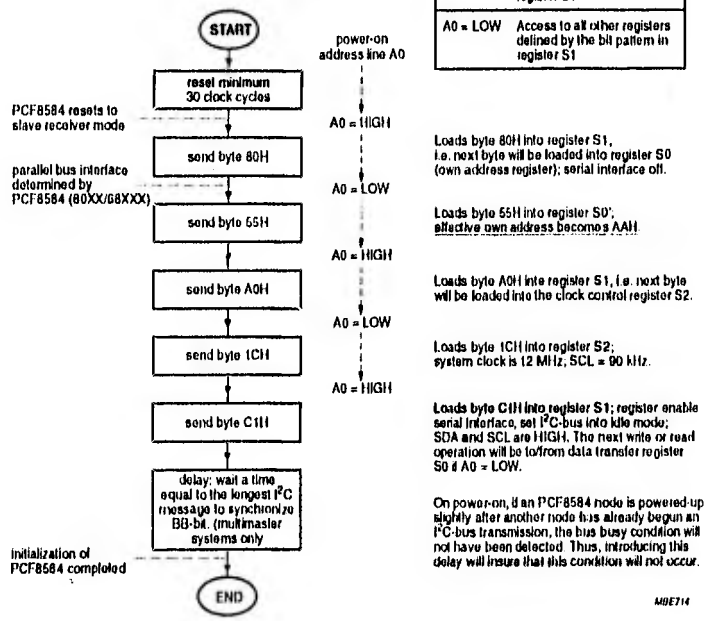


Fig.5 PCF8584 initialization sequence.

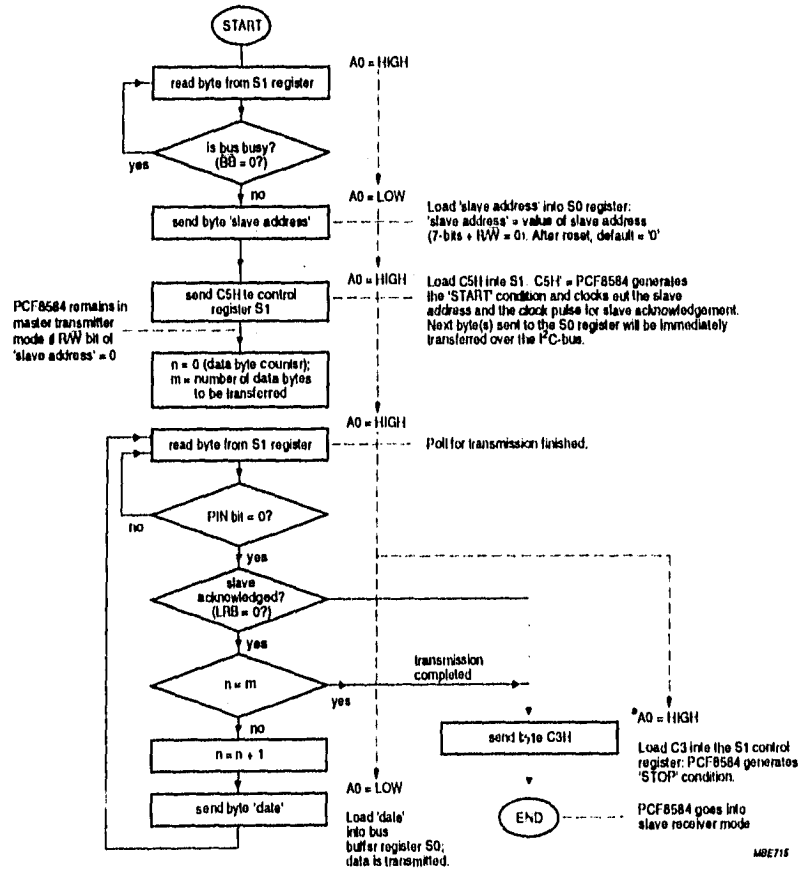


Fig.6 PCF8584 master transmitter mode.

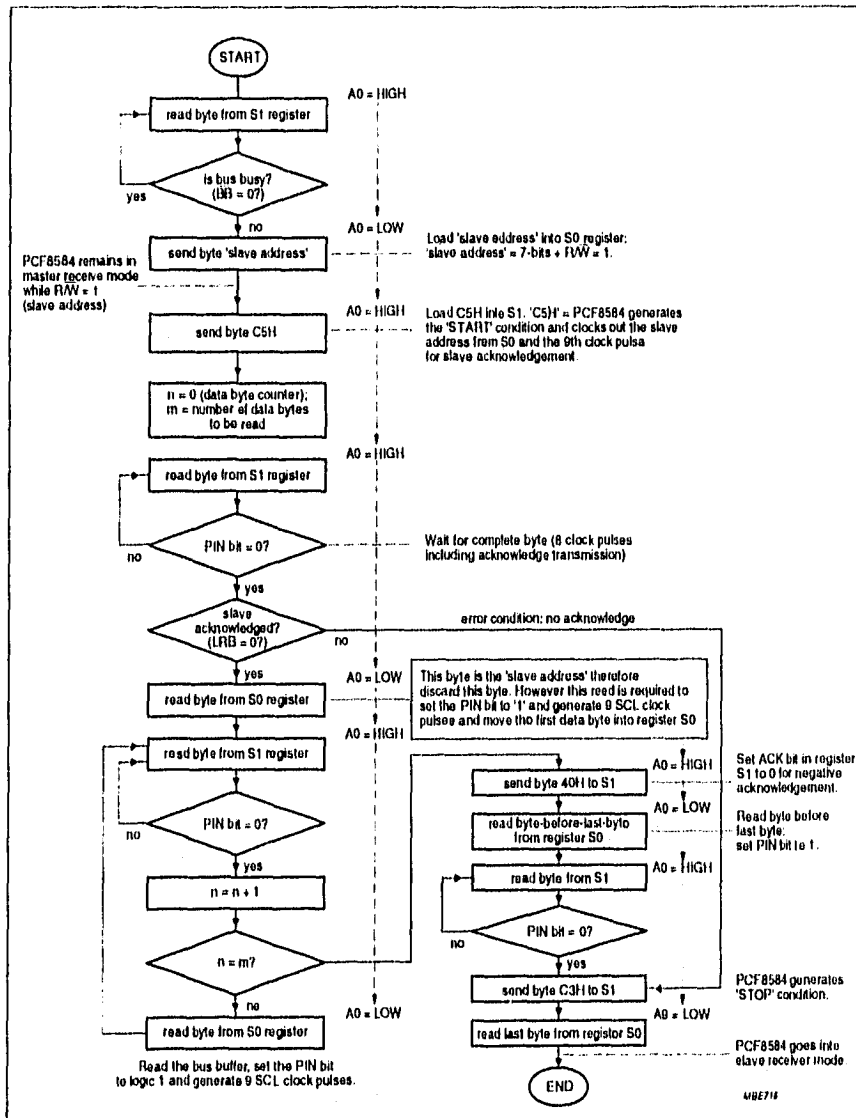


Fig.7 PCF8584 master receiver mode.

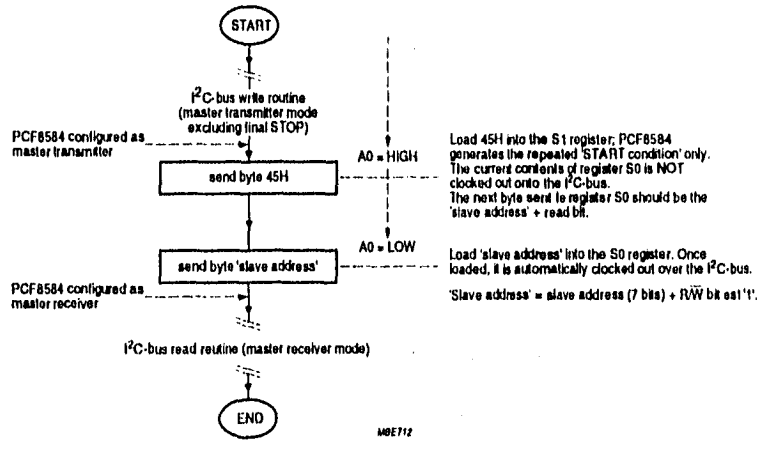


Fig.8 Master transmitter followed by repeated START and becoming master receiver.

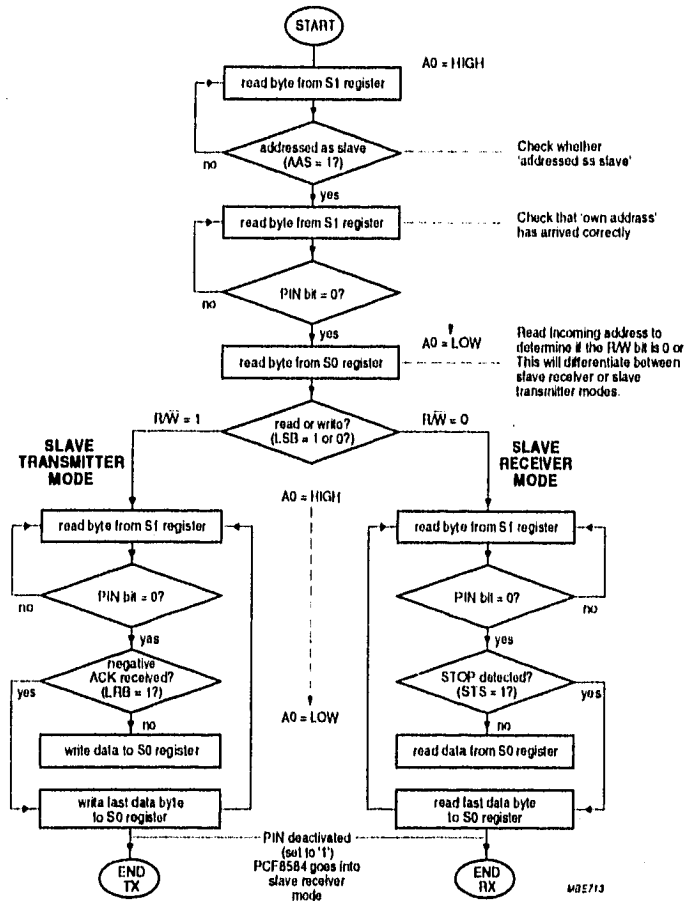
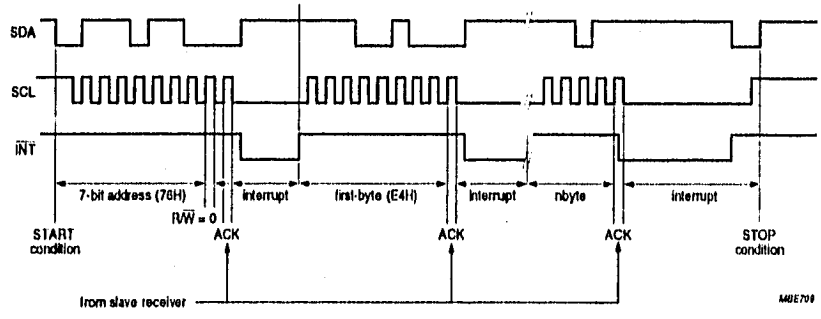


Fig.9 Slave receiver/slave transmitter modes.

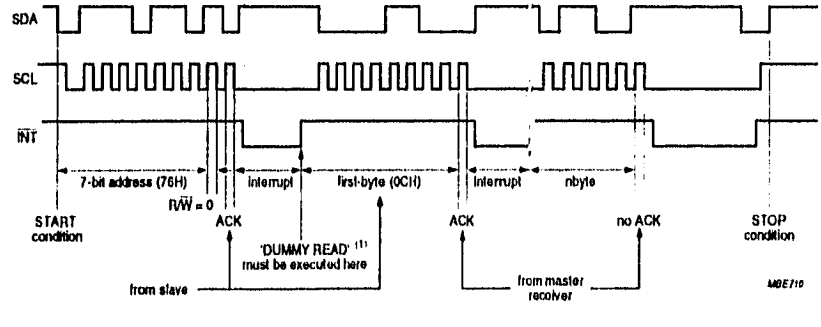
8 I²C-BUS TIMING DIAGRAMS

The diagrams (Figs 10 to 13) illustrate typical timing diagrams for the PCF8584 in master/slave functions. For detailed description of the I²C-bus protocol, please refer to *"The I²C-bus and how to use it"*; Philips document ordering number 9398 393 40011.



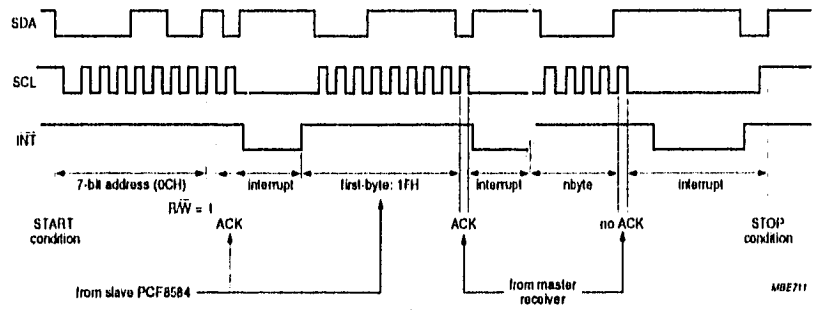
Master PCF8584 writes data to slave receiver.

Fig.10 Bus timing diagram; master transmitter mode.



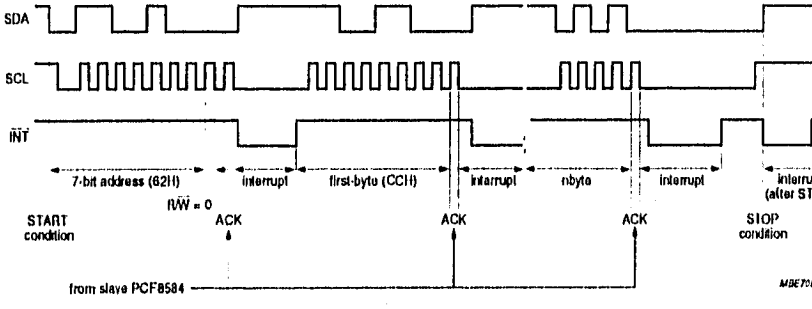
Master PCF8584 reads data from slave receiver.

Fig.11 Bus timing diagram; master receiver mode.



External master receiver reads data from PCF8584.

Fig.12 Bus timing diagram; slave transmitter mode.



Slave PCF8584 is written to by external master transmitter.

Fig.13 Bus timing diagram; slave receiver mode.

9 LIMITING VALUES

In accordance with the Absolute Maximum Rating System (IEC 134).

SYMBOL	PARAMETER	MIN.	MAX.	UNIT
V_{DD}	supply voltage	−0.3	+7.0	V
V_I	voltage range (any input)	−0.8	V_{DD} + 0.5	V
I_I	DC input current (any input)	−10	+10	mA
I_O	DC output current (any output)	−10	+10	mA
P_{tot}	total power dissipation	−	300	mW
P_O	power dissipation per output	−	50	mW
T_{amb}	operating ambient temperature	−40	+85	°C
T_{stg}	storage temperature	−65	+150	°C

10 HANDLING

Inputs and outputs are protected against electrostatic discharge in normal handling. However, to be totally safe, it is good practice to take normal precautions appropriate to handling MOS devices (see *"Handling MOS Devices"*).

11 DC CHARACTERISTICS

V_{DD} = 5 V ±10%; T_{amb} = -40 to +85 °C; unless otherwise specified.

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
Supply						
V_{DD}	supply voltage		4.5	5.0	5.5	V
I_{DD}	supply current	standby; note 1	–	–	2.5	µA
		operating; notes 1 and 2	–	–	1.5	mA
Inputs						
CLK, $\overline{IACK}$, A0, $\overline{CS}$, $\overline{WR}$, $\overline{RD}$, $\overline{RESET}$ AND D0 to D7						
V_{IL}	LOW level input voltage	note 3	0	–	0.8	V
V_{IH}	HIGH level input voltage	note 3	2.0	–	V_{DD}	V
SDA AND SCL						
V_{IL}	LOW level input voltage	note 4	0	–	$0.3V_{DD}$	V
V_{IH}	HIGH level input voltage	note 4	$0.7V_{DD}$	–	V_{DD}	V
R_i	resistance to V_{DD}	T_{amb} = 25 °C; note 5	25	–	100	kΩ
Outputs						
I_{OL}	LOW level output current	V_{OH} = 2.4 V; note 6	-2.4	–	–	mA
I_{OH}	HIGH level output current	V_{OL} = 0.4 V; note 6	3.0	–	–	mA
I_{OL}	leakage current	note 7	-1	–	+1	µA

Notes

1. Test conditions: 22 kΩ pull-up resistors on D0 to D7; 10 kΩ pull-up resistors on SDA, SCL, $\overline{RD}$; $\overline{RESET}$ connected to V_{SS}; remaining pins open-circuit.
2. CLK waveform of 12 MHz with 50% duty factor.
3. CLK, $\overline{IACK}$, A0, $\overline{CS}$, $\overline{WR}$, $\overline{RD}$, $\overline{RESET}$ and D0 to D7 are TTL level inputs.
4. SDA and SCL are CMOS level inputs.
5. CLK, $\overline{IACK}$, A0, $\overline{CS}$ and $\overline{WR}$.
6. D0 to D7.
7. D0 to D7 3-state, SDA, SCL, $\overline{INT}$, $\overline{RD}$, $\overline{RESET}$.

12 I²C-BUS TIMING SPECIFICATIONS

All the timing limits are valid within the operating supply voltage and ambient temperature range; V_{DD} = 5 V ±10%; T_{amb} = −40 to +85 °C; and refer to V_{IL} and V_{IH} with an input voltage of V_{SS} to V_{DD}.

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNIT
f_{SCL}	SCL clock frequency	–	–	100	kHz
t_{SW}	tolerable spike width on bus	–	–	100	ns
t_{BUF}	bus free time	4.7	–	–	µs
$t_{SU;STA}$	START condition set-up time	4.7	–	–	µs
$t_{HD;STA}$	START condition hold time	4.0	–	–	µs
t_{LOW}	SCL LOW time	4.7	–	–	µs
t_{HIGH}	SCL HIGH time	4.0	–	–	µs
t_r	SCL and SDA rise time	–	–	1.0	µs
t_f	SCL and SDA fall time	–	–	0.3	µs
$t_{SU;DAT}$	data set-up time	250	–	–	ns
$t_{HD;DAT}$	data hold time	0	–	–	ns
$t_{VD;DAT}$	SCL LOW to data out valid	–	–	3.4	µs
$t_{SU;STO}$	STOP condition set-up time	4.0	–	–	µs

13 PARALLEL INTERFACE TIMING

All the timing limits are valid within the operating supply voltage and ambient temperature range: V_{DD} = 5 V ±10%; T_{amb} = −40 to +85 °C; and refer to V_{IL} and V_{IH} with an input voltage of V_{SS} to V_{DD}. C_L = 100 pF; R_L = 1.5 kΩ (connected to V_{DD}) for open-drain and high-impedance outputs, where applicable (for measurement purposes only).

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
t_r	clock rise time	see Fig.14	–	–	6	ns
t_f	clock fall time	see Fig.14	–	–	6	ns
t_{CLK}	input clock period (50% ±5% duty factor)	see Fig.14	83	–	333	ns
t_{CLRL}	$\overline{CS}$ set-up to $\overline{RD}$ LOW	see Fig.16 and note 1	20	–	–	ns
t_{CLWL}	$\overline{CS}$ set-up to $\overline{WR}$ LOW	see Fig.15 and note 1	20	–	–	ns
t_{RHCH}	$\overline{CS}$ hold from $\overline{RD}$ HIGH	see Fig.16	0	–	–	ns
t_{WHCH}	$\overline{CS}$ hold from $\overline{WR}$ HIGH	see Fig.15	0	–	–	ns
t_{AVWL}	A0 set-up to $\overline{WR}$ LOW	see Fig.15	10	–	–	ns
t_{AVRL}	A0 set-up to $\overline{RD}$ LOW	see Fig.16	10	–	–	ns
t_{WHAI}	A0 hold from $\overline{WR}$ HIGH	see Fig.15	20	–	–	ns
t_{RHAI}	A0 hold from $\overline{RD}$ HIGH	see Fig.16	10	–	–	ns
t_{WLWH}	$\overline{WR}$ pulse width	see Fig.15	250	–	1000	ns
t_{RLRH}	$\overline{RD}$ pulse width	see Fig.16	250	–	1000	ns
t_{DVWH}	data set-up before $\overline{WR}$ HIGH	see Fig.15	150	–	–	ns
t_{RLDV}	data valid after $\overline{RD}$ LOW	see Fig.16	–	160	230	ns
t_{WHDI}	data hold after $\overline{WR}$ HIGH	see Fig.15	20	–	–	ns
t_{RHDF}	data bus floating after $\overline{RD}$ HIGH	see Fig.16 and note 4	–	160	180	ns
t_{AVCL}	A0 set-up to $\overline{CS}$ LOW	see Figs 17 and 18	10	–	–	ns

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
t_{WLCL}	R/$\overline{W}$R set-up to $\overline{CS}$ LOW	see Fig.17	10	–	–	ns
t_{RHCL}	R/$\overline{W}$R set-up to $\overline{CS}$ LOW	see Fig.18	10	–	–	ns
t_{CLDV}	data valid after $\overline{CS}$ LOW	see Fig.18 and note 3	–	160	230	ns
t_{CLDL}	$\overline{DTACK}$ LOW after $\overline{CS}$ LOW	see Figs 17 and 18	–	$2t_{CLK}$ + 75	$3t_{CLK}$ + 150	ns
t_{CHAI}	A0 hold from $\overline{CS}$ HIGH	see Fig.18	0	–	–	ns
t_{CHRL}	R/$\overline{W}$R hold from $\overline{CS}$ HIGH	see Fig.18	0	–	–	ns
t_{CHWH}	R/$\overline{W}$R hold from $\overline{CS}$ HIGH	see Fig.17	0	–	–	ns
t_{CHDF}	data bus float after $\overline{CS}$ HIGH	see Fig.18 and note 4	–	160	180	ns
t_{CHDE}	$\overline{DTACK}$ HIGH from $\overline{CS}$ HIGH	see Figs 17 and 18	–	100	120	ns
t_{CHDI}	data hold after $\overline{CS}$ HIGH	see Fig.17 and note 4	0	–	–	ns
t_{DVCL}	data set-up to $\overline{CS}$ LOW	see Fig.17	0	–	–	ns
t_{ALIE}	$\overline{INT}$ HIGH from $\overline{IACK}$ LOW	see Figs 19 and 20	–	130	180	ns
t_{ALDV}	data valid after $\overline{IACK}$ LOW	see Figs 19 and 20	–	200	250	ns
t_{ALAE}	$\overline{IACK}$ pulse width	see Fig.20	280	–	–	ns
t_{AHDI}	data hold after $\overline{IACK}$ HIGH	see Fig.20	–	–	150	ns
t_{ALDL}	$\overline{DTACK}$ LOW from $\overline{IACK}$ LOW	see Fig.20	–	$2t_{CLK}$ + 75	$3t_{CLK}$ + 150	ns
t_{AHDE}	$\overline{DTACK}$ HIGH from $\overline{IACK}$ HIGH	see Fig.20	–	120	140	ns
t_{W4}	$\overline{RESET}$ pulse width	see Fig.21	$30t_{CLK}$	–	–	ns
t_{W5}	$\overline{STROBE}$ pulse width	see Fig.22	$8t_{CLK}$	$8t_{CLK}$ + 90	–	ns
t_{CLCL}	$\overline{CS}$ LOW	see Figs 17 and 18	–	$t_{CLDL} + t_{CHDE}$	–	ns

Notes

1. A minimum of 6 clock cycles must elapse between consecutive parallel-bus accesses when the I²C-bus controller operates at 8 or 12 MHz. This may be reduced to 3 clock cycles for lower operating frequencies.
2. After reset the chip clock default is 12 MHz.
3. Not for S1.
4. Not tested.

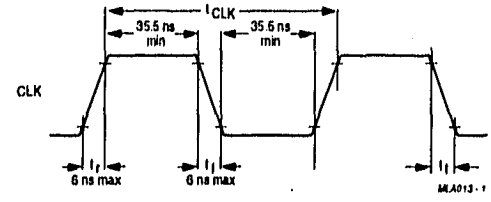


Fig.14 Clock input timing.

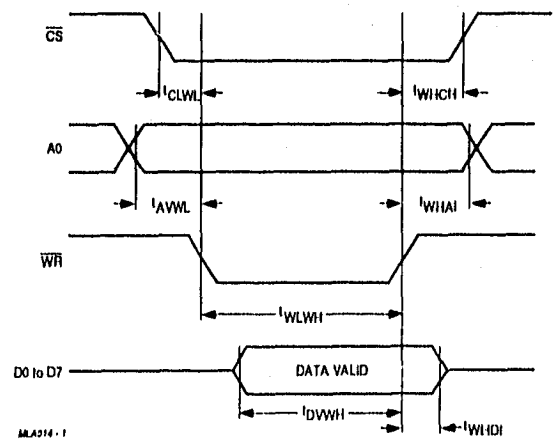


Fig.15 Bus timing (80XX mode); write cycle.

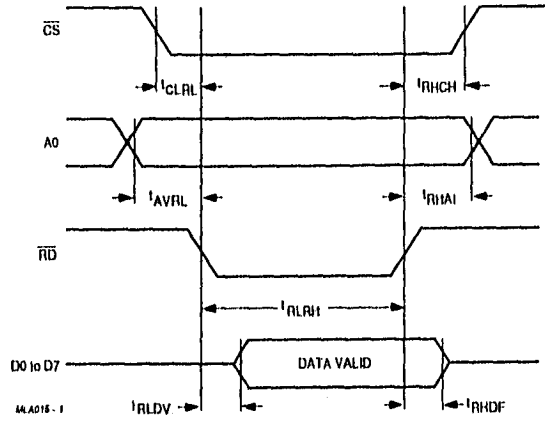


Fig.16 Bus timing (80XX mode); read cycle.

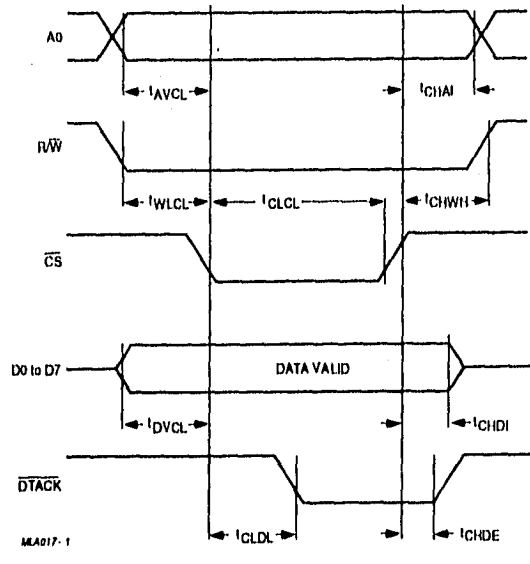


Fig.17 Bus timing (68000 mode); write cycle.

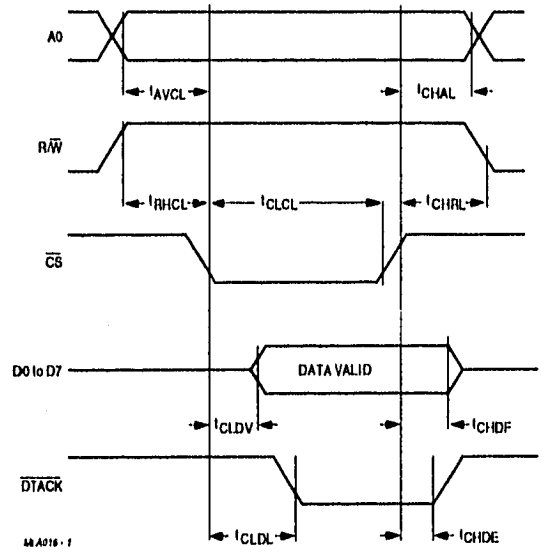


Fig.18 Bus timing (68000 mode); read cycle.

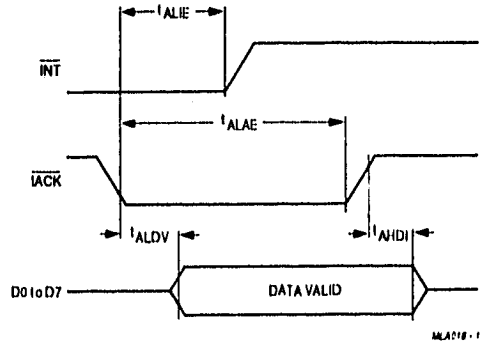


Fig.19 Interrupt timing (80XX mode).

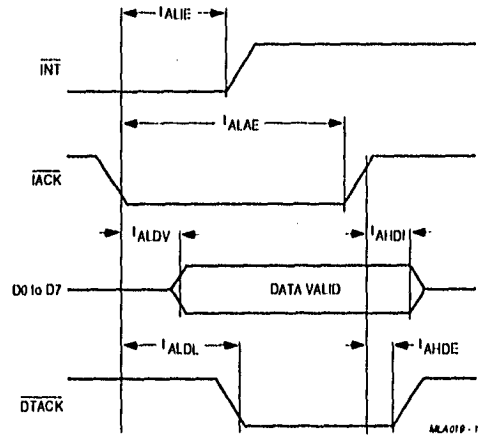


Fig.20 Interrupt timing (68000 mode).

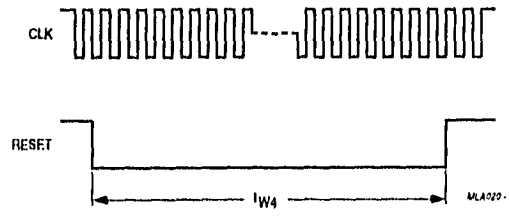


Fig.21 Reset timing.

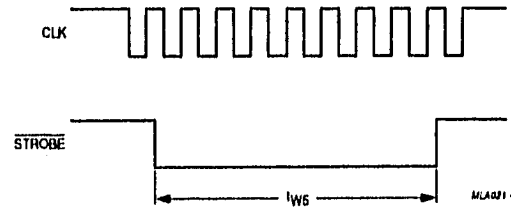


Fig.22 Strobe timing.

14 APPLICATION INFORMATION

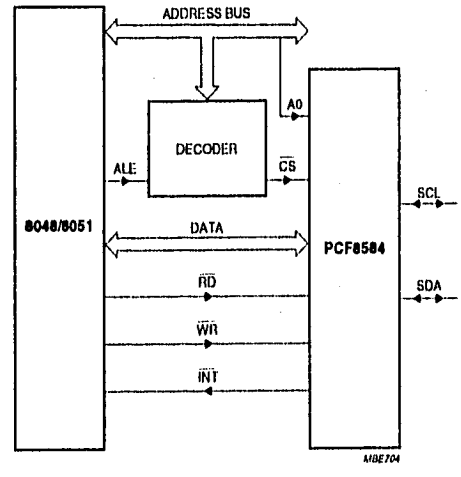


Fig.23 Application diagram using the 8048/8051.

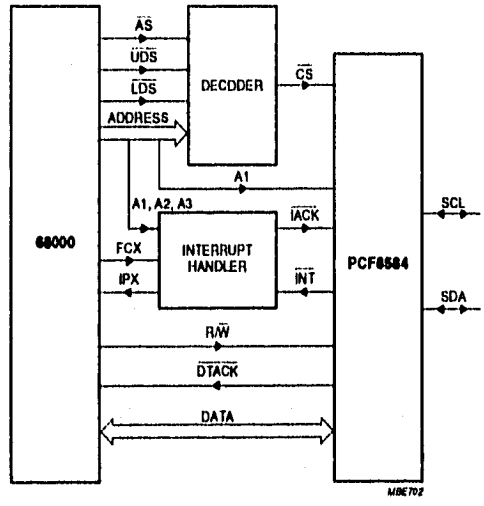


Fig.24 Application diagram using the 68000.

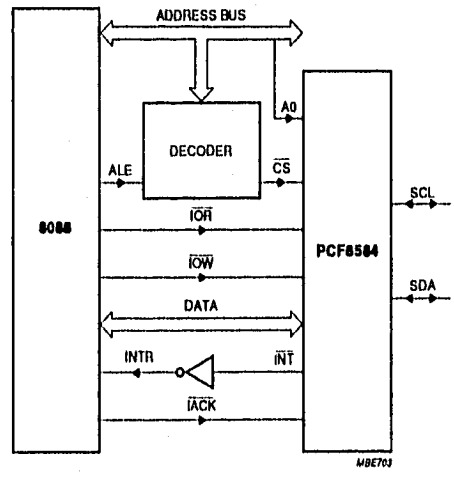
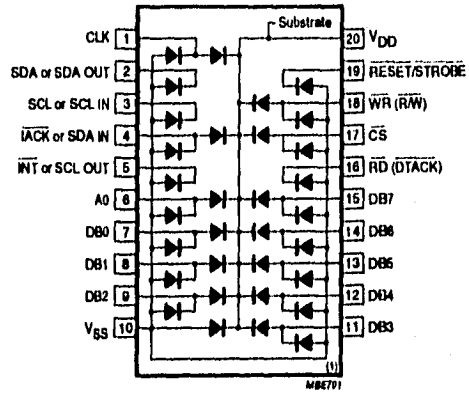


Fig.25 Application diagram using the 8088.

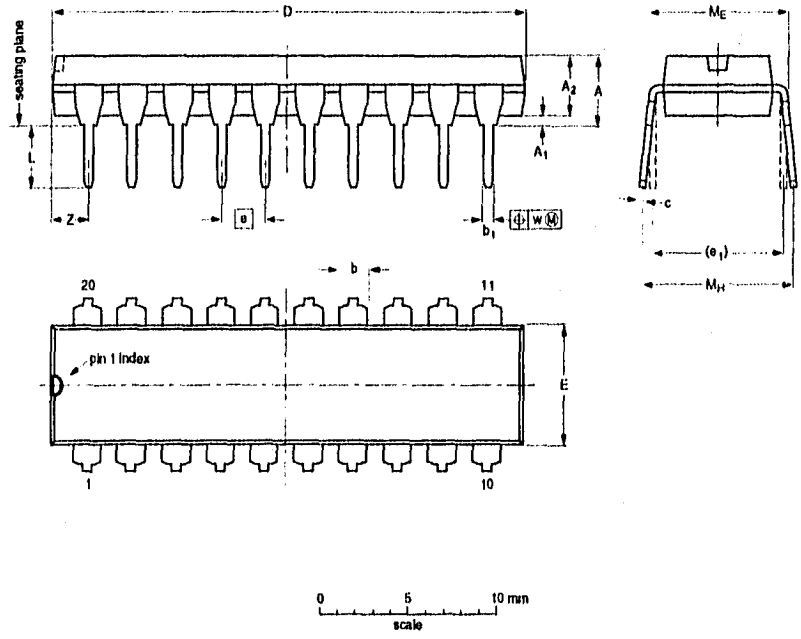


Maximum forward current: 5 mA; maximum reverse voltage: 5 V.

Fig.26 PCF8584 diode protection.

15 PACKAGE OUTLINES

DIP20: plastic dual in-line package; 20 leads (300 mil) **SOT146-1**



DIMENSIONS (inch dimensions are derived from the original mm dimensions)

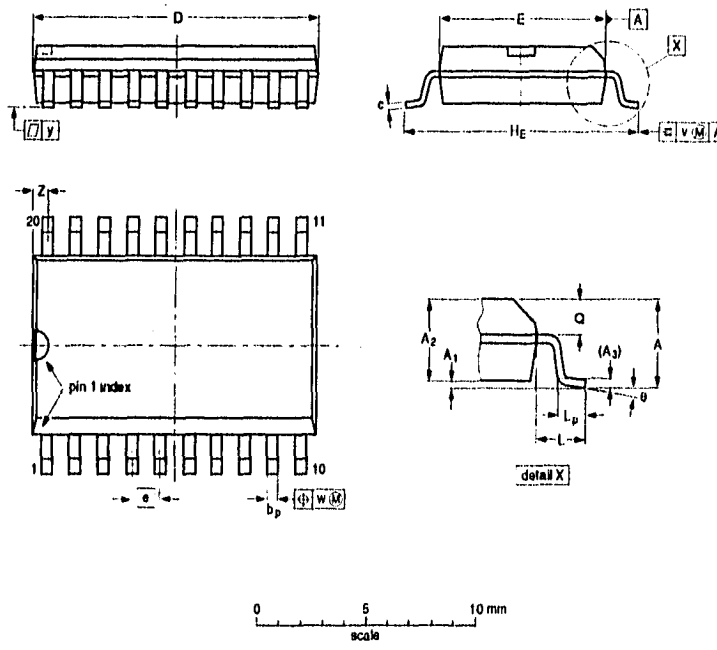
UNIT	A max.	A_1 min.	A_2 max.	b	b_1	c	D[1]	E[1]	e	e_1	L	M_E	M_H	w	Z[1] max.
mm	4.2	0.51	3.2	1.73 1.30	0.53 0.38	0.36 0.23	26.92 26.54	6.40 6.22	2.54	7.62	3.60 3.05	8.25 7.80	10.0 8.3	0.254	2.0
inches	0.17	0.020	0.13	0.068 0.051	0.021 0.015	0.014 0.009	1.060 1.045	0.25 0.24	0.10	0.30	0.14 0.12	0.32 0.31	0.39 0.33	0.01	0.078

Note

1. Plastic or metal protrusions of 0.25 mm maximum per side are not included.

OUTLINE VERSION	REFERENCES			EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ		
SOT146-1			SC603		92-11-17 95-05-24

SO20: plastic small outline package; 20 leads; body width 7.5 mm **SOT163-1**



DIMENSIONS (inch dimensions are derived from the original mm dimensions)

UNIT	A max.	A_1	A_2	A_3	b_p	c	$D^{(1)}$	$E^{(1)}$	e	H_E	L	L_p	Q	v	w	y	$Z^{(1)}$	θ
mm	2.65	0.30 0.10	2.45 2.25	0.25	0.49 0.36	0.32 0.23	13.0 12.6	7.6 7.4	1.27	10.65 10.00	1.4	1.1 0.4	1.1 1.0	0.25	0.25	0.1	0.9 0.4	0° 8°
inches	0.10	0.012 0.004	0.096 0.089	0.01	0.019 0.014	0.013 0.009	0.51 0.49	0.30 0.29	0.050	0.42 0.39	0.055	0.043 0.016	0.043 0.039	0.01	0.01	0.004	0.035 0.016	

Note

1. Plastic or metal protrusions of 0.15 mm maximum per side are not included.

OUTLINE VERSION	REFERENCES			EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ		
SOT163-1	075E04	MS-013AC			92-11-17 95-01-24

16 SOLDERING

16.1 Introduction

There is no soldering method that is ideal for all IC packages. Wave soldering is often preferred when through-hole and surface mounted components are mixed on one printed-circuit board. However, wave soldering is not always suitable for surface mounted ICs, or for printed-circuits with high population densities. In these situations reflow soldering is often used.

This text gives a very brief insight to a complex technology. A more in-depth account of soldering ICs can be found in our *"IC Package Databook"* (order code 9398 652 90011).

16.2 DIP

16.2.1 SOLDERING BY DIPPING OR BY WAVE

The maximum permissible temperature of the solder is 260 °C; solder at this temperature must not be in contact with the joint for more than 5 seconds. The total contact time of successive solder waves must not exceed 5 seconds.

The device may be mounted up to the seating plane, but the temperature of the plastic body must not exceed the specified maximum storage temperature ($T_{stg\ max}$). If the printed-circuit board has been pre-heated, forced cooling may be necessary immediately after soldering to keep the temperature within the permissible limit.

16.2.2 REPAIRING SOLDERED JOINTS

Apply a low voltage soldering iron (less than 24 V) to the lead(s) of the package, below the seating plane or not more than 2 mm above it. If the temperature of the soldering iron bit is less than 300 °C it may remain in contact for up to 10 seconds. If the bit temperature is between 300 and 400 °C, contact may be up to 5 seconds.

16.3 SO

16.3.1 REFLOW SOLDERING

Reflow soldering techniques are suitable for all SO packages.

Reflow soldering requires solder paste (a suspension of fine solder particles, flux and binding agent) to be applied to the printed-circuit board by screen printing, stencilling or pressure-syringe dispensing before package placement.

Several techniques exist for reflowing; for example, thermal conduction by heated belt. Dwell times vary between 50 and 300 seconds depending on heating method. Typical reflow temperatures range from 215 to 250 °C.

Preheating is necessary to dry the paste and evaporate the binding agent. Preheating duration: 45 minutes at 45 °C.

16.3.2 WAVE SOLDERING

Wave soldering techniques can be used for all SO packages if the following conditions are observed:

- A double-wave (a turbulent wave with high upward pressure followed by a smooth laminar wave) soldering technique should be used.
- The longitudinal axis of the package footprint must be parallel to the solder flow.
- The package footprint must incorporate solder thieves at the downstream end.

During placement and before soldering, the package must be fixed with a droplet of adhesive. The adhesive can be applied by screen printing, pin transfer or syringe dispensing. The package can be soldered after the adhesive is cured.

Maximum permissible solder temperature is 260 °C, and maximum duration of package immersion in solder is 10 seconds, if cooled to less than 150 °C within 6 seconds. Typical dwell time is 4 seconds at 250 °C.

A mildly-activated flux will eliminate the need for removal of corrosive residues in most applications.

16.3.3 REPAIRING SOLDERED JOINTS

Fix the component by first soldering two diagonally-opposite end leads. Use only a low voltage soldering iron (less than 24 V) applied to the flat part of the lead. Contact time must be limited to 10 seconds at up to 300 °C. When using a dedicated tool, all other leads can be soldered in one operation within 2 to 5 seconds between 270 and 320 °C.

17 DEFINITIONS

Data sheet status	
Objective specification	This data sheet contains target or goal specifications for product development.
Preliminary specification	This data sheet contains preliminary data; supplementary data may be published later.
Product specification	This data sheet contains final product specifications.
Limiting values	
Limiting values given are in accordance with the Absolute Maximum Rating System (IEC 134). Stress above one or more of the limiting values may cause permanent damage to the device. These are stress ratings only and operation of the device at these or at any other conditions above those given in the Characteristics sections of the specification is not implied. Exposure to limiting values for extended periods may affect device reliability.	
Application information	
Where application information is given, it is advisory and does not form part of the specification.	

18 LIFE SUPPORT APPLICATIONS

These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Philips customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Philips for any damages resulting from such improper use or sale.

19 PURCHASE OF PHILIPS I²C COMPONENTS

Purchase of Philips I²C components conveys a license under the Philips' I²C patent to use the components in the I²C system provided the system conforms to the I²C specification defined by Philips. This specification can be ordered using the code 9398 393 40011.

NOTES

Philips Semiconductors – a worldwide company

Argentina: IEROD, Av. Juramento 1992 - 14.b, (1428)
BUENOS AIRES, Tel. (541)786 7633, Fax. (541)786 9367

Australia: 34 Waterloo Road, NORTH RYDE, NSW 2113,
Tel. (02)805 4455, Fax. (02)805 4460

Austria: Triester Str. 64, A-1101 WIEN, P.O. Box 213,
Tel. (01)60 101-1236, Fax. (91)60 101-1211

Belgium: Postbus 90050, 5600 PB EINDHOVEN, The Netherlands,
Tel. (31)40 783 749, Fax. (31)40 788 399

Brazil: Rua do Rocio 220 - 5th floor, Suite 51,
CEP: 04552-903-SÃO PAULO-SP, Brazil.
P.O. Box 7383 (01064-970),
Tel. (011)821-2333, Fax. (011)829-1649

Canada: PHILIPS SEMICONDUCTORS/COMPONENTS:
Tel. (800) 234-7381, Fax. (708) 296-8556

Chile: Av. Santa Maria 9760, SANTIAGO,
Tel. (02)773 816, Fax. (02)777 6730

China/Hong Kong: 501 Hong Kong Industrial Technology Centre,
72 Tat Chee Avenue, Kowloon Tong, HONG KONG,
Tel. (852)2319 7888, Fax. (852)2319 7700

Colombia: IPRELENSO LTDA, Carrara 21 No. 66-17,
77621 BOGOTA, Tel. (571)249 7624/(571)217 4609,
Fax. (571)217 4549

Denmark: Prags Boulevard 80, PB 1919, DK-2300
COPENHAGEN S, Tel. (032)88 2636, Fax. (031)57 1949

Finland: Sinikalliontie 3, FIN-02630 ESPOO,
Tel. (358)0-615 800, Fax. (358)0-81580 920

France: 4 Rue du Port-aux-Vins, BP317,
92156 SURESNES Cedex,
Tel. (01)4099 6161, Fax. (01)4099 6427

Germany: P.O. Box 10 63 23, 20043 HAMBURG,
Tel. (040)3296-0, Fax. (040)3296 213.

Greece: No. 15, 25th March Street, GR 17778 TAVROS,
Tel. (01)4894 339/4894 911, Fax. (01)4814 240

India: Philips INDIA Ltd, Shivsagar Estate, A Block,
Dr. Annie Besant Rd. Worli, Bombay 400 019
Tel. (022)4938 541, Fax. (022)4938 722

Indonesia: Philips House, Jalan H.R. Rasuna Said Kav. 3-4,
P.O. Box 4252, JAKARTA 12950,
Tel. (021)5201 122, Fax. (021)5205 189

Ireland: Newstead, Clonskeagh, DUBLIN 14,
Tel. (01)7640 000, Fax. (01)7640 200

Italy: PHILIPS SEMICONDUCTORS S.r.l.,
Piazza IV Novembre 3, 20124 MILANO,
Tel. (0039)2 6752 2531, Fax. (0039)2 6752 2557

Japan: Philips Bldg 13-37, Kohnan 2-chome, Minato-ku, TOKYO 108,
Tel. (03)3740 5130, Fax. (03)3740 5077

Korea: Philips House, 260-199 Itaewon-dong,
Yongsan-ku, SEOUL, Tel. (02)709-1412, Fax. (02)709-1415

Malaysia: No. 76 Jalan Universiti, 46200 PETALING JAYA,
SELANGOR, Tel. (03)750 5214, Fax. (03)757 4880

Mexico: 5900 Gateway East, Suite 200, EL PASO, TX 79905,
Tel. 9-5(800)234-7381, Fax. (708)296-8556

Netherlands: Postbus 90050, 5600 PB EINDHOVEN, Bldg. VB,
Tel. (040)783749, Fax. (040)788399
(From 10-10-1995: Tel. (040)2783749, Fax. (040)2788399)

New Zealand: 2 Wagener Place, C.P.O. Box 1041, AUCKLAND,
Tel. (00)849-4160, Fax. (09)849-7811

Norway: Box 1, Manglerud 0612, OSLO,
Tel. (022)74 0000, Fax. (022)74 8341

Pakistan: Philips Electrical Industries of Pakistan Ltd.,
Exchange Bldg. ST-2/A, Block 9, KDA Scheme 5, Clifton,
KARACHI 75600, Tel. (021)587 4641-49,
Fax. (021)5770357/5874546

Philippines: PHILIPS SEMICONDUCTORS PHILIPPINES Inc,
106 Valero St. Salcedo Village, P.O. Box 2108 MCC, MAKATI,
Metro MANILA, Tel. (02)810 0161, Fax. (02)817 3474

Portugal: PHILIPS PORTUGUESA, S.A.,
Rua dr. António Loureiro Borges 5, Arquiparque - Miraflores,
Apartado 300, 2795 LINDA-A-VELHA,
Tel. (01)4163160/4163333, Fax. (01)4163174/4163366

Singapore: Lorong 1, Toa Payoh, SINGAPORE 1231,
Tel. (65)350 2000, Fax. (65)251 6500

South Africa: S.A. PHILIPS Pty Ltd.,
195-215 Main Road Martindale, 2092 JOHANNESBURG,
P.O. Box 7430, Johannesburg 2000,
Tel. (011)470-5911, Fax. (011)470-5494.

Spain: Balmes 22, 08007 BARCELONA,
Tel. (03)301 6312, Fax. (03)301 42 43

Sweden: Kottbygatan 7, Akalla, S-164 85 STOCKHOLM,
Tel. (0)8-632 2000, Fax. (0)8-632 2745

Switzerland: Allmendstrasse 140, CH-8027 ZÜRICH,
Tel. (01)488 2211, Fax. (01)481 77 30

Taiwan: PHILIPS TAIWAN Ltd., 23-30F, 66, Chung Hsiao West
Road, Sec. 1. Taipeh, Taiwan ROC, P.O. Box 22978,
TAIPEI 100, Tel. (02)388 7666, Fax. (02)382 4382

Thailand: PHILIPS ELECTRONICS (THAILAND) Ltd.,
209/2 Sanpavuth-Bangna Road Prakanong,
Bangkok 10260, THAILAND,
Tel. (662)390-0141, Fax. (662)398-3319

Turkey: Talatpasa Cad. No. 5, 80640 GÜLTEPE/ISTANBUL,
Tel. (0212)279 27 70, Fax. (0212)282 67 07

United Kingdom: Philips Semiconductors LTD.,
276 Bath Road, Hayes, MIDDLESEX UB3 5BX,
Tel. (0181)730-5000, Fax. (0181)754-8421

United States: 811 East Arques Avenue, SUNNYVALE,
CA 94088-3409, Tel. (800)234-7381, Fax. (708)296-8556

Uruguay: Coronel Mora 433, MONTEVIDEO,
Tel. (02)70-4044, Fax. (02)92 0601

Internet: http://www.semiconductors.philips.com/ps/

For all other countries apply to: Philips Semiconductors,
International Marketing and Sales, Building BE-p,
P.O. Box 218, 5600 MD EINDHOVEN, The Netherlands,
Telex 35000 phtcnl, Fax. +31-40-724825 (from 10-10-1995: +31-40-2724825)

Printed in The Netherlands

s9306l/1500/02/pp40 Date of release: 1995 Aug 29
Document order number: 9397 750 00277

Philips Semiconductors

PHILIPS

Single-chip 8-bit microcontroller — 80C552/83C552

Single-chip 8-bit microcontroller with 10-bit A/D, capture/compare timer, high-speed outputs, PWM

DESCRIPTION

The 80C552/83C552 (hereafter generically referred to as 8XC552) Single-Chip 8-Bit Microcontroller is manufactured in an advanced CMOS process and is a derivative of the 80C51 microcontroller family. The 8XC552 has the same instruction set as the 80C51. Three versions of the derivative exist:

- 83C552—8k bytes mask programmable ROM
- 80C552—ROMless version of the 83C552
- 87C552—8k bytes EPROM (described in a separate chapter)

The 8XC552 contains a non-volatile 8k × 8 read-only program memory (83C552), a volatile 256 × 8 read/write data memory, five 8-bit I/O ports, one 8-bit input port, two 16-bit timer/event counters (identical to the timers of the 80C51), an additional 16-bit timer coupled to capture and compare latches, a 15-source, two-priority-level, nested interrupt structure, an 8-input ADC, a dual DAC pulse width modulated interface, two serial interfaces (UART and I²C-bus), a "watchdog" timer and on-chip oscillator and timing circuits. For systems that require extra capability, the 8XC552 can be expanded using standard TTL compatible memories and logic.

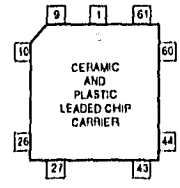
In addition, the 8XC552 has two software selectable modes of power reduction—idle mode and power-down mode. The idle mode freezes the CPU while allowing the RAM, timers, serial ports, and interrupt system to continue functioning. The power-down mode saves the RAM contents but freezes the oscillator, causing all other chip functions to be inoperative.

The device also functions as an arithmetic processor having facilities for both binary and BCD arithmetic plus bit-handling capabilities. The instruction set consists of over 100 instructions: 49 one-byte, 45 two-byte, and 17 three-byte. With a 16MHz (24MHz) crystal, 58% of the instructions are executed in 0.75µs (0.5µs) and 40% in 1.5µs (1µs). Multiply and divide instructions require 3µs (2µs).

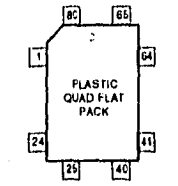
FEATURES

- 80C51 central processing unit
- 8k × 8 ROM expandable externally to 64k bytes
- An additional 16-bit timer/counter coupled to four capture registers and three compare registers
- Two standard 16-bit timer/counters
- 256 × 8 RAM, expandable externally to 64k bytes
- Capable of producing eight synchronized, timed outputs
- A 10-bit ADC with eight multiplexed analog inputs
- Two 8-bit resolution, pulse width modulation outputs
- Five 8-bit I/O ports plus one 8-bit input port shared with analog inputs
- I²C-bus serial I/O port with byte oriented master and slave functions
- Full-duplex UART compatible with the standard 80C51
- On-chip watchdog timer
- Three speed ranges:
  - 1.2 to 16MHz
  - 1.2 to 24MHz (ROM, ROMless only)
  - 1.2 to 30MHz (ROM, ROMless only)
- Three operating ambient temperature ranges:
  - PCB83C552-5: 0°C to +70°C
  - PCF83C552-5: –40°C to +85°C (XTAL frequency max. 24 MHz)
  - PCA83C552-5: –40°C to +125°C (XTAL frequency max. 16 MHz)

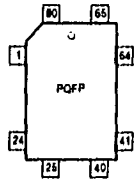
PIN CONFIGURATIONS



Pin	Function	Pin	Function
1	P5.0/ADC8	35	XTAL1
2	V_{DD}	36	V_{SS}
3	STADC	37	V_{SS}
4	PWM0	38	NC
5	PWM1	39	P2.0/A08
6	EW	40	P2.1/A09
7	P4.0/CMSR0	41	P2.2/A10
8	P4.1/CMSR1	42	P2.3/A11
9	P4.2/CMSR2	43	P2.4/A12
10	P4.3/CMSR3	44	P2.5/A13
11	P4.4/CMSR4	45	P2.6/A14
12	P4.5/CMSR5	46	P2.7/A15
13	P4.6/CMT0	47	PSEN
14	P4.7/CMT1	48	ALE
15	RST	49	EA
16	P1.0/CT0I	50	P0.7/AD7
17	P1.1/CT1I	51	P0.6/AD6
18	P1.2/CT2I	52	P0.5/AD5
19	P1.3/CT3I	53	P0.4/AD4
20	P1.4/T2	54	P0.3/AD3
21	P1.5/RT2	55	P0.2/AD2
22	P1.6/SCL	56	P0.1/AD1
23	P1.7/SDA	57	P0.0/AD0
24	P3.0/RxD	58	AVref-
25	P3.1/TxD	59	AVref+
26	P3.2/INT0	60	AV_{SS}
27	P3.3/INT1	61	AV_{DD}
28	P3.4/T0	62	P5.7/ADC7
29	P3.5/T1	63	P5.6/ADC6
30	P3.6/WR	64	P5.5/ADC5
31	P3.7/RD	65	P5.4/ADC4
32	NC	66	P5.3/ADC3
33	NC	67	P5.2/ADC2
34	XTAL2	68	P5.1/ADC1



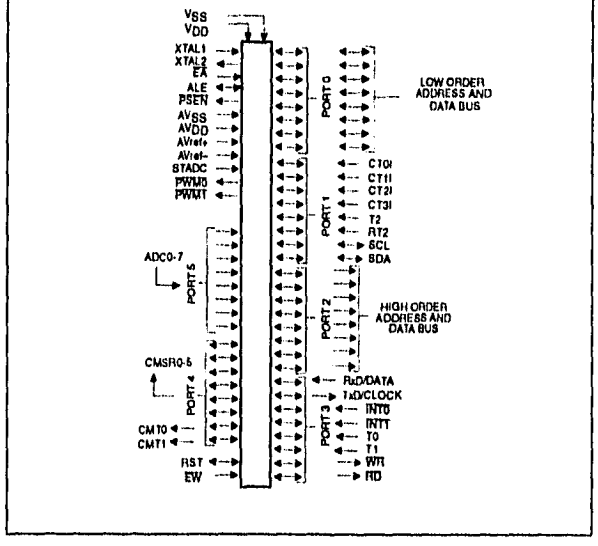
PLASTIC QUAD FLAT PACK PIN FUNCTIONS



Pin	Function	Pin	Function
1	P4.1/CMSR1	41	P2.3/A11
2	P4.2/CMSR2	42	P2.4/A12
3	NC	43	NC
4	P4.3/CMSR3	44	NC
5	P4.4/CMSR4	45	P2.5/A13
6	P4.5/CMSR5	46	P2.6/A14
7	P4.6/CMT0	47	P2.7/A15
8	P4.7/CMT1	48	PSEN
9	RST	49	ALE
10	P1.0/CT0I	50	EA
11	P1.1/CT1I	51	P0.7/AD7
12	P1.2/CT2I	52	P0.6/AD6
13	P1.3/CT3I	53	P0.5/AD5
14	P1.4/T2	54	P0.4/AD4
15	P1.5/T2T2	55	P0.3/AD3
16	P1.6/SCL	56	P0.2/AD2
17	P1.7/SDA	57	P0.1/AD1
18	P3.0/RxD	58	P0.0/AD0
19	P3.1/TxD	59	AVref-
20	P3.2/INT0	60	AVref+
21	NC	61	AV_{SS}
22	NC	62	NC
23	P3.3/INT1	63	AV_{DD}
24	P3.4/T0	64	P5.7/ADC7
25	P3.5/T1	65	P5.6/ADC6
26	P3.6/WR	66	P5.5/ADC5
27	P3.7/RD	67	P5.4/ADC4
28	NC	68	P5.3/ADC3
29	NC	69	P5.2/ADC2
30	NC	70	P5.1/ADC1
31	XTAL2	71	P5.0/ADC0
32	XTAL1	72	V_{DD}
33	IC	73	IC
34	V_{SS}	74	STADC
35	V_{SS}	75	PWM0
36	V_{SS}	76	PWM1
37	NC	77	EW
38	P2.0/A08	78	NC
39	P2.1/A09	79	NC
40	P2.2/A10	80	P4.0/CMSR0

NC = not connected
IC = internally connected (do not use)

LOGIC SYMBOL



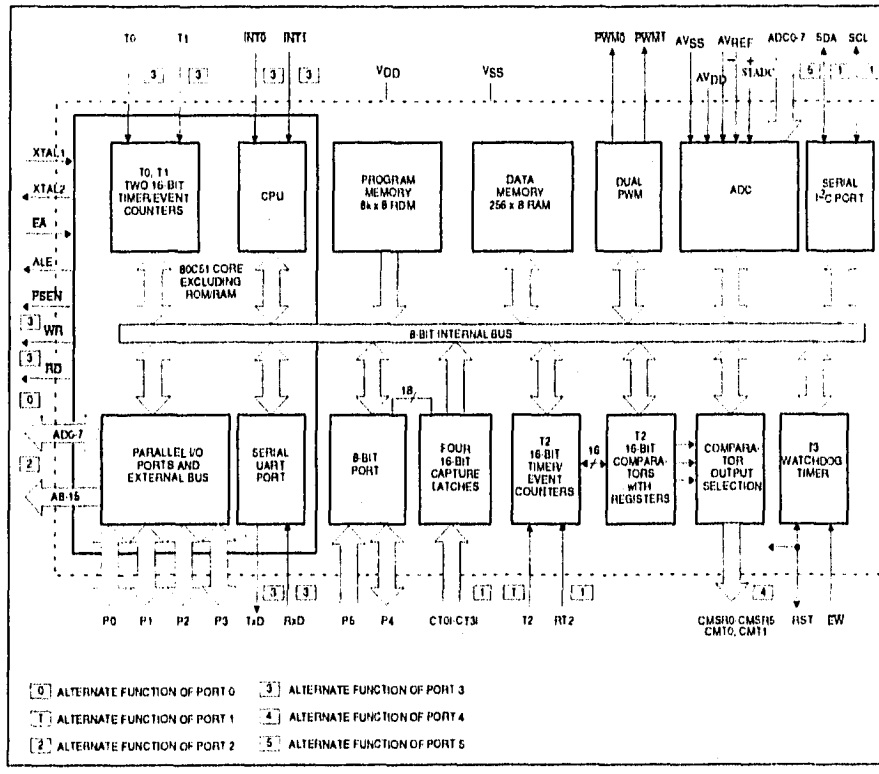
ORDERING INFORMATION

PHILIPS PART ORDER NUMBER PART MARKING		NORTH AMERICA PHILIPS PART ORDER NUMBER		DRAWING	TEMPERATURE °C	FREQ
ROMless	ROM	ROMless	ROM	NUMBER	AND PACKAGE	MHz
PCB80C552-5-16WP	PCB83C552-5WP/xxx	S80C552-4A68	S83C552-4A68	SOT188	0 to +70, Plastic Leaded Chip Carrier	16
PCB80C552-5-16H	PCB83C552-5H/xxx	S80C552-4B	S83C552-4B	SOT318	0 to +70, Plastic Quad Flat Pack	16
PCF80C552-5-16WP	PCF83C552-5WP/xxx	S80C552-5A68	S83C552-5A68	SOT188	–40 to +85, Plastic Leaded Chip Carrier	16
PCF80C552-5-16H	PCF83C552-5H/xxx	S80C552-5B	S83C552-5B	SOT318	–40 to +85, Plastic Quad Flat Pack	16
PCA80C552-5-16WP	PCA83C552-5WP/xxx	S80C552-6A68	S83C552-6A68	SOT188	–40 to +125, Plastic Leaded Chip Carrier	16
PCA80C552-5-16H	PCA83C552-5H/xxx	S80C552-6B	S63C552-8B	SOT318	–40 to +125, Plastic Quad Flat Pack	16
PCB80C552-5-24WP	PCB83C552-5WP/xxx	S80C552-AA68	S83C552-AA68	SOT188	0 to +70, Plastic Leaded Chip Carrier	24
PCB80C552-5-24H	PCB83C552-5H/xxx	S80C552-AB	S83C552-AB	SOT318	0 to +70, Plastic Quad Flat Pack	24
PCF80C552-5-24WP	PCF83C552-5WP/xxx	S80C552-BA58	S83C552-BA68	SOT188	–40 to +85, Plastic Leaded Chip Carrier	24
PCF80C552-5-24H	PCF83C552-5H/xxx	S80C552-BB	S63C552-BB	SOT316	–40 to +85, Plastic Quad Flat Pack	24
PCB80C552-5-30WP	PCB83C552-5WP/xxx	S80C552-CA68	S83C552-CA68	SOT188	0 to +70, Plastic Leaded Chip Carrier	30
PCB80C552-5-30H	PCB83C552-5H/xxx	S80C552-CB	S83C552-CB	SOT318	0 to +70, Plastic Quad Flat Pack	30

NOTE:
1. xxx denotes the ROM code number.

EPROM	DRAWING NUMBER	TEMPERATURE °C AND PACKAGE	FREQ MHz
S87C552-4A68	0398E	0 to +70, Plastic Leaded Chip Carrier	16
S87C552-4K68	1473A	0 to +70, Ceramic Leaded Chip Carrier w/Window	16
S87C552-4B	SOT318	0 to +70, Plastic Quad Flat Pack	16
S87C552-5A68	0398E	–40 to +85, Plastic Leaded Chip Carrier	16
S87C552-5K68	1473A	–40 to +85, Ceramic Leaded Chip Carrier w/Window	16
S87C552-5B	SOT318	–40 to +85, Plastic Quad Flat Pack	16

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. PLCC	PIN NO. QFP	TYPE	NAME AND FUNCTION
V_{DD}	2	72	I	**Digital Power Supply:** +5V power supply pin during normal operation, idle and power-down mode.
STADC	3	74	I	**Start ADC Operation:** Input starting analog to digital conversion (ADC operation can also be started by software). This pin must not float.
PWM0	4	75	O	**Pulse Width Modulation:** Output 0.
PWM1	5	76	O	**Pulse Width Modulation:** Output 1.
EW	6	77	I	**Enable Watchdog Timer:** Enable for T3 watchdog timer and disable power-down mode. This pin must not float.
P0.0-P0.7	57-50	58-51	I/O	**Port 0:** Port 0 is an 8-bit open-drain bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application it uses strong internal pull-ups when emitting 1s.
P1.0-P1.7	16-23	10-17	I/O	**Port 1:** 8-bit I/O port. Alternate functions include:
	18-21	10-15	I/O	**(P1.0-P1.5):** Quasi-bidirectional port pins.
	22-23	16-17	I/O	**(P1.6, P1.7):** Open drain port pins.
	16-19	10-13	I	**CT0I-CT3I (P1.0-P1.3):** Capture timer input signals for timer T2.
	20	14	I	**T2 (P1.4):** T2 event input.
	21	15	I	**RT2 (P1.5):** T2 timer reset signal. Rising edge triggered.
	22	18	I/O	**SCL (P1.6):** Serial port clock line I^2C-bus.
	23	17	I/O	**SDA (P1.7):** Serial port data line I^2C-bus.
				Port 1 is also used to input the lower order address byte during EPROM programming and verification. A0 is on P1.0, etc.
P2.0-P2.7	39-46	38-42, 45-47	I/O	**Port 2:** 6-bit quasi-bidirectional I/O port. Alternate function: High-order address byte for external memory (A08-A15).
P3.0-P3.7	24-31	18-20, 23-27	I/O	**Port 3:** 8-bit quasi-bidirectional I/O port. Alternate functions include:
	24	18		**RxD(P3.0):** Serial input port.
	25	19		**TxD (P3.1):** Serial output port.
	26	20		**INT0 (P3.2):** External interrupt.
	27	23		**INT1 (P3.3):** External interrupt.
	28	24		**T0 (P3.4):** Timer 0 external input.
	29	25		**T1 (P3.5):** Timer 1 external input.
	30	26		**WR (P3.6):** External data memory write strobe.
	31	27		**RD (P3.7):** External data memory read strobe.
P4.0-P4.7	7-14	80, 1-2 4-8	I/O	**Port 4:** 6-bit quasi-bidirectional I/O port. Alternate functions include:
	7-12	80, 1-2 4-6	O	**CMSR0-CMSR5 (P4.0-P4.5):** Timer T2 compare and set/reset outputs on a match with timer T2.
	13, 14	7, 8	O	**CMT0, CMT1 (P4.6, P4.7):** Timer T2 compare and toggle outputs on a match with timer T2.
P5.0-P5.7	68-62, 1	71-64,	I	**Port 5:** 8-bit input port. **ADC0-ADC7 (P5.0-P5.7):** Alternate function: Eight input channels to ADC.
RST	15	9	I/O	**Reset:** Input to reset the 8XC552. It also provides a reset pulse as output when timer T3 overflows.
XTAL1	35	32	I	**Crystal Input 1:** Input to the inverting amplifier that forms the oscillator, and input to the internal clock generator. Receives the external clock signal when an external oscillator is used.
XTAL2	34	31	O	**Crystal Input 2:** Output of the inverting amplifier that forms the oscillator. Left open-circuit when an external clock is used.

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO. PLCC	PIN NO. QFP	TYPE	NAME AND FUNCTION
V_{SS}	36, 37	34-36	I	**Two Digital ground pins.**
PSEN	47	48	O	**Program Store Enable:** Active-low read strobe to external program memory.
ALE	48	49	O	**Address Latch Enable:** Latches the low byte of the address during accesses to external memory. It is activated every six oscillator periods. During an external data memory access, one ALE pulse is skipped. ALE can drive up to eight LS TTL inputs and handles CMOS inputs without an external pull-up.
EA	49	50	I	**External Access:** When EA is held at TTL level high, the CPU executes out of the internal program ROM provided the program counter is less than 8192. When EA is held at TTL low level, the CPU executes out of external program memory. EA is not allowed to float.
AV_{REF-}	58	59	I	**Analog to Digital Conversion Reference Resistor:** Low-end.
AV_{REF+}	59	60	I	**Analog to Digital Conversion Reference Resistor:** High-end.
AV_{SS}	60	61	I	**Analog Ground**
AV_{DD}	61	63	I	**Analog Power Supply**

NOTE:
1. To avoid "latch-up" effect at power-on, the voltage on any pin at any time must not be higher or lower than V_{DD} + 0.5 V or V_{SS} – 0.5V, respectively.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the logic symbol, page 2.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

RESET

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running. To insure a good power-on reset, the RST pin must be high long enough to allow the oscillator time to start up (normally a few milliseconds) plus two machine cycles. At power-on, the voltage on V_{DD} and RST must come up at the same time for a proper start-up.

IDLE MODE

In the idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

POWER-DOWN MODE

In the power-down mode, the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. Only the contents of the on-chip RAM are preserved. A hardware reset is the only way to terminate the power-down mode. The control bits for the reduced power modes are in the special function register PCON. Table 1 shows the state of the I/O ports during low current operating modes.

Table 1. External Pin Status During Idle and Power-Down Modes

MODE	PROGRAM MEMORY	ALE	PSEN	PORT 0	PORT 1	PORT 2	PORT 3	PORT 4	PWM0/ PWM1
Idle	Internal	1	1	Data	Data	Data	Data	Data	1
Idle	External	1	1	Float	Data	Address	Data	Data	1
Power-down	Internal	0	0	Data	Data	Data	Data	Data	1
Power-down	External	0	0	Float	Data	Data	Data	Data	1

Serial Control Register (S1CON) – See Table 2

S1CON (D8H)	CR2	ENS1	STA	STO	SI	AA	CR1	CR0

Bits CR0, CR1 and CR2 determine the serial clock frequency that is generated in the master mode of operation.

Table 2. Serial Clock Rates

CR2	CR1	CR0	BIT FREQUENCY (kHz) AT f_{osc} 6MHz	12MHz	16MHz	24MHz[2]	30MHz[2]	f_{osc} DIVIDED BY
0	0	0	23	47	62.5	94	117 [1]	256
0	0	1	27	54	71	107 [1]	134 [1]	224
0	1	0	31	63	83.3	125 [1]	155 [1]	192
0	1	1	37	75	100	150 [1]	188 [1]	160
1	0	0	6.25	12.5	17	25	31	960
1	0	1	50	100	133 [1]	200 [1]	250 [1]	120
1	1	0	100	200	267 [1]	400 [1]	500 [1]	60
1	1	1	0.24 < 62.5	0.49 < 62.5	0.65 < 65.6	0.98 < 50.0	1.22 < 52.1	96 × (256 – (reload value Timer 1))
			0 < 255	0 < 254	0 < 253	0 <251	0 < 250	reload value Timer 1 in Mode 2.

NOTES:
1. These frequencies exceed the upper limit of 100kHz of the I²C-bus specification and cannot be used in an I²C-bus application.
2. At f_{OSC} = 24MHz/ 30MHz the maximum I²C bus rate of 100kHz cannot be realized due to the fixed divider rates.

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Storage temperature range	–65 to +150	°C
Voltage on any other pin to V_{SS}	–0.5 to +6.5	V
Input, output DC current on any single I/O pin	5.0	mA
Power dissipation (based on package heat transfer limitations, not device power consumption)	1.0	W

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DEVICE SPECIFICATIONS

TYPE	SUPPLY VOLTAGE (V) MIN	MAX	FREQUENCY (MHz) MIN	MAX	TEMPERATURE RANGE (°C)
PCB83(0)C552-5-16	4.0	6.0	1.2	16	0 to +70
PCF83(0)C552-5-16	4.0	6.0	1.2	16	–40 to +85
PCA83(0)C552-5-16	4.5	5.5	1.2	16	–40 to +125
PCB83(0)C552-5-24	4.5	5.5	1.2	24	0 to +70
PCF83(0)C552-5-24	4.5	5.5	1.2	24	–40 to +85
PCB83(0)C552-5-30	4.5	5.5	1.2	30	0 to +70

DC ELECTRICAL CHARACTERISTICS

V_{SS}, AV_{SS} = 0V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
I_{DD}	Supply current operating:	See notes 1 and 2			
	PCB8XC552-5-16	f_{OSC} = 16MHz		45	mA
	PCF8XC552-5-16	f_{OSC} = 16MHz		45	mA
	PCA8XC552-5-16	f_{OSC} = 16MHz		40	mA
	PCB8XC552-5-24	f_{OSC} = 24MHz		55	mA
	PCF8XC552-5-24	f_{OSC} = 24MHz		55	mA
	PCB8XC552-5-30	f_{OSC} = 30MHz		68	mA
I_{ID}	Idle mode:	See notes 1 and 3			
	PCB8XC552-5-16	f_{OSC} = 16MHz		10	mA
	PCF8XC552-5-16	f_{OSC} = 16MHz		10	mA
	PCA8XC552-5-16	f_{OSC} = 16MHz		9	mA
	PCB8XC552-5-24	f_{OSC} = 24MHz		12.5	mA
	PCF8XC552-5-24	f_{OSC} = 24MHz		12.5	mA
	PCB8XC552-5-30	f_{OSC} = 30MHz		15	mA
I_{PD}	Power-down current:	See notes 1 and 4; $2V < V_{PD} < V_{DD}$ max			
	PCB8XC552			50	µA
	PCF8XC552			50	µA
	PCA8XC552			150	µA
Inputs					
V_{IL}	Input low voltage, except EA, P1.6, P1.7		-0.6	$0.2V_{DD}-0.1$	V
V_{IL1}	Input low voltage to EA		-0.5	$0.2V_{DD}-0.3$	V
V_{IL2}	Input low voltage to P1.6/SCL, P1.7/SDA[5]		-0.5	$0.3V_{DD}$	V
V_{IH}	Input high voltage, except XTAL1, RST, P1.6/SCL, P1.7/SDA		$0.2V_{DD}+0.9$	$V_{DD}+0.5$	V
V_{IH1}	Input high voltage, XTAL1, RST		$0.7V_{DD}$	$V_{DD}+0.5$	V
V_{IH2}	Input high voltage, P1.6/SCL, P1.7/SDA[5]		$0.7V_{DD}$	6.0	V
I_{IL}	Logical 0 input current, ports 1, 2, 3, 4, except P1.6, P1.7	V_{IN} = 0.45V		-50	µA
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3, 4, except P1.6, P1.7	See note 6		-650	µA
$\pm I_{IL1}$	Input leakage current, port 0, EA, STADC, EW	$0.45V < V_I < V_{DD}$		10	µA
$\pm I_{IL2}$	Input leakage current, P1.6/SCL, P1.7/SDA	$0V < V_I < 6V$ $0V < V_{DD} < 5.5V$		10	µA
$\pm I_{IL3}$	Input leakage current, port 5	$0.45V < V_I < V_{DD}$		1	µA
Outputs					
V_{OL}	Output low voltage, ports 1, 2, 3, 4, except P1.6, P1.7	I_{OL} = 1.6mA[7]		0.45	V
V_{OL1}	Output low voltage, port 0, ALE, PSEN, PWM0, PWM1	I_{OL} = 3.2mA[7]		0.45	V
V_{OL2}	Output low voltage, P1.6/SCL, P1.7/SDA	I_{OL} = 3.0mA[7]		0.4	V
V_{OH}	Output high voltage, ports 1, 2, 3, 4, except P1.6/SCL, P1.7/SDA	V_{DD} = 5V ±10% $-I_{OH}$ = 50µA $-I_{OH}$ = 25µA $-I_{OH}$ = 10µA	 2.4 $0.75V_{DD}$ $0.9V_{DD}$		 V V V

DC ELECTRICAL CHARACTERISTICS (Continued)

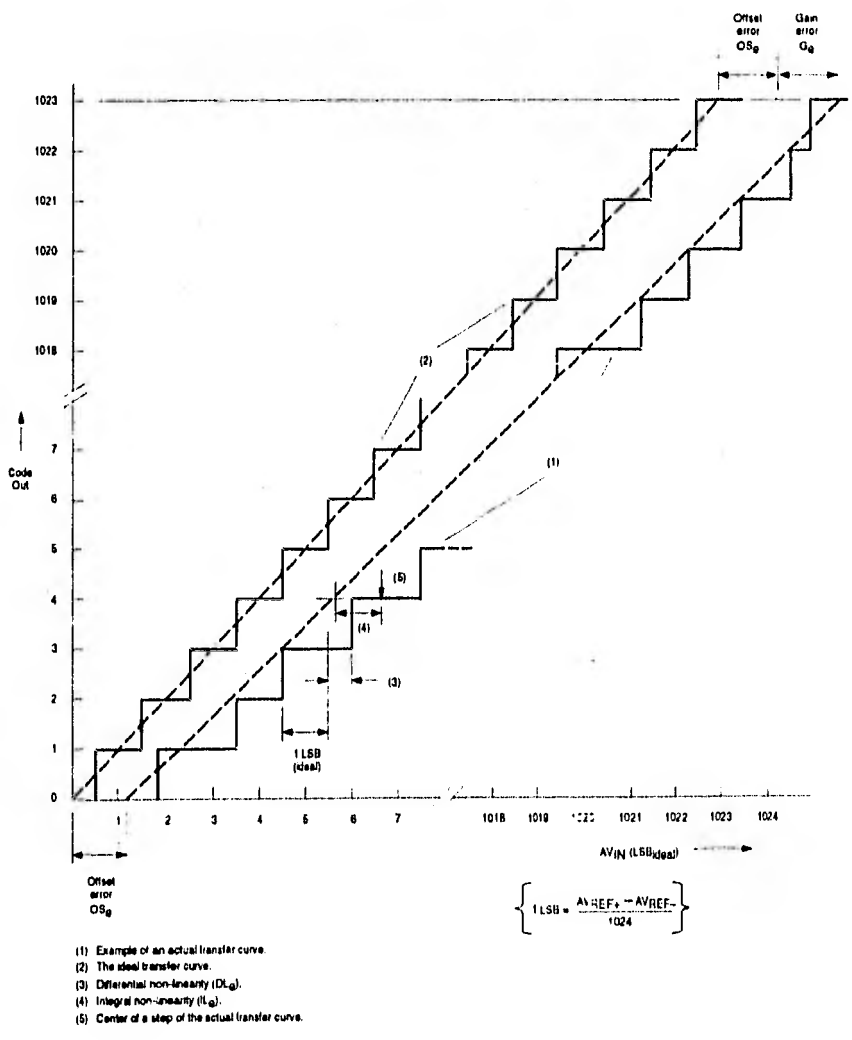
SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
Outputs (Continued)					
V_{OH1}	Output high voltage (port 0 in external bus mode, ALE, $\overline{PSEN}$, PWM0, $\overline{PWM1}$)[8]	V_{DD} = 5V ±10% $-I_{OH}$ = 400µA $-I_{OH}$ = 150µA $-I_{OH}$ = 40µA	 2.4 0.75V_{DD} 0.9V_{DD}		 V V V
V_{OH2}	Output high voltage (RST)	$-I_{OH}$ = 400µA $-I_{OH}$ = 120µA	2.4 0.8V_{DD}		V V
R_{RST}	Internal reset pull-down resistor		50	150	kΩ
C_{IO}	Pin capacitance	Test freq = 1MHz, T_{amb} = 25°C		10	pF
Analog Inputs					
AV_{DD}	Analog supply voltage: PCB8XC552-5-16 PCF8XC552-5-18 PCA8XC552-5-16 PCB8XC552-5-24 PCF8XC552-5-24 PCB8XC552-5-30	 AV_{DD} = V_{DD}±0.2V AV_{DD} = V_{DD}±0.2V AV_{DD} = V_{DD}±0.2V AV_{DD} = V_{DD}±0.2V AV_{DD} = V_{DD}±0.2V AV_{DD} = V_{DD}±0.2V	 4.0 4.0 4.5 4.5 4.5 4.5	 6.0 6.0 5.5 5.5 5.5 5.5	 V V V V V V
AI_{DD}	Analog supply current: operating: (16MHz) Analog supply current: operating: (24MHz, 30MHz)	Port 5 = 0 to AV_{DD} Port 5 = 0 to AV_{DD}		1.2 1.0	mA mA
AI_{ID}	Idle mode: PCB8XC552-5-18 PCF8XC552-5-16 PCA8XC552-5-16 PCB8XC552-5-24 PCF8XC552-5-24 PCB8XC552-5-30			 50 50 100 50 50 50	 µA µA µA µA µA µA
AI_{PD}	Power-down mode: PCB8XC552 PCF8XC552 PCA8XC552	2V < AV_{PC} < AV_{DD} max		 50 50 100	 µA µA µA
AV_{IN}	Analog input voltage		AV_{SS}–0.2	AV_{DD}+0.2	V
AV_{REF}	Reference voltage: AV_{REF-} AV_{REF+}		 AV_{SS}–0.2	 AV_{DD}+0.2	 V V
R_{REF}	Resistance between AV_{REF+} and AV_{REF-}		10	50	kΩ
C_{IA}	Analog input capacitance			15	pF
t_{ADS}	Sampling time			8t_{CY}	µs
t_{ADC}	Conversion time (including sampling time)			50t_{CY}	µs
DL_e	Differential non-linearity[10, 11, 12]			±1	LSB
IL_e	Integral non-linearity[10, 13]			±2	LSB
OS_e	Offset error[10, 14]			±2	LSB

DC ELECTRICAL CHARACTERISTICS (Continued)

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
Analog Inputs (continued)					
G_e	Gain error[10, 15]			±0.4	%
A_e	Absolute voltage error[10, 16]			±3	LSB
M_{CTC}	Channel to channel matching			±1	LSB
C_I	Crosstalk between inputs of port 5[17]	0–100kHz		–60	dB

NOTES FOR DC ELECTRICAL CHARACTERISTICS:

1. See Figures 10 through 15 for I_{DD} test conditions.
2. The operating supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{DD} - 0.5V$; XTAL2 not connected; $\overline{EA}$ = RST = Port 0 = $\overline{EW}$ = V_{DD}; STADC = V_{SS}.
3. The idle mode supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{DD} - 0.5V$; XTAL2 not connected; Port 0 = $\overline{EW}$ = V_{DD}; $\overline{EA}$ = RST = STADC = V_{SS}.
4. The power-down current is measured with all output pins disconnected; XTAL2 not connected; Port 0 = $\overline{EW}$ = V_{DD}; $\overline{EA}$ = RST = STADC = XTAL1 = V_{SS}.
5. The input threshold voltage of P1.6 and P1.7 (SIO1) meets the I^2C specification, so an input voltage below 1.5V will be recognized as a logic 0 while an input voltage above 3.0V will be recognized as a logic 1.
6. Pins of ports 1 (except P1.6, P1.7), 2, 3, and 4 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.
7. Capacitive loading on ports 0 and 2 may cause spurious noise to be superimposed on the V_{OL}s of ALE and ports 1 and 3. The noise is due to external bus capacitance discharging into the port 0 and port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases, it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input. I_{OL} can exceed these conditions provided that no single output sinks more than 5mA and no more than two outputs exceed the test conditions.
8. Capacitive loading on ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the $0.9V_{DD}$ specification when the address bits are stabilizing.
9. The following condition must not be exceeded: $V_{DD} - 0.2V < AV_{DD} < V_{DD} + 0.2V$.
10. Conditions: $AV_{REF-} = 0V$; $AV_{DD} = 5.0V$, AV_{REF+} (80C552, 83C552) = 5.12V. ADC is monotonic with no missing codes. Measurement by continuous conversion of $AV_{IN} = -20mV$ to 5.12V in steps of 0.5mV.
11. The differential non-linearity (DL_e) is the difference between the actual step width and the ideal step width. (See Figure 1.)
12. The ADC is monotonic; there are no missing codes.
13. The integral non-linearity (IL_e) is the peak difference between the center of the steps of the actual and the ideal transfer curve after appropriate adjustment of gain and offset error. (See Figure 1.)
14. The offset error (OS_e) is the absolute difference between the straight line which fits the actual transfer curve (after removing gain error), and a straight line which fits the ideal transfer curve. (See Figure 1.)
15. The gain error (G_e) is the relative difference in percent between the straight line fitting the actual transfer curve (after removing offset error), and the straight line which fits the ideal transfer curve. Gain error is constant at every point on the transfer curve. (See Figure 1.)
16. The absolute voltage error (A_e) is the maximum difference between the center of the steps of the actual transfer curve of the non-calibrated ADC and the ideal transfer curve.
17. This should be considered when both analog and digital signals are simultaneously input to port 5.



(1) Example of an actual transfer curve.
(2) The ideal transfer curve.
(3) Differential non-linearity (DL_E).
(4) Integral non-linearity (IL_E).
(5) Center of a step of the actual transfer curve.

Figure 1. ADC Conversion Characteristic

AC ELECTRICAL CHARACTERISTICS[1, 2]

16 MHz version

SYMBOL	FIGURE	PARAMETER	16MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$	2	Oscillator frequency			1.2	16	MHz
t_{LHLL}	2	ALE pulse width	85		$2t_{CLCL}-40$		ns
t_{AVLL}	2	Address valid to ALE low	8		$t_{CLCL}-55$		ns
t_{LLAX}	2	Address hold after ALE low	28		$t_{CLCL}-35$		ns
t_{LLIV}	2	ALE low to valid instruction in		150		$4t_{CLCL}-100$	ns
t_{LLPL}	2	ALE low to PSEN low	23		$t_{CLCL}-40$		ns
t_{PLPH}	2	PSEN pulse width	143		$3t_{CLCL}-45$		ns
t_{PLIV}	2	PSEN low to valid instruction in		83		$3t_{CLCL}-105$	ns
t_{PXIX}	2	Input instruction hold after PSEN	0		0		ns
t_{PXIZ}	2	Input instruction float after PSEN		38		$t_{CLCL}-25$	ns
t_{AVIV}	2	Address to valid instruction in		208		$5t_{CLCL}-105$	ns
t_{PLAZ}	2	PSEN low to address float		10		10	ns
Data Memory							
t_{RLRH}	3	RD pulse width	275		$6t_{CLCL}-100$		ns
t_{WLWH}	4	WR pulse width	275		$6t_{CLCL}-100$		ns
t_{RLDV}	3	RD low to valid data in		148		$5t_{CLCL}-165$	ns
t_{RHDX}	3	Data hold after RD	0		0		ns
t_{RHDZ}	3	Data float after RD		55		$2t_{CLCL}-70$	ns
t_{LLDV}	3	ALE low to valid data in		350		$8t_{CLCL}-150$	ns
t_{AVDV}	3	Address to valid data in		398		$9t_{CLCL}-165$	ns
t_{LLWL}	3, 4	ALE low to RD or WR low	138	238	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	3, 4	Address valid to WR low or RD low	120		$4t_{CLCL}-130$		ns
t_{QVWX}	4	Data valid to WR transition	3		$t_{CLCL}-60$		ns
t_{QW}	4	Data before WR	288		$7t_{CLCL}-150$		ns
t_{WHQX}	4	Data hold after WR	13		$t_{CLCL}-50$		ns
t_{RLAZ}	3	RD low to address float		0		0	ns
t_{WHLH}	3, 4	RD or WR high to ALE high	23	103	$t_{CLCL}-40$	$t_{CLCL}+40$	ns
External Clock							
t_{CHCX}	5	High time[4]	20		20		ns
t_{CLCX}	5	Low time[4]	20		20		ns
t_{CLCH}	5	Rise time[4]		20		20	ns
t_{CHCL}	5	Fall time[4]		20		20	ns
Serial Timing – Shift Register Mode[4] (Test Conditions: T_{amb} = 0°C to +70°C; V_{SS} = 0V; Load Capacitance = 80pF)							
t_{XLXL}	6	Serial port clock cycle time	0.75		$12t_{CLCL}$		µs
t_{QVXH}	6	Output data setup to clock rising edge	492		$10t_{CLCL}-133$		ns
t_{XHQX}	6	Output data hold after clock rising edge	8		$2t_{CLCL}-117$		ns
t_{XHDX}	6	Input data hold after clock rising edge	0		0		ns
t_{XHDV}	6	Clock rising edge to input data valid		492		$10t_{CLCL}-133$	ns

NOTES:

1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and PSEN = 100pF; load capacitance for all other outputs = 80pF.
3. $t_{CLCL} = 1/f_{OSC}$ = one oscillator clock period.
   t_{CLCL} = 83.3ns at f_{OSC} = 12MHz.
   t_{CLCL} = 62.5ns at f_{OSC} = 16MHz.
4. These values are characterized but not 100% production tested.

AC ELECTRICAL CHARACTERISTICS (Continued)[1,2]

24/30 MHz version

SYMBOL	FIGURE	PARAMETER	24MHz CLOCK		30MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	MIN	MAX	
$1/t_{CLCL}$	2	Oscillator frequency					1.2	24	MHz
t_{LHLL}	2	ALE pulse width	43		27		$2t_{CLCL}-40$		ns
t_{AVLL}	2	Address valid to ALE low	17		8		$t_{CLCL}-25$		ns
t_{LLAX}	2	Address hold after ALE low	17		8		$t_{CLCL}-25$		ns
t_{LLIV}	2	ALE low to valid instruction in		102		68		$4t_{CLCL}-65$	ns
t_{LLPL}	2	ALE low to $\overline{PSEN}$ low	17		8		$t_{CLCL}-25$		ns
t_{PLPH}	2	$\overline{PSEN}$ pulse width	80		55		$3t_{CLCL}-45$		ns
t_{PLIV}	2	$\overline{PSEN}$ low to valid instruction in		65		40		$3t_{CLCL}-60$	ns
t_{PXIX}	2	Input instruction hold after $\overline{PSEN}$	0		0		0		ns
t_{PXIZ}	2	Input instruction float after $\overline{PSEN}$		17		8		$t_{CLCL}-25$	ns
t_{AVIV}	2	Address to valid instruction in		128		87		$5t_{CLCL}-80$	ns
t_{PLAZ}	2	$\overline{PSEN}$ low to address float		10		10		10	ns
Data Memory									
t_{RLRH}	3	$\overline{RD}$ pulse width	150		100		$6t_{CLCL}-100$		ns
t_{WLWH}	4	$\overline{WR}$ pulse width	150		100		$6t_{CLCL}-100$		ns
t_{RLDV}	3	$\overline{RD}$ low to valid data in		118		77		$5t_{CLCL}-90$	ns
t_{RHDX}	3	Data hold after $\overline{RD}$	0		0		0		ns
t_{RHDZ}	3	Data float after $\overline{RD}$		55		39		$2t_{CLCL}-28$	ns
t_{LLDV}	3	ALE low to valid data in		183		117		$8t_{CLCL}-150$	ns
t_{AVDV}	3	Address to valid data in		210		135		$9t_{CLCL}-165$	ns
t_{LLWL}	3, 4	ALE low to $\overline{RD}$ or $\overline{WR}$ low	75	175	50	150	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	3, 4	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	92		58		$4t_{CLCL}-75$		ns
t_{QVWX}	4	Data valid to $\overline{WR}$ transition	12		3		$t_{CLCL}-30$		ns
t_{DW}	4	Data before $\overline{WR}$	162		103		$7t_{CLCL}-130$		ns
t_{WHQX}	4	Data hold after $\overline{WR}$	17		8		$t_{CLCL}-25$		ns
t_{RLAZ}	3	$\overline{RD}$ low to address float		0		0		0	ns
t_{WHLH}	3, 4	$\overline{RD}$ or $\overline{WR}$ high to ALE high	17	67	8	58	$t_{CLCL}-25$	$t_{CLCL}+25$	ns
External Clock									
t_{CHCX}	5	High time[3]	17		15		17		ns
t_{CLCX}	5	Low time[3]	17		15		17		ns
t_{CLCH}	5	Rise time[3]		5		3		20	ns
t_{CHCL}	5	Fall time[3]		5		3		20	ns
Serial Timing – Shift Register Mode[3] (Test Conditions: T_{amb} = 0°C to +70°C; V_{SS} = 0V; Load Capacitance = 80pF)									
t_{XLXL}	8	Serial port clock cycle time	0.5		0.4		$12t_{CLCL}$		µs
t_{QVXH}	6	Output data setup to clock rising edge	283		200		$10t_{CLCL}-133$		ns
t_{XHQX}	6	Output data hold after clock rising edge	23		6.6		$2t_{CLCL}-60$		ns
t_{XHDX}	8	Input data hold after clock rising edge	0		0		0		ns
t_{XHDV}	5	Clock rising edge to input data valid		283		200		$10t_{CLCL}-133$	ns

NOTES:

1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF
3. These values are characterized but not 100% production tested.
4. $t_{CLCL} = 1/f_{OSC}$ = one oscillator clock period.
   t_{CLCL} = 41.7ns at f_{OSC} = 24MHz.

AC ELECTRICAL CHARACTERISTICS (Continued)

SYMBOL	PARAMETER	INPUT	OUTPUT
I²C Interface (Refer to Figure 9)			
t_{HD}; STA	START condition hold time	≥ 14 t_{CLCL}	> 4.0µs [1]
t_{LOW}	SCL low time	≥ 16 t_{CLCL}	> 4.7µs [1]
t_{HIGH}	SCL high time	≥ 14 t_{CLCL}	> 4.0µs [1]
t_{RC}	SCL rise time	≤ 1µs	– [2]
t_{FC}	SCL fall time	≤ 0.3µs	< 0.3µs [3]
t_{SU}; DAT1	Data set-up time	≥ 250ns	> 20 t_{CLCL} – t_{RD}
t_{SU}; DAT2	SDA set-up time (before rep. START cond.)	≥ 250ns	> 1µs [1]
t_{SU}; DAT3	SDA set-up tims (before STOP cond.)	≥ 250ns	> 8 t_{CLCL}
t_{HD}; DAT	Data hold time	≥ 0ns	> 8 t_{CLCL} – t_{FC}
t_{SU}; STA	Repested START set-up time	≥ 14 t_{CLCL}	> 4.7µs [1]
t_{SU}; STO	STOP condition set-up time	≥ 14 t_{CLCL}	> 4.0µs [1]
t_{BUF}	Bus free time	≥ 14 t_{CLCL}	> 4.7µs [1]
t_{RD}	SDA rise time	≤ 1µs	– [2]
t_{FD}	SDA fall time	≤ 0.3µs	< 0.3µs [3]

NOTES:
1. At 100 kbit/s. At other bit rates this value is inversely proportional to the bit-rate of 100 kbit/s.
2. Determined by the external bus-line capacitance and the external bus-line pull-resistor, this must be < 1µs.
3. Spikes on the SDA and SCL lines with a duration of less than 3 t_{CLCL} will be filtered out. Maximum capacitance on bus-lines SDA and SCL = 400pF.
4. t_{CLCL} = 1/f_{OSC} = one oscillator clock period at pin XTAL1. For 62ns, 42ns, 33.3ns < t_{CLCL} < 285ns (16MHz, 24MHz, 30MHz > f_{OSC} > 1.2MHz) the SIO1 interface meets the I²C-bus specification for bit-rates up to 100 kbit/s.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – PSEN
Q – Output data
R – RD signal
t – Time
V – Valid
W – WR signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to PSEN low.

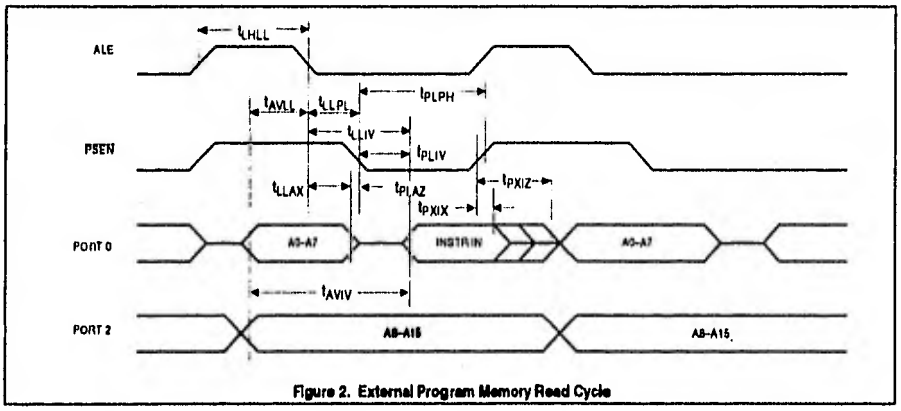


Figure 2. External Program Memory Read Cycle

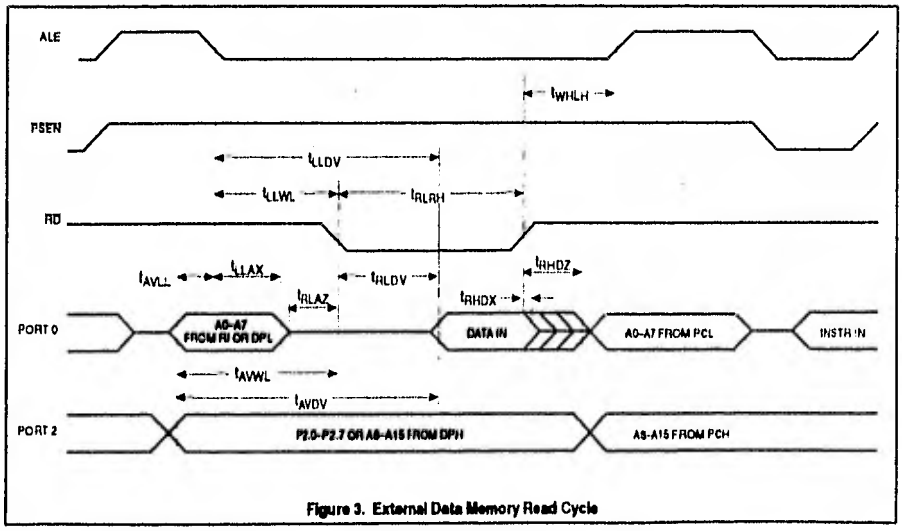


Figure 3. External Data Memory Read Cycle

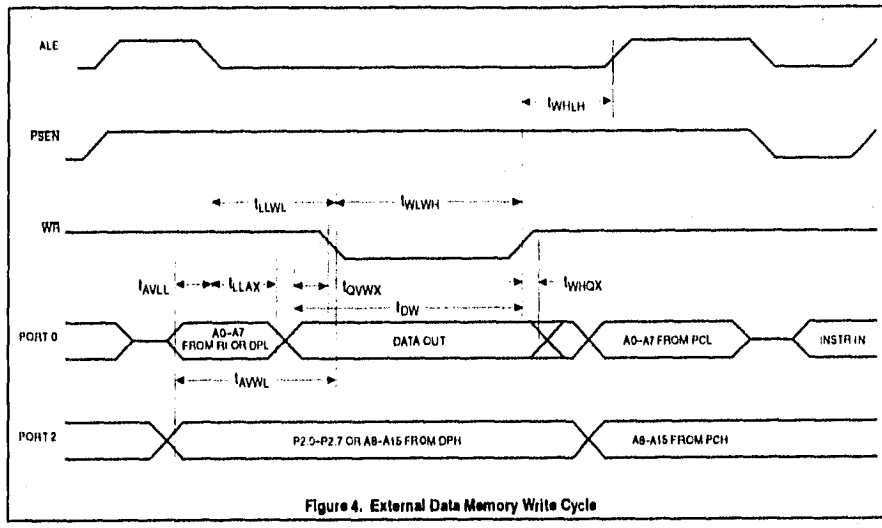


Figure 4. External Data Memory Write Cycle

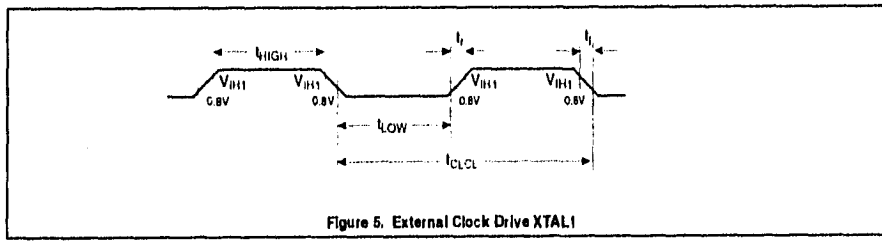


Figure 5. External Clock Drive XTAL1

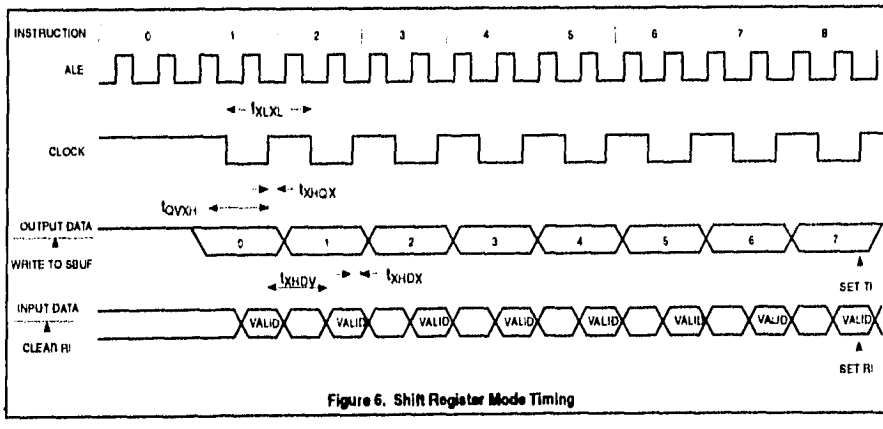
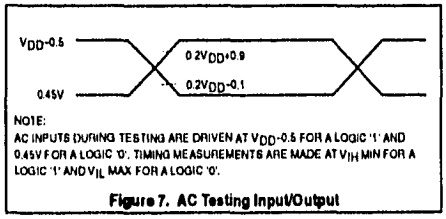
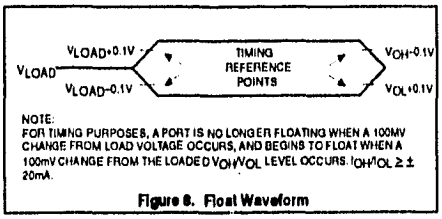


Figure 6. Shift Register Mode Timing



NOTE:
AC INPUTS DURING TESTING ARE DRIVEN AT V_{DD}-0.5 FOR A LOGIC '1' AND 0.45V FOR A LOGIC '0'. TIMING MEASUREMENTS ARE MADE AT V_{IH} MIN FOR A LOGIC '1' AND V_{IL} MAX FOR A LOGIC '0'.

Figure 7. AC Testing Input/Output



NOTE:
FOR TIMING PURPOSES, A PORT IS NO LONGER FLOATING WHEN A 100MV CHANGE FROM LOAD VOLTAGE OCCURS, AND BEGINS TO FLOAT WHEN A 100mV CHANGE FROM THE LOADED V_{OH}/V_{OL} LEVEL OCCURS. I_{OH}/I_{OL} ≥ ± 20mA.

Figure 8. Float Waveform

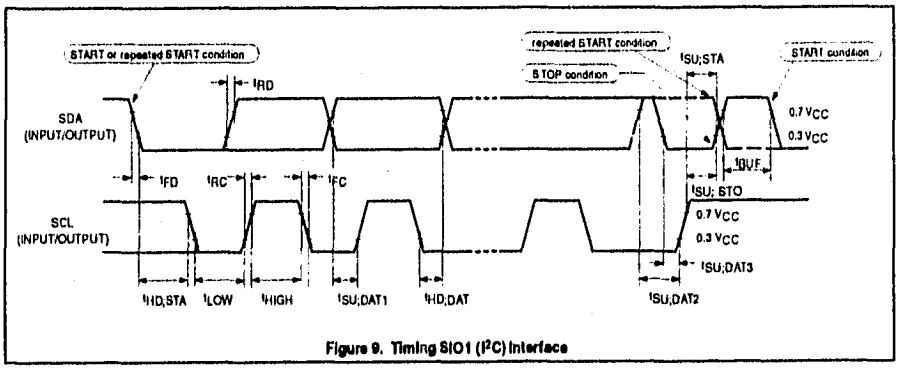
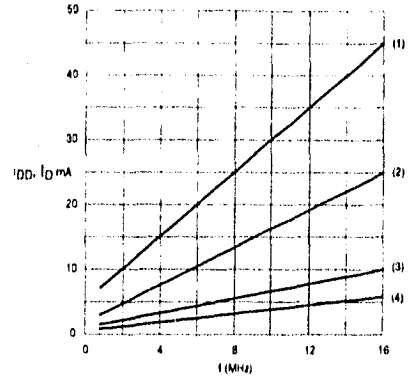


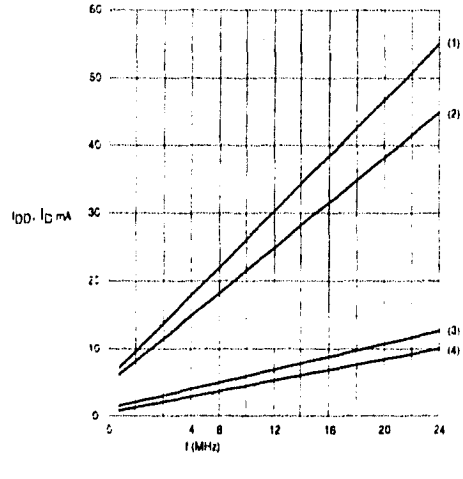
Figure 9. Timing SIO1 (I^2C) Interface



(1) Maximum operating mode; V_{DD} = 6V
(2) Maximum operating mode; V_{DD} = 4V
(3) Maximum idle mode; V_{DD} = 6V
(4) Maximum idle mode; V_{DD} = 4V

NOTE:
These values are valid only within the frequency specifications of the device under test.

Figure 10. 16MHz Version Supply Current (I_{DD}) as a Function of Frequency at XTAL1 (f_{OSC})



(1) Maximum operating mode; V_{DD} = 5.5V
(2) Maximum operating mode; V_{DD} = 4.5V
(3) Maximum idle mode, V_{DD} = 5.5V
(4) Maximum idle mode; V_{DD} = 4.5V

NOTE:
These values are valid only within the frequency specifications of the device under test.

Figure 11. 24MHz Version Supply Current (I_{DD}) as a Function of Frequency at XTAL1 (f_{OSC})

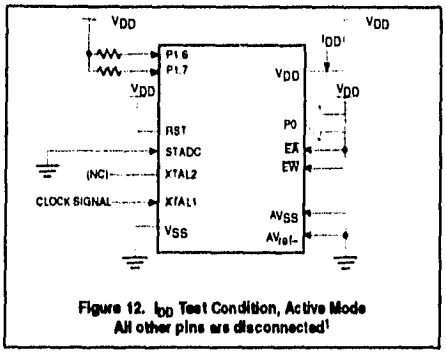


Figure 12. I_{DD} Test Condition, Active Mode
All other pins are disconnected[1]

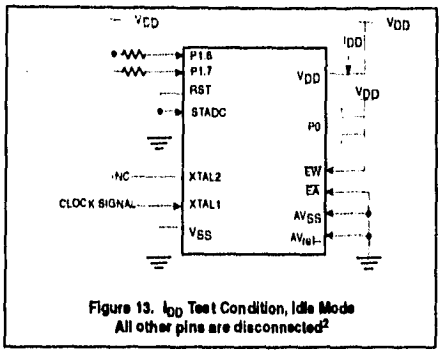


Figure 13. I_{DD} Test Condition, Idle Mode
All other pins are disconnected[2]

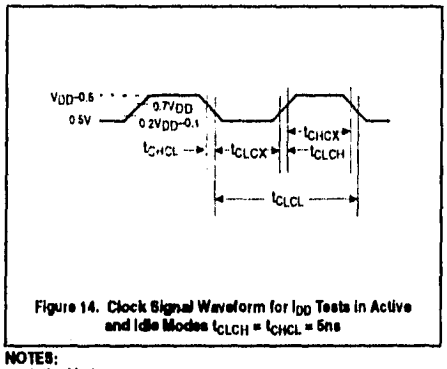


Figure 14. Clock Signal Waveform for I_{DD} Tests in Active and Idle Modes $t_{CLCH} = t_{CHCL} = 5ns$

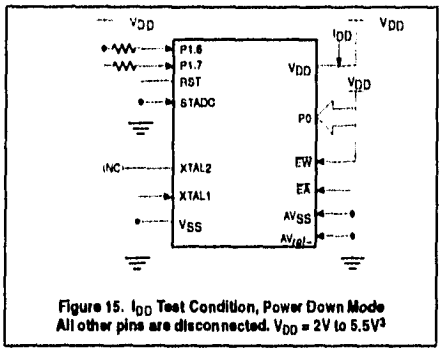


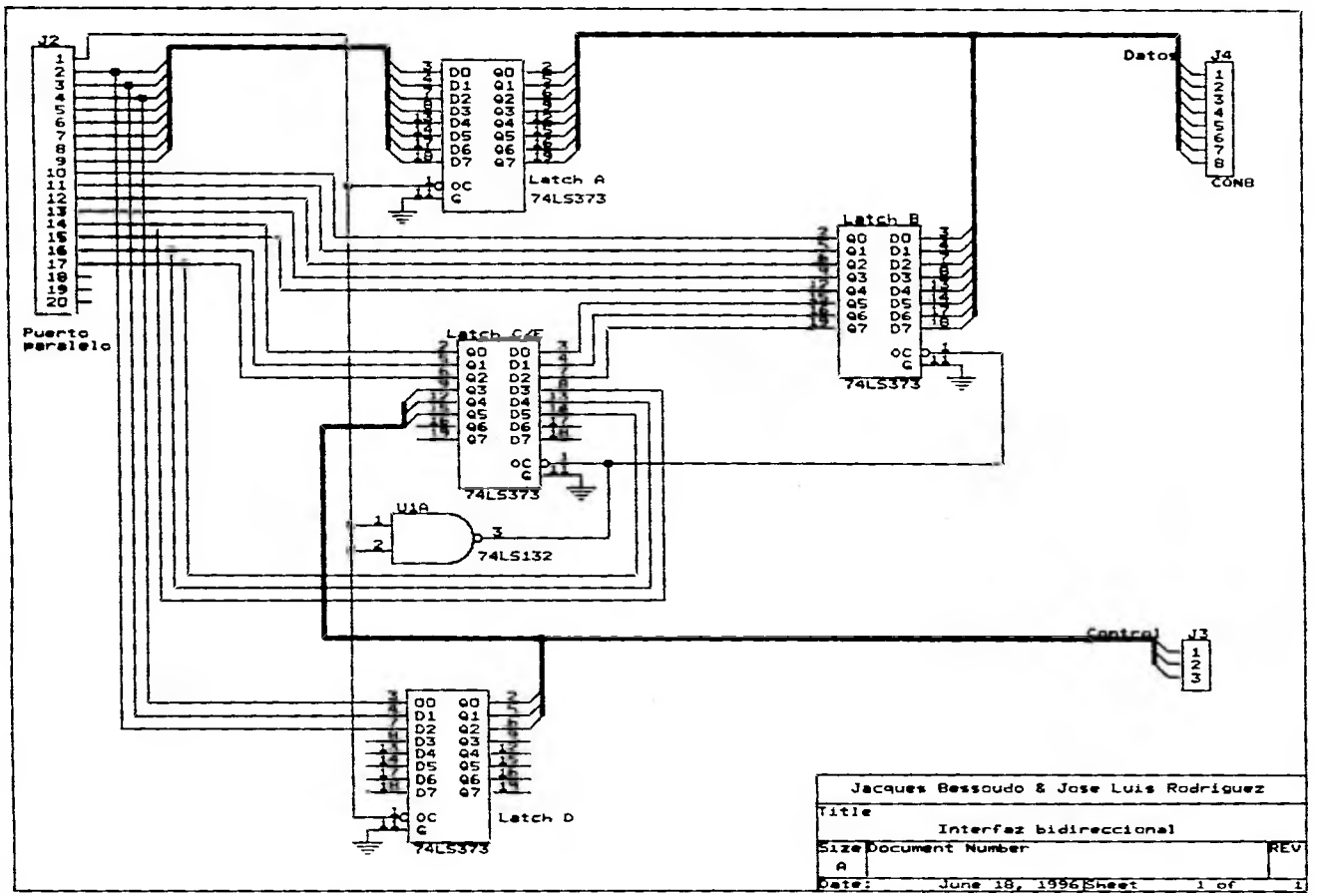
Figure 15. I_{DD} Test Condition, Power Down Mode
All other pins are disconnected. V_{DD} = 2V to 5.5V[3]

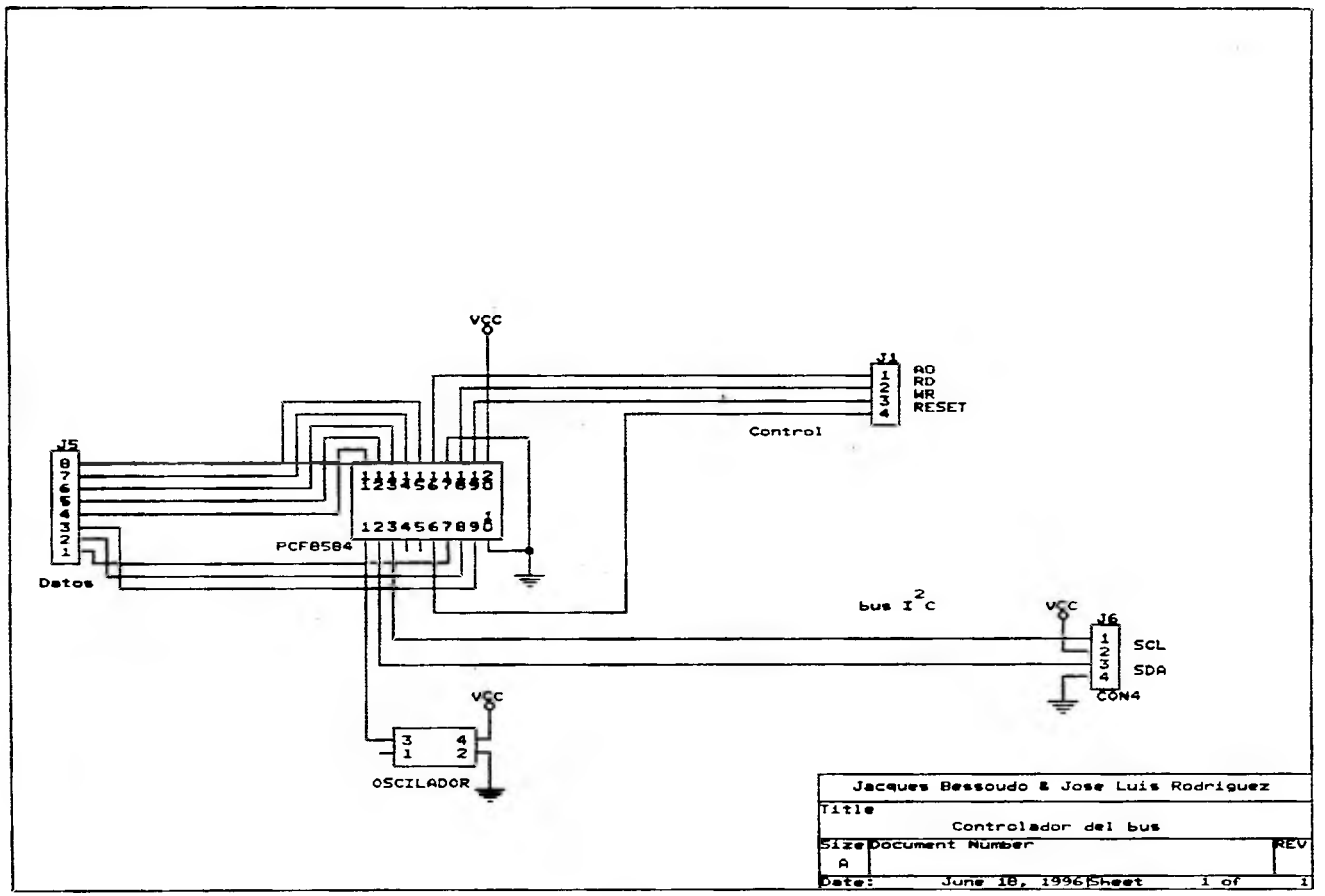
NOTES:

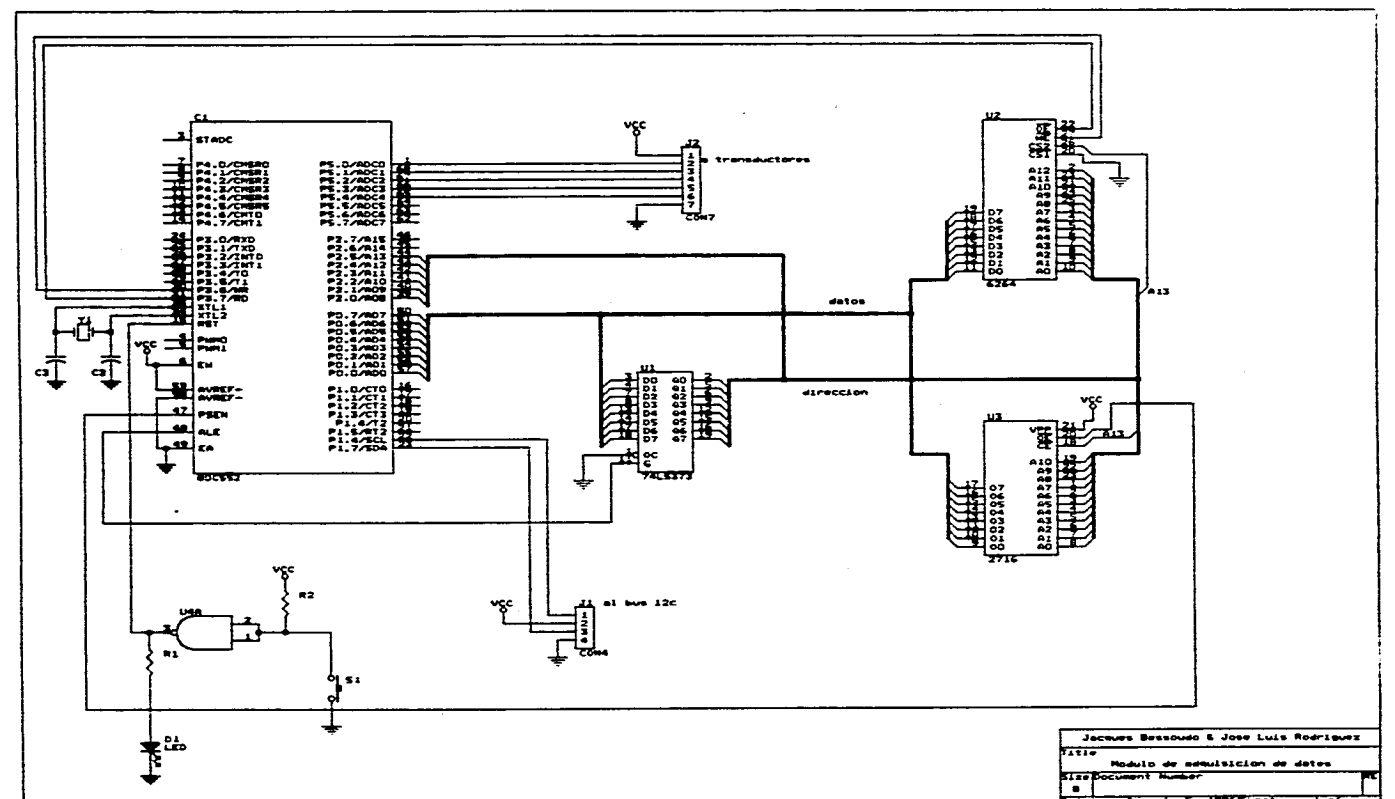
1. Active Mode:
   a. The following pins must be forced to V_{DD}: $\overline{EA}$, RST, Port 0, and EW.
   b. The following pins must be forced to V_{SS}: STADC, AV_{SS}, and AV_{ref-}.
   c. Ports 1.6 and 1.7 should be connected to V_{DD} through resistors of sufficiently high value such that the sink current into these pins cannot exceed the I_{OL1} spec of these pins.
   d. The following pins must be disconnected: XTAL2 and all pins not specified above.
2. Idle Mode:
   a. The following pins must be forced to V_{DD}: Port 0 and EW.
   b. The following pins must be forced to V_{SS}: RST, STADC, AV_{SS}, AV_{ref-}, and $\overline{EA}$.
   c. Ports 1.6 and 1.7 should be connected to V_{DD} through resistors of sufficiently high value such that the sink current into these pins cannot exceed the I_{OL1} spec of these pins. These pins must not have logic 0 written to them prior to this measurement.
   d. The following pins must be disconnected: XTAL2 and all pins not specified above.
3. Power Down Mode:
   a. The following pins must be forced to V_{DD}: Port 0 and EW.
   b. The following pins must be forced to V_{SS}: RST, STADC, XTAL1, AV_{SS}, AV_{ref-}, and $\overline{EA}$.
   c. Ports 1.6 and 1.7 should be connected to V_{DD} through resistors of sufficiently high value such that the sink current into these pins cannot exceed the I_{OL} spec of these pins. These pins must not have logic 0 written to them prior to this measurement.
   d. The following pins must be disconnected: XTAL2 and all pins not specified above.

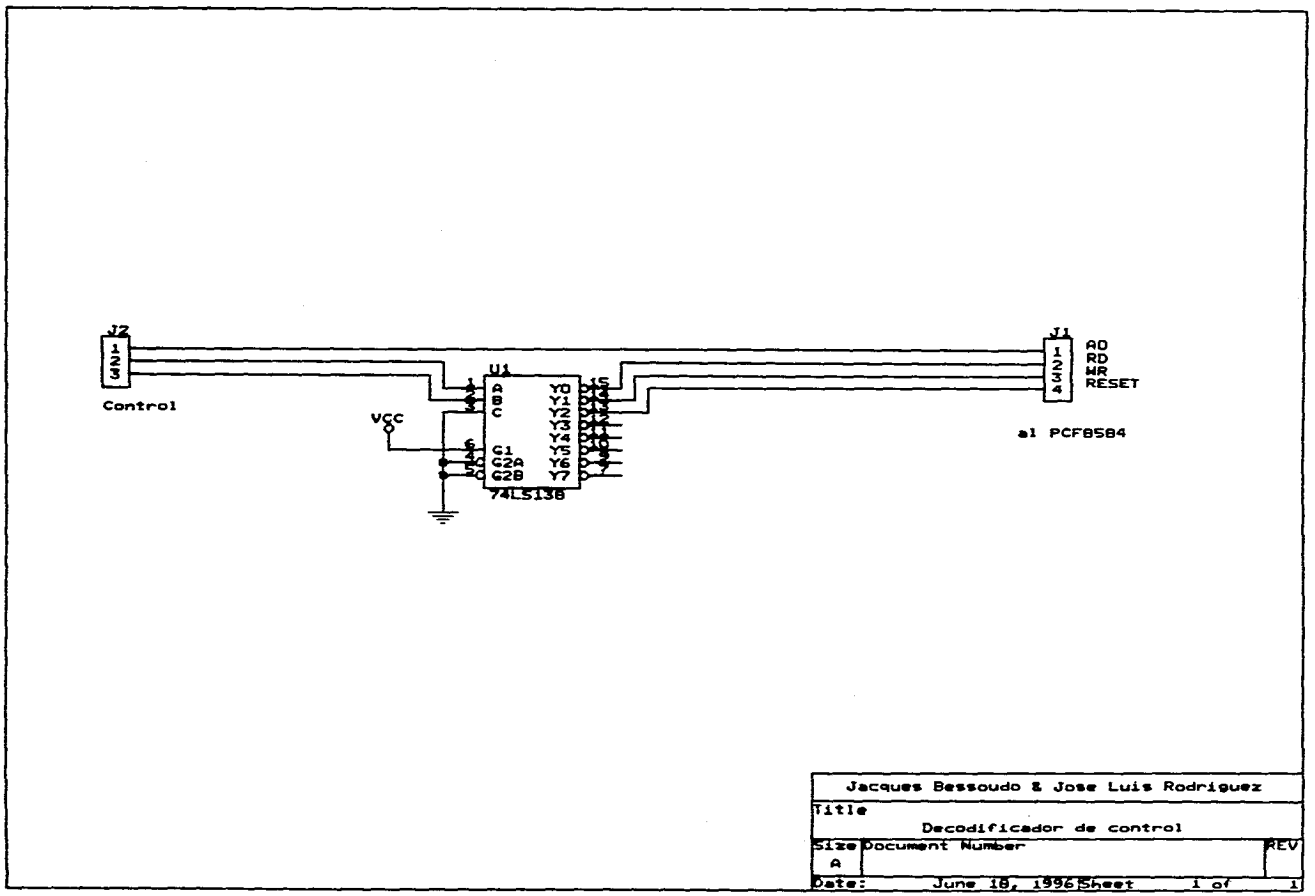
Purchase of Philips' I²C components conveys a license under the Philips' I²C patent to use the components in the I²C-system provided the system conforms to the I²C specifications defined by Philips.

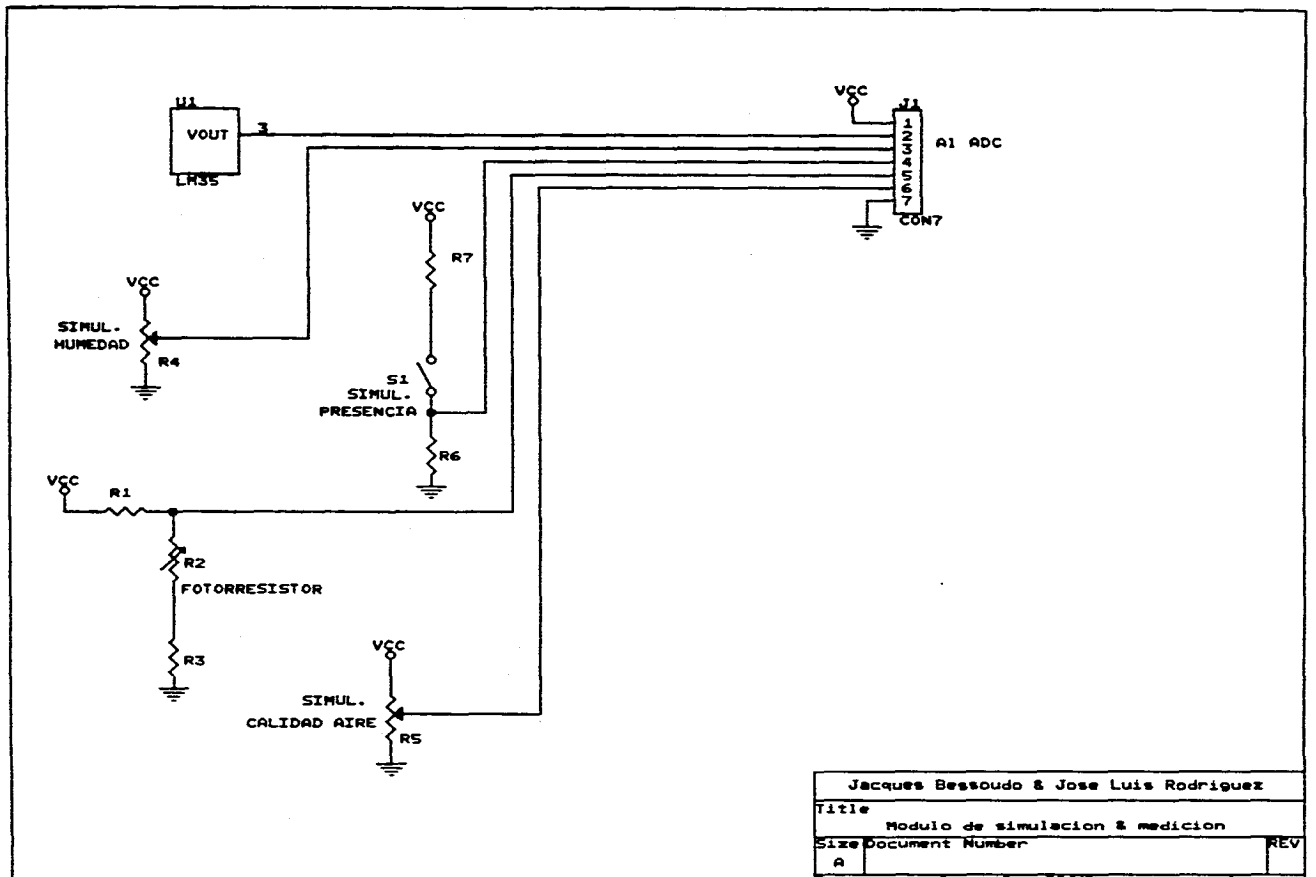
Apéndice B: Diagramas electrónicos


J2
Puerto paralelo
Latch A
74LS373
Latch B
74LS373
Latch C/E
74LS373
U1A
74LS132
Latch D
74LS373
Datos
J4
CONB
Control
J3
Jacques Bessoudo & Jose Luis Rodriguez
Title
Interfaz bidireccional
Size
A
Document Number
REV
Date:
June 18, 1996
Sheet
1 of
1


VCC
J1
AD
RD
WR
RESET
Control
J5
Datos
PCF8584
bus I2C
VCC
J6
SCL
SDA
CON4
VCC
OSCILADOR
Jacques Bessoudo & Jose Luis Rodriguez
Title
Controlador del bus
Size
A
Document Number
REV
Date:
June 18, 1996
Sheet
1 of
1


datos
direccion
8 transductores
al bus 12c
Jacques Bessoudo & Jose Luis Rodriguez
Modulo de adquisicion de datos


J2
1
2
3
Control
U1
A
B
C
G1
G2A
G2B
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
74LS138
VCC
J1
1
2
3
4
AD
RD
WR
RESET
al PCF8584
Jacques Bessoudo & Jose Luis Rodriguez
Title
Decodificador de control
Size
A
Document Number
REV
Date:
June 18, 1996
Sheet
1 of
1


U1
VOUT
3
LM35
VCC
J1
1
2
3
4
5
6
7
A1 ADC
CON7
VCC
R7
VCC
SIMUL.
HUMEDAD
R4
S1
SIMUL.
PRESENCIA
R6
VCC
R1
R2
FOTORRESISTOR
R3
VCC
SIMUL.
CALIDAD AIRE
R5
Jacques Bessoudo & Jose Luis Rodriguez
Title
Modulo de simulacion & medicion
Size
A
Document Number
REV
Date:
June 19, 1996
Sheet
1 of
1

Apéndice C: Listado de programas

```
{$M $4000,0,0}
PROGRAM Control_pcf;
uses dos,crt;
var
 direcc,figtree:byte;
 modulo,puerto,k,x:integer;
FUNCTION ESCALA(X:BYTE):INTEGER;(*****************************************)
begin
 If(figtree=1)
 then Escala:= (x * 50 DIV 255) else
 if(figtree=3) then Escala:=x DIV 255 + 50 else Escala:=x * 100 DIV 255;
 end;
PROCEDURE INITPUERTO;(*************************************************)
begin
{writeln('Seleccione el puerto paralelo para I2C:  (1,2)   ');
readln (puerto);
If (puerto=1) then puerto:=$300 else} puerto:=$200;
end;
PROCEDURE COPY; (*********************************************************)
 begin
  SwapVectors;
  Exec (GetEnv('comspec'), '/c copy c:\monitmp.tmp c:\moni.tmp ');
  SwapVectors;
  end;
PROCEDURE STROBE(dir:integer);(*******************************************)
  var
   k:byte;
  begin
    port[puerto+$78]:=0;
    if(dir=1)
          (cambio a transmision )
    then k:=$B
    ELSE    {cambio a recepcion}
     k:=0;
    port[puerto+$7A]:= k;
   end;
PROCEDURE ALARCHIVO(dato:byte); (*****************************************)
(*LOS RENGLONES DEL ARCHIVO SON: MODULO PARAMETRO FECHA HORA LECTURA *)
 var
  a,b,c,d:Word;
  hora,fecha:real;      (* variables para conseguir la fecha y hora *)
  archivo,moni:Text;
  x:word;
  Info:SearchRec;
  i,param:integer;            (*variables para manejar archivos*)
begin
          (*figtree indica el n£mero de par metro que se envia*)
 param:=figtree;
 GetDate(a,b,c,d);
 fecha:=frac(int(a)/100)*1000000+int(b)*100+int(c);
 fecha:=Round(fecha);
 GetTime(a,b,c,d);
 hora:=10000*(int(c)/10000+int(b)/100+a);
                  (*meter los datos: modulo,figtree,fecha,hora,dato*)
```

```
 ASSIGN(archivo,'c:\tmp.tmp');
 (**archivo de monitoreo*)
   ASSIGN(moni,'c:\monitmp.tmp');
   FindFirst('c:\monitmp.tmp',x,info);
   If (DosError=0) and (figtree<5)
    thon APPEND(moni)  (*continuar escritura de archivo para monitoreo*)
    else (*Copiar archivo y reiniciar uno nuevo*)
      begin
        COPY;
        REWRITE(moni);
      end;
 FindFirst('c:\tmp.tmp',x,info);
 IF (DosError=0) then APPEND(archivo) else REWRITE(archivo);
  WRITE(archivo,modulo);write(archivo,' ');
  WRITE(moni,modulo);write(moni,' ');
  WRITE(archivo,param);write(archivo,' ');
  WRITE(moni,param);write(moni,' ');
  WRITE(archivo,fecha);write(archivo,' ');
  WRITE(moni,fecha);write(moni,' ');
  write(archivo,hora);write(archivo,' ');
  WRITE(moni,hora);write(moni,' ');
  DATO:=ESCALA(DATO);
  WRITE(archivo,dato);writeln(archivo,'');
  WRITE(moni,dato);writeln(moni,'');
 CLOSE(archivo);
 CLOSE (moni);
 If(param=5) then (*Copiar monitmp.tmp a monitor.tmp *) param:=5;
end;
PROCEDURE A0(x,dir,cambio:integer);(*************************************)
 var
  k:byte;
 begin
   if(dir=1) {transmision)
   then
     begin
       If(cambio=1) then STROBE(1);
       k:=port[puerto+$7A];
       If(x=1) then k:=k AND $F7 ELSE k:=k OR $8;
       port[puerto+$7A]:=k;
     end
   else {recepcion}
     begin
     If(cambio=1) then STROBE(0);
     k:=port[puerto+$78];
     If(x=1) then k:=k OR $4 ELSE k := k AND $FB;
     port[puerto+$78]:=k;
     end;
 end;
PROCEDURE RWRES(rd,wr,res,dir:integer);(*********************************)
 var k:byte;
 begin
  If (dir=1) {transmision) then
     begin
       k:=port[puerto+$7A];
```

```
      If (rd=1)
      then K:=k AND $F9
      else If (wr=1)
         then k:= k OR 8
         else if (res=1)
            then K:=K AND $FD OR 4
            else k:= k AND $FB OR 2   ;
      port[puerto+$7A]:=k;
     end
   else {recepci¢n}
    begin
     k:=port[puerto+$78];
     If (rd=1)
     then k:= k and $FD OR 1
     else If (wr=1)
        then k:= k AND $FE OR 2
        else if (res=1)
           then k:= k or 3
           else k:= k AND $FC ;
     port[puerto+$78]:=k;
    end;
 end;
PROCEDURE TXBYTE(dato:byte);(*********************************************)
(*Recibe A0(X,1,X) *)
 begin
  PORT[puerto+$78]:= dato;
  RWRES(0,1,0,1);
  DELAY(1);
  RWRES(0,0,0,1);
  delay(1);
  port[puerto+$78]:=0;
 end;
(*Termina con A0(X,1,X) *)
FUNCTION RXBYTE:byte;(*********************************************)
(*Recibe A0(X,0,X)  *)
 VAR
  RX1,RX2,RXT,RXtemp:byte;
 begin
    port[puerto+$7A]:= 4;  {Pone las salidas Open Collector en Alto}
    RWRES(1,0,0,0);
    delay(1);
    RX1:=port[puerto+$79];
    RX2:=port[puerto+$7A];
    RWRES(0,0,0,0);
    delay(1);
    RX1:=RX1 AND $F8;
    RX2:=RX2 DIV 2;
    RX2:=(RX2 AND $7);
    RXT:=RX1 OR RX2;
    RXtemp:=RXT AND $85;
    RXtemp:= NOT RXtemp;
    RXtemp:=RXtemp AND $85;
    RXT:=RXT AND $7A;
    RXT:=RXT OR RXtemp;
```

```
    RXBYTE:=RXT;
  end;
(*Termina con A0(X,0,X) *)
PROCEDURE initpcf;(*********************************************************)
 var k:byte;
 begin;
   strobe(1);
   port[puerto+$78]:=0;
   RWRES(0,0,1,1);
   delay(1);                    (*Asegura modo 80XX  *)
   RWRES(0,0,0,1);
   RWRES(0,1,0,1);
   DELAY(40);
   RWRES(0,0,0,1);
   A0(1,1,0);
   delay(1);
   TXBYTE($80);
   delay(40);
   A0(0,1,0);
   TXBYTE($55);                    (dir I2C=$AA)
                    (*Establecer frecuencia de operacion*)
   A0(1,1,0);
   TXBYTE($A0);
   A0(0,1,0);
   TXBYTE($1F);
                    (*Establecer vector de interrupciones *)
                    (*innecesario en esta aplicacion*)
   a0(1,1,1);
   txbyte($10);
   a0(0,1,0);
   txbyte(2);          (*Vector = 2  *)
                    (*Encender interfaz i2c  *)
   A0(1,1,0);
   TXBYTE($C1);
   a0(0,0,1);
   (*Termina con A0 en cero, lista para recibir*)
end;
PROCEDURE MRECEIVER(direcc,CUANTOS:integer);(**********************)
  var
   k,stat,chk,DATO: byte;
   begin;
                          (*"Bus libre?*)
                          (*Innecesario en este proyecto*)
     A0(1,0,0);
     REPEAT
     K:=RXBYTE;
     until (k AND 1=1);
     writeln('...Bus libre ... ');
                         (*Cargar direccion*)
    A0(0,1,1);
    direcc:=direcc*2 OR 1;
    TXBYTE(direcc);
                          (*Enviar direccion*)
    A0(1,1,0);
```

```
    TXBYTE($C5);
                              (*"Recibis byte?*)
    A0(1,0,1);
    repeat                  stat:=RXBYTE;
     delay(2000);             chk:=RXBYTE;
    until(stat=chk);

    A0(0,0,0);
    IF (stat AND 8 =0) (*Revisar que el esclavo haya enviado ACK=0*)
    THEN
      begin
      writeln('...Recibio ACK...');
      delay(100);
      A0(0,0,0);
      DATO:=RXBYTE;   (*dir i2c del esclavo *)
      repeat
        dato:=rxbyte;   (*Esperar sincronia*)
      until (dato=$99);
      For K :=1 to cuantos do begin
        for figtres:=1 to 6 DO
          begin
           dato:=rxbyte;
           delay(300);
           writeln(dato);
           If(dato=$99) then figtres:=8
            else IF (figtres<6) then alarchivo(dato);
           end;
       end;
      a0(1,1,1);
      TXBYTE($C0);
      A0(0,0,1);
      DATO:=RXBYTE;
      dato:=RXBYTE;
      dato:=rxbyte;
      end
    ELSE begin
      writeln('El modulo ',direcc AND 254 DIV 2,' no responde');
      delay(5000);
     end;
   end;
PROCEDURE MONITOR;(**************************************************)
 var lectu:byte;
 begin
  {**LECTURA DEL REGISTRO S1 - status **}
  a0(1,0,0);
   clrscr;
  lectu:=rxbyte;
  write('Estado del bus : ');writeln(lectu);
  writeln;
  {*LECTURA DEL REGISTRO S0' - dir i2c corrida **}
  a0(1,1,1);
  txbyte(0);
  a0(0,0,1);
  lectu:=rxbyte;
```

```
    if(lectu=0) then x:=0;
    write('Direccion propia : '); writeln (lectu);
    {** LECTURA DEL REGISTRO S3 - Interrupt vector **}
    a0(1,1,1);
    txbyte(16);
    a0(0,0,1);
    lectu:=rxbyte;
    write('Vector de interrupciones  :  ');writeln(lectu);
    {* LECTURA DEL REGISTRO S2 - frecuencia ** }
    a0(1,1,1);
    txbyte($20);
    a0(0,0,1);
    lectu:=rxbyte;
    write('Divisor de frecuencia  :  ');writeln(lectu);
    a0(1,1,1);
    txbyte($C1);
    a0(0,0,1);
  end;
(*PROGRAMA PRINCIPAL **************************************************)

begin;
 x:=200;
 repeat
  clrscr;
  Initpuerto;
  initpcf;
  writeln('...Fin de configuracion....');
  delay(500);
  monitor;
  WRITELN('Oprima ENTER para iniciar la adquisicion de datos');
  readln;
  modulo:=1;
  Mreceiver($57,x);                {DIR I2C DEL MODULO,#juegos}
  clrscr;
  writeln('Cuantos juegos mas desea adquirir?');
  readln(x);
until(x=0);
end.
```

```
$MOD552
$TITLE(almacenamiento en memoria RAM loop V1.0)
$PAGEWIDTH(132)
;$DEBUG
;$OBJECT
;$NOPAGING
;                   UNAM
;                   INSTITUTO DE INGENIERIA
;                   INSTRUMENTACION
; TITULO            TRANSFERENCIA DE DATOS DEL
;                   CONV A/D POR I2C EN UN 80C552
; FECHA
; VERSION           1.0
DIREC  EQU          71H
TOPEH       EQU     72H
TOPEL       EQU     73H
            ORG 00H
            AJMP INICIO
            ORG 80H
INICIO:     MOV  R4,#00H
            MOV  DPTR,#02001H       ;PRIMERA LOCALIDAD DE RAM
            MOV   DIREC,#0AFH       ;DIREC I2C=57H, Y GC=1
            MOV  S1ADR,DIREC        ;INIT I2C
            ORL  S1CON,#44H
            ANL  S1CON,#0C7H
LOOP0:       CJNE       R4,#05H,CONT ;A LA QUINTA CONVERSION
            CALL LDELAY             ;ESPERAR ANTES DEL SIG CICLO
            CALL I2C                ;VERIFICAR LLAMADA I2C
            MOV  R4,#00H            ;INICIAR NUEVAMENTE
CONT:            MOV  ADCON,R4      ;SELECCION DE CANAL E
            ORL  ADCON,#08H         ;INICIALIZACION CONV A/D
TERMAD:          MOV  A,ADCON       ;VERIFICAR TERMINACION DE
            JNB  ACC.4,TERMAD       ;CONVERSION A/D
            MOV  ADCON,#00H         ;CONCLUIR CONVERSION A/D
            INC  R4                 ;CAMBIO DE CANAL
            CALL DXARAM             ;PASAR EL DATO A LA RAM
            CALL RAMAP4             ;VERIFICAR A TRAVES DE P4
            CJNE R4,#05H,CUCU
            INC  DPTR
            MOV  A,#99H
            MOVX@DPTR,A
CUCU:       MOV  A,DPH              ;VERIFICAR QUE NO SE DE
            CJNE A,#07FH,BRINCO     ;UN OVERFLOW DE MEMORIA
            MOV  A,DPL
            CJNE A,#0FFH,BRINCO
            AJMP LOOP0
```

```
BRINCO:  INC  DPTR
         AJMP LOOP0
;********RUTINA DE TRANSFERENCIA I2C**************
I2C:     MOV  TOPEL,DPL
         MOV  TOPEH,DPH
         MOV  A,S1STA
         CJNE A,#0A8H,FIN          ;RECIBIO DIR ESCLAVA?
         MOV  DPTR,#2001H
         ANL  S1CON,#0E7H          ;PERMITE OTRA TRANSMISION
         ORL  S1CON,#04H
         MOV  S1DAT,#99H
NOMA1:   MOV  A,S1STA              ;ASEGURAR TRANSMISION
         CJNE A,#0B8H,NOMA1
         MOV  S1DAT,#99H
SIG:     MOVXA,@DPTR               ;TRANSMISION I2C DE DATOS
         MOV  S1DAT,A
OTRA:    MOV  A,S1STA              ;RETRANSMITIR HASTA ACK
         CJNE A,#0C0H,PO1
         AJMP FUERA
PO1:     CJNE A,#0B8H,OTRA
         ANL  S1CON,#0E7H          ;PERMITE OTRA TRANSMISION
         ORL  S1CON,#04H
         INC  DPTR                 ;SIGUIENTE DATO
         MOV  A,DPL
         CJNE A,TOPEL,SIG          ;SE LLEGO AL TOPE?
         MOV  A,TOPEH
         CJNE A,TOPEH,SIG
         CALL MONITOR
                                   ;ULTIMO DATO
         MOV  A,S1DAT
         MOVXA,@DPTR
OTRA2:   ANL  S1CON,#0E3H          ;PREP PARA TRANSM ULT BYTE
         MOV  A,S1STA
         CJNE A,#0C8H,OTRA2
         MOV  DPTR,#2001H          ;REINICIALIZACION I2C
FIN:     ORL  S1CON,#04H
         RET
MONITOR: MOV  ADCON,#00H           ;SELECCION DE CANAL E
         ORL  ADCON,#08H           ;INICIALIZACION CONV A/D
TE1:     MOV  A,ADCON              ;VERIFICAR TERMINACION DE
         JNB  ACC.4,TE1            ;CONVERSION A/D
         MOV  ADCON,#00H           ;CONCLUIR CONVERSION A/D
         MOV  R0,ADCH
         MOV  ADCON,#01H           ;SELECCION DE CANAL E
         ORL  ADCON,#08H           ;INICIALIZACION CONV A/D
TE2:     MOV  A,ADCON              ;VERIFICAR TERMINACION DE
```

```
        JNB  ACC.4,TE2          ;CONVERSION A/D
        MOV  ADCON,#00H         ;CONCLUIR CONVERSION A/D
        MOV  R1,ADCH
        MOV  ADCON,#02H         ;SELECCION DE CANAL E
        ORL  ADCON,#08H         ;INICIALIZACION CONV A/D
TE3:    MOV  A,ADCON            ;VERIFICAR TERMINACION DE
        JNB  ACC.4,TE3          ;CONVERSION A/D
        MOV  ADCON,#00H         ;CONCLUIR CONVERSION A/D
        MOV  R2,ADCH
        MOV  ADCON,#03H         ;SELECCION DE CANAL E
        ORL  ADCON,#08H         ;INICIALIZACION CONV A/D
TE4:    MOV  A,ADCON            ;VERIFICAR TERMINACION DE
        JNB  ACC.4,TE4          ;CONVERSION A/D
        MOV  ADCON,#00H         ;CONCLUIR CONVERSION A/D
        MOV  R3,ADCH
        MOV  ADCON,#04          ;SELECCION DE CANAL E
        ORL  ADCON,#08H         ;INICIALIZACION CONV A/D
TE5:    MOV  A,ADCON            ;VERIFICAR TERMINACION DE
        JNB  ACC.4,TE5          ;CONVERSION A/D
        MOV  ADCON,#00H         ;CONCLUIR CONVERSION A/D
        MOV  R4,ADCH
        MOV  S1DAT,#99H
S1:     MOV  A,S1STA            ;ASEGURAR TRANSMISION
        CJNE A,#0C0H,PO2
        AJMP FUERA
PO2:    CJNE A,#0B8H,S1
        ANL  S1CON,#0E7H        ;PERMITE OTRA TRANSMISION
        ORL  S1CON,#04H
        MOV  S1DAT,R0
S2:     MOV  A,S1STA            ;ASEGURAR TRANSMISION
        CJNE A,#0C0H,PO3
        AJMP FUERA
PO3:    CJNE A,#0B8H,S2
        ANL  S1CON,#0E7H        ;PERMITE OTRA TRANSMISION
        ORL  S1CON,#04H
        MOV  S1DAT,R1
S3:     MOV  A,S1STA            ;ASEGURAR TRANSMISION
        CJNE A,#0C0H,PO4
        AJMP FUERA
PO4:    CJNE A,#0B8H,S3
        ANL  S1CON,#0E7H        ;PERMITE OTRA TRANSMISION
        ORL  S1CON,#04H
        MOV  S1DAT,R2
S4:     MOV  A,S1STA            ;ASEGURAR TRANSMISION
        CJNE A,#0C0H,PO5
        AJMP FUERA
```

```
PO5:        CJNE A,#0B8H,S4
            ANL  S1CON,#0E7H        ;PERMITE OTRA TRANSMISION
            ORL  S1CON,#04H
            MOV  S1DAT,R3
S5:         MOV  A,S1STA            ;ASEGURAR TRANSMISION
            CJNE A,#0C0H,PO6
            AJMP FUERA
PO6:        CJNE A,#0B8H,S5
            ANL  S1CON,#0E7H        ;PERMITE OTRA TRANSMISION
            ORL  S1CON,#04H
            MOV  S1DAT,R4
S6:         MOV  A,S1STA            ;ASEGURAR TRANSMISION
            CJNE A,#0C0H,PO7
            AJMP FUERA
PO7:        CJNE A,#0B8H,S6
            ANL  S1CON,#0E7H        ;PERMITE OTRA TRANSMISION
            ORL  S1CON,#04H
            AJMP MONITOR
            RET
FUERA:      MOV  S1CON,#14H
            MOV  P4,#55H
            AJMP INICIO
DXARAM:     MOV  A,ADCH             ;SACAR EL DATO DEL CONV
            CJNE A,#99H,NOSINC
            MOV  A,#98H
NOSINC:     MOVX      @DPTR,A       ;Y ESCRIBIRLO A RAM
            RET
CONTARAM:MOV  A,R4                  ;NUMERO DE SENSOR EN RAM
            MOVX@DPTR,A
            RET
RAMAP4:     MOV  A,#00H             ;RECUPERAR EL ULTIMO DATO
            MOVX      A,@DPTR           ;GUARDADO EN RAM
            MOV  P4,A
            RET
LDELAY:     MOV  R1,#00H            ;PAUSA LARGA ENTRE
NESTED:     MOV  R0,#00H            ;CONVERSION Y CONVERSION
VUELTA:     INC  R0
            CJNE R0,#0FFH,VUELTA
            INC  R1
            CJNE R1,#0FFH,NESTED
            RET
            END
```

```
MENU.FRM - 1

VERSION 2.00
Begin Form menuform
   AutoRedraw      =   -1  'True
   BackColor       =   &H008080000&
   Caption         =   "Adquisición de Datos"
   ClientHeight    =   4536
   ClientLeft      =   1368
   ClientTop       =   1464
   ClientWidth     =   7368
   ForeColor       =   &H000000000&
   Height          =   5088
   Icon            =   (Icon)
   Left            =   1320
   LinkMode        =   1  'Source
   LinkTopic       =   "Form2"
   ScaleHeight     =   4536
   ScaleWidth      =   7368
   Top             =   960
   Width           =   7464
   WindowState     =   2  'Maximized
   Begin Frame Frame3
      BackColor       =   &H008080000&
      Caption         =   "Abrir Archivo"
      ForeColor       =   &H00E0E0E0&
      Height          =   1812
      Left            =   480
      TabIndex        =   10
      Top             =   600
      Visible         =   0   'False
      Width           =   3492
      Begin TextBox Text2
         BackColor       =   &H00E0E0E0&
         ForeColor       =   &H000000000&
         Height          =   288
         Left            =   360
         TabIndex        =   6
         Text            =   ""
         Top             =   840
         Visible         =   0   'False
         Width           =   2652
      End
      Begin CommandButton Command6
         Caption         =   "Cancelar"
         Height          =   252
         Left            =   1800
         TabIndex        =   12
         Top             =   1440
         Visible         =   0   'False
         Width           =   1212
      End
      Begin CommandButton Command5
         Caption         =   "Aceptar"
         Height          =   252
         Left            =   240
         TabIndex        =   11
         Top             =   1440
         Visible         =   0   'False
         Width           =   1092
      End
      Begin Label Label3
         BackColor       =   &H008080000&
         Caption         =   "Nombre del Archivo"
         ForeColor       =   &H00E0E0E0&
         Height          =   252
         Left            =   360
         TabIndex        =   13
         Top             =   480
         Visible         =   0   'False
         Width           =   2532
      End
   End
   Begin Frame Frame2
      BackColor       =   &H008080000&
      Caption         =   "Conservar"
      ForeColor       =   &H00E0E0E0&
```

```
      Height          =   1812
      Left            =   480
      TabIndex        =   4
      Top             =   600
      Visible         =   0   'False
      Width           =   3492
      Begin CommandButton Command4
         Caption         =   "Cancelar"
         Height          =   252
         Left            =   2040
         TabIndex        =   9
         Top             =   1320
         Visible         =   0   'False
         Width           =   1092
      End
      Begin CommandButton Command3
         Caption         =   "Aceptar"
         Default         =   -1  'True
         Height          =   252
         Left            =   360
         TabIndex        =   8
         Top             =   1320
         Visible         =   0   'False
         Width           =   1092
      End
      Begin TextBox Text1
         BackColor       =   &H00E0E0E0&
         ForeColor       =   &H00000000&
         Height          =   288
         Left            =   240
         TabIndex        =   7
         Text            =   "c:\temp\test1.txt"
         Top             =   720
         Visible         =   0   'False
         Width           =   2652
      End
      Begin Label Label2
         BackColor       =   &H00808000&
         Caption         =   "Nombre del archivo"
         ForeColor       =   &H00E0E0E0&
         Height          =   372
         Left            =   240
         TabIndex        =   5
         Top             =   360
         Visible         =   0   'False
         Width           =   2892
      End
   End
   Begin Frame Frame1
      BackColor       =   &H00808000&
      Caption         =   "Borrar archivo de datos"
      ForeColor       =   &H00C0C0C0&
      Height          =   2772
      Left            =   2160
      TabIndex        =   0
      Top             =   1080
      Visible         =   0   'False
      Width           =   3972
      Begin CommandButton Command2
         Caption         =   "NO"
         Height          =   372
         Left            =   2280
         TabIndex        =   1
         Top             =   2160
         Visible         =   0   'False
         Width           =   852
      End
      Begin CommandButton Command1
         Caption         =   "SI"
         Height          =   372
         Left            =   720
         TabIndex        =   3
         Top             =   2160
         Visible         =   0   'False
         Width           =   852
```

MENU.FRM - 3

```
      End
      Begin Label Label1
         BackColor       =   &H008080000&
         Caption         =   "Esto borrará TODOS los datos adquiridos hasta ahora !
         FontBold        =   -1  'True
         FontItalic      =   0   'False
         FontName        =   "MS Sans Serif"
         FontSize        =   13.8
         FontStrikethru  =   0   'False
         FontUnderline   =   0   'False
         ForeColor       =   &H00E0E0E0&
         Height          =   1572
         Left            =   480
         TabIndex        =   2
         Top             =   240
         Visible         =   0   'False
         Width           =   3252
      End
   End
   Begin Menu mnuArchivo
      Caption         =   "&Archivo"
      Begin Menu Abrir
         Caption         =   "&Abrir"
      End
      Begin Menu Keepit
         Caption         =   "&Guardar como..."
      End
      Begin Menu Borrar
         Caption         =   "&Eliminar"
      End
      Begin Menu Salir
         Caption         =   "&Salir"
      End
   End
   Begin Menu Modulo
      Caption         =   "&Módulo"
      Begin Menu ModUno
         Caption         =   "&1"
         Checked         =   -1  'True
      End
      Begin Menu ModDos
         Caption         =   "&2"
      End
      Begin Menu ModTres
         Caption         =   "&3"
      End
      Begin Menu ModCua
         Caption         =   "&4"
      End
   End
   Begin Menu Parametro
      Caption         =   "&Parámetro"
      Begin Menu Temp
         Caption         =   "&Temperatura"
      End
      Begin Menu Humedad
         Caption         =   "&Humedad"
         Checked         =   -1  'True
      End
      Begin Menu Presencia
         Caption         =   "&Presencia"
      End
      Begin Menu Ilum
         Caption         =   "&Iluminación"
      End
      Begin Menu Calidad
         Caption         =   "&Calidad del Aire"
      End
   End
   Begin Menu Resultados
      Caption         =   "&Resultados"
      Begin Menu Grafica
         Caption         =   "&Gráfica"
      End
      Begin Menu Reporte
```

```
MENU.FRM - 4

        Caption          =   "&Reporte"
      End
      Begin Menu Monitor
        Caption          =   "&Monitoreo"
      End
   End
   Begin Menu Ay
      Caption         =   "A&yuda"
      Begin Menu jbkjlrl
        Caption          =   "&Autores"
      End
      Begin Menu jaja
        Caption          =   "A&yuda"
      End
   End
End
```

MENU.FRM - 1

```
Sub Bus_Click ()

End Sub

Sub Config_Click ()
End Sub

Sub Knuconfig_Click ()
End Sub

Sub Abrir_Click ()
Frame3.Visible = True
Text2.Visible = True
Label3.Visible = True
Command5.Visible = True
Command6.Visible = True
Command5.Default = True
End Sub

Sub Borrar_Click ()
Frame1.Visible = True
Label1.Visible = True
Command1.Visible = True
Command2.Visible = True
End Sub

Sub Calidad_Click ()
    Temp.Checked = 0
    Presencia.Checked = 0
    Humedad.Checked = 0
    Ilum.Checked = 0
    Calidad.Checked = 1
End Sub

Sub Command1_Click ()
Label1.Visible = False
Frame1.Visible = False
Command1.Visible = False
Command1.Visible = False
Kill Archivo$

End Sub

Sub Command2_Click ()
Label1.Visible = False
Frame1.Visible = False
Command1.Visible = False
Command1.Visible = False

End Sub

Sub Command3_Click ()
Frame2.Visible = False
Label2.Visible = False
Text1.Visible = False
Command3.Visible = False
Command4.Visible = False
FileCopy Archivo$, Text1.Text

End Sub

Sub Command4_Click ()
Frame2.Visible = False
Label2.Visible = False
Text1.Visible = False
Command3.Visible = False
Command4.Visible = False
```

```
End Sub

Sub Command5_Click ()

Frame3.Visible = False
Label3.Visible = False
Text2.Visible = False
Command5.Visible = False
Command6.Visible = False
Archivo$ = Text2.Text
If Dir$(Archivo$) = "" Then
 MsgBox "Archivo Inexistente"
 Text2.Text = ""
End If

End Sub

Sub Command6_Click ()
Frame3.Visible = False
Label3.Visible = False
Text2.Visible = False
Command5.Visible = False
Command6.Visible = False


End Sub

Sub Form_Load ()
On Error Resume Next
FileCopy "c:\tmp.tmp", "c:\j&jl.tmp"
Archivo$ = "C:\J&JL.tmp"

End Sub



Sub Form_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)
Unload ResForm
Unload Moni
If Tag = "GU" Or Tag = "RMU" Then

 If ModUno.Checked = True Then
    ModDos_Click
    ElseIf ModDos.Checked = True Then ModTres_Click
    ElseIf ModTres.Checked = True Then ModCua_Click
    ElseIf ModCua.Checked = True Then ModUno_Click
 End If
End If

If Tag = "GD" Or Tag = "RMD" Then
 If ModUno.Checked = True Then
  ModCua_Click
  ElseIf ModDos.Checked = True Then ModUno_Click
  ElseIf ModTres.Checked = True Then ModDos_Click
  ElseIf ModCua.Checked = True Then ModTres_Click
 End If

End If

If Tag = "GPU" Or Tag = "RPU" Then

 If Temp.Checked = True Then
    Humedad_Click
    ElseIf Humedad.Checked = True Then Presencia_Click
    ElseIf Presencia.Checked = True Then Ilum_Click
    ElseIf Ilum.Checked = True Then Calidad_Click
    ElseIf Calidad.Checked = True Then Temp_Click
 End If
End If

If Tag = "GPD" Or Tag = "RPD" Then
 If Temp.Checked = True Then
    Calidad_Click
```

```
    ElseIf Humedad.Checked = True Then Temp_Click
    ElseIf Presencia.Checked = True Then Humedad_Click
    ElseIf Ilum.Checked = True Then Presencia_Click
    ElseIf Calidad.Checked = True Then Ilum_Click
 End If
End If

If Tag = "GPD" Or Tag = "GPU" Or Tag = "GU" Or Tag = "GD" Then Grafica_Click
If Tag = "RMD" Or Tag = "RMU" Or Tag = "RPU" Or Tag = "RPD" Then Reporte_Click

If Tag = "R2G" Then Grafica_Click
If Tag = "G2R" Then Reporte_Click

If Tag = "REFG" Then Grafica_Click
If Tag = "REFR" Then Reporte_Click


End Sub

Sub Form_Unload (Cancel As Integer)

MenuForm.Enabled = True
MenuForm.Show

End Sub

Sub Grafica_Click ()

ResForm.Picture2.Visible = True

ResForm.pic.Visible = True
Tag = ""

k$ = Dir$(Archivo$)
If k$ <> "" Then
 If ModUno.Checked = True Then a = 1
 If ModDos.Checked = True Then a = 2
 If ModTres.Checked = True Then a = 3
 If ModCua.Checked = True Then a = 4

 If Temp.Checked = True Then
     b = "Temperatura"
     be = 1
     ResForm.hot.Visible = True
     c = "Lecturas, en ºC"
     f = 16
     End If
 If Humedad.Checked = True Then
     b = "Humedad"
     be = 2
     ResForm.nube.Visible = True
     c = "Lecturas, en %"
     f = 100
     End If
 If Presencia.Checked = True Then
     b = "Presencia"
     be = 3
     c = ""
     ResForm.pres.Visible = True
     f = 100
     End If
 If Ilum.Checked = True Then
     b = "Iluminacion"
     be = 4
     c = "Lecturas, en %"
     ResForm.Ilum.Visible = True
     f = 100
     End If
 If Calidad.Checked = True Then
     b = "Calidad del Aire"
     be = 5
     c = ""
     ResForm.Imecas.Visible = True
     f = 280
     End If
```

```
MENU.FRM - 4

 ResForm.Label2.Caption = b
 ResForm.Label5.Caption = c
 ResForm.Show
 MenuForm.Enabled = False


Ram grafica
ResForm.Graph1.Visible = True
ResForm.Graph1.NumPoints = 3
ResForm.Graph1.ThisPoint = 1
j = 0
       Open Archivo$ For Input As #1
       Do While Not EOF(1)
           ModAdhoc = 0
           ParAdhoc = 0
           Rem
           For i = 0 To 4
                Input #1, datos
                If i = 0 And datos = a Then ModAdhoc = 1
                If i = 1 And datos = be Then ParAdhoc = 1
                If ModAdhoc = 1 And ParAdhoc = 1 Then

                   If i = 3 Then morig = Int(datos)
                   If i = 4 Then
                    If be = 3 Then datos = datos - 50
                    ResForm.Graph1.GraphData = Int(datos)
                    msec = (morig - Int(morig / 100) * 100)
                    mmin = (morig - Int(morig / 10000) * 10000) - msec
                    mhor = morig - mmin - msec
                    mnuev = mhor + Int(mmin / 60 * 100) + Int(msec / 60 * 100) * 1.67
                    If j < 5 Then rest = mnuev
                    ResForm.Graph1.XPosData = (mnuev - rest)
                    ResForm.Graph1.NumPoints = ResForm.Graph1.NumPoints + 1
                    ResForm.Graph1.ThisPoint = ResForm.Graph1.ThisPoint + 1
                    End If
                End If

          Next i
            j = j + 1

       Loop
       Close #1
       ResForm.Label1.Visible = True
       ResForm.Label2.Visible = True
       ResForm.Label1.Caption = a
 Else
 MsgBox ("Defina el archivo a utilizar (Archivo - Abrir)"), 48, "ERROR"
 End If


Grafica.Tag = 0







End Sub

Sub Humedad_Click ()
    Temp.Checked = 0
    Presencia.Checked = 0
    Humedad.Checked = 1
    Ilum.Checked = 0
    Calidad.Checked = 0
End Sub

Sub Ilum_Click ()
    Temp.Checked = 0
    Presencia.Checked = 0
    Humedad.Checked = 0
    Ilum.Checked = 1
```

```
    Calidad.Checked = 0
End Sub

Sub jaja_Click ()
MsgBox ("No hay ayuda. El programa es suficientemente intuitivo."), 48, "Ayuda???!!!!"
End Sub

Sub jbkjlrl_Click ()
MsgBox "Elaborado por Jacques Bessoudo Korzenny y José Luis Rodríguez López . Informes:    +52(5)
544-25-00      jrl@pumas.iingen.unam.mx"
End Sub

Sub Keepit_Click ()
Frame2.Visible = True
Label2.Visible = True
Text1.Visible = True
Command3.Visible = True
Command4.Visible = True
Command3.Default = True
Text1.SelLength = Len(Text1.Text)

End Sub

Sub ModCua_Click ()
    ModUno.Checked = 0
    ModDos.Checked = 0
    ModTres.Checked = 0
    ModCua.Checked = 1

End Sub

Sub ModDos_Click ()
    ModUno.Checked = 0
    ModDos.Checked = 1
    ModTres.Checked = 0
    ModCua.Checked = 0

End Sub

Sub ModTres_Click ()
    ModUno.Checked = 0
    ModDos.Checked = 0
    ModTres.Checked = 1
    ModCua.Checked = 0

End Sub

Sub ModUno_Click ()
    ModUno.Checked = 1
    ModDos.Checked = 0
    ModTres.Checked = 0
    ModCua.Checked = 0

End Sub

Sub Monitor_Click ()
Moni.Show
Moni.SetFocus
MenuForm.Enabled = False

End Sub

Sub Presencia_Click ()
    Temp.Checked = 0
    Presencia.Checked = 1
    Humedad.Checked = 0
    Ilum.Checked = 0
    Calidad.Checked = 0
End Sub

Sub Reporte_Click ()
ResForm.pic.Visible = True
ResForm.Picture3.Visible = True

Tag = ""
```

```
k$ = Dir$(Archivo$)
If k$ <> "" Then
 If ModUno.Checked = True Then a = 1
 If ModDos.Checked = True Then a = 2
 If ModTres.Checked = True Then a = 3
 If ModCua.Checked = True Then a = 4
 ResForm.Label1.Caption = a
 If Temp.Checked = True Then
     b = "Temperatura"
     be = 1
     ResForm.hot.Visible = True
     c = "Lecturas, en ºC"
     End If
 If Humedad.Checked = True Then
     b = "Humedad"
     be = 2
     ResForm.nube.Visible = True
     c = "Lecturas, en %"
     End If
 If Presencia.Checked = True Then
     b = "Presencia"
     be = 3
     c = ""
     ResForm.pres.Visible = True
     End If
 If Ilum.Checked = True Then
     b = "Iluminacion"
     be = 4
     c = "Lecturas, en %"
     ResForm.Ilum.Visible = True
     End If
 If Calidad.Checked = True Then
     b = "Calidad del Aire"
     be = 5
     c = ""
     ResForm.Imecas.Visible = True
     End If
 ResForm.Label2.Caption = b
 ResForm.Label5.Caption = c
 ResForm.Show
 MenuForm.Enabled = False
 ResForm.Grid1.Visible = True
 ResForm.Label1.Visible = True
 ResForm.Label2.Visible = True
 ResForm.Grid1.ColWidth(0) = 812
 ResForm.Grid1.ColWidth(1) = 812
 ResForm.Grid1.ColWidth(2) = 812
 ResForm.Grid1.Col = 0
 ResForm.Grid1.Row = 0
 ResForm.Grid1.Text = "Fecha"
 ResForm.Grid1.Col = 1
 ResForm.Grid1.Text = "Hora"
 ResForm.Grid1.Col = 2
 ResForm.Grid1.Text = "Lectura"
 j = 0
       Open Archivo$ For Input As #1
       Do While Not EOF(1)
        ResForm.Grid1.Row = j + 1
        ModAdhoc = 0
        ParAdhoc = 0
          For i = 0 To 4
           Input #1, datos
           If i = 0 And datos = a Then ModAdhoc = 1
           If i = 1 And datos = be Then ParAdhoc = 1
           If i >= 2 Then
             ResForm.Grid1.Col = i - 2
             If ModAdhoc = 1 And ParAdhoc = 1 Then
               If be = 3 Then datos = datos - 50
               ResForm.Grid1.Text = datos
               Else
               ResForm.Grid1.Text = ""

             End If
           End If
           ResForm.Grid1.Rows = j + 3
```

```
          Next i
          If ModAdhoc = 1 And ParAdhoc = 1 Then j = j + 1

        Loop
        Close #1

        ResForm.Grid1.Rows = ResForm.Grid1.Rows - 1
 Else
 MsgBox ("Defina el archivo a utilizar (Archivo - Abrir)"), 48, "ERROR"
 End If

 End Sub

Sub Salir_Click ()
End
End Sub

Sub Temp_Click ()
    Temp.Checked = 1
    Presencia.Checked = 0
    Humedad.Checked = 0
    Ilum.Checked = 0
    Calidad.Checked = 0
End Sub
```

```
MONI.FRM - 1

VERSION 2.00
Begin Form Moni
   BackColor       =   &H008080000&
   Caption         =   "Monitor de Parámetros"
   ClientHeight    =   5556
   ClientLeft      =   72
   ClientTop       =   1236
   ClientWidth     =   9468
   FillColor       =   &H008080000&
   Height          =   6108
   Left            =   24
   LinkTopic       =   "Form1"
   ScaleHeight     =   5556
   ScaleWidth      =   9468
   Top             =   732
   Width           =   9564
   Begin Timer Timer1
      Interval        =   500
      Left            =   8232
      Top             =   2880
   End
   Begin Gauge Gauge4
      Autosize        =   -1  'True
      BackColor       =   &H00E0E0E0&
      ForeColor       =   &H00FFFFFF&
      Height          =   936
      InnerBottom     =   5
      InnerLeft       =   5
      InnerRight      =   5
      InnerTop        =   5
      Left            =   1515
      Max             =   100
      NeedleWidth     =   1
      Picture         =   (Bitmap)
      Style           =   2  ''Semi' Needle
      TabIndex        =   15
      Top             =   3270
      Value           =   80
      Width           =   1824
   End
   Begin Gauge Gauge5
      Autosize        =   -1  'True
      BackColor       =   &H00E0E0E0&
      ForeColor       =   &H00404040&
      Height          =   1155
      InnerBottom     =   5
      InnerLeft       =   5
      InnerRight      =   5
      InnerTop        =   5
      Left            =   5145
      Max             =   100
      NeedleWidth     =   1
      Picture         =   (none)
      Style           =   1  'Vertical Bar
      TabIndex        =   12
      Top             =   3270
      Value           =   53
      Width           =   2295
   End
   Begin Gauge Gauge3
      Autosize        =   -1  'True
      BackColor       =   &H00E0E0E0&
      ForeColor       =   &H00E0FFFF&
      Height          =   1185
      InnerBottom     =   7
      InnerLeft       =   7
      InnerRight      =   7
      InnerTop        =   7
      Left            =   6855
      Max             =   100
      NeedleWidth     =   1
      Picture         =   (none)
      Style           =   1  'Vertical Bar
      TabIndex        =   7
      Top             =   855
```

```
      Value           =   51
      Width           =   2295
   End
   Begin Gauge Gauge2
      Autosize        =   -1  'True
      BackColor       =   &H00E0E0E0&
      ForeColor       =   &H00FFFFFF&
      Height          =   936
      InnerBottom     =   5
      InnerLeft       =   5
      InnerRight      =   5
      InnerTop        =   5
      Left            =   3960
      Max             =   100
      NeedleWidth     =   1
      Picture         =   (Bitmap)
      Style           =   2  ''Semi' Needle
      TabIndex        =   5
      Top             =   855
      Value           =   17
      Width           =   1824
   End
   Begin Gauge Gauge1
      Autosize        =   -1  'True
      ForeColor       =   &H000040C0&
      Height          =   936
      InnerBottom     =   5
      InnerLeft       =   5
      InnerRight      =   5
      InnerTop        =   5
      Left            =   840
      Max             =   50
      NeedleWidth     =   1
      Picture         =   (Bitmap)
      Style           =   2  ''Semi' Needle
      TabIndex        =   1
      Top             =   885
      Value           =   25
      Width           =   1824
   End
   Begin CommandButton Command1
      Caption         =   "Regresar"
      Height          =   252
      Left            =   3105
      TabIndex        =   0
      Top             =   4950
      Width           =   2052
   End
   Begin Label Label10
      Alignment       =   2  'Center
      BackColor       =   &H00808000&
      Caption         =   "Label10"
      ForeColor       =   &H00E0E0E0&
      Height          =   255
      Left            =   5415
      TabIndex        =   14
      Top             =   4515
      Width           =   1815
   End
   Begin Label Label9
      Alignment       =   2  'Center
      BackColor       =   &H00808000&
      Caption         =   "Calidad del Aire"
      ForeColor       =   &H00E0E0E0&
      Height          =   375
      Left            =   5385
      TabIndex        =   13
      Top             =   2790
      Width           =   1815
   End
   Begin Label Label8
      Alignment       =   2  'Center
      BackColor       =   &H00808000&
      Caption         =   "Label8"
      ForeColor       =   &H00E0E0E0&
```

```
      Height          =   255
      Left            =   1680
      TabIndex        =   11
      Top             =   4530
      Width           =   1815
   End
   Begin Label Label7
      Alignment       =   2  'Center
      BackColor       =   &H008080000&
      Caption         =   "Iluminación, en % de la máxima"
      ForeColor       =   &H00E0E0E0&
      Height          =   375
      Left            =   1785
      TabIndex        =   10
      Top             =   2790
      Width           =   1815
   End
   Begin Label Label6
      Alignment       =   2  'Center
      BackColor       =   &H008080000&
      Caption         =   "No hay nadie"
      ForeColor       =   &H00E0E0E0&
      Height          =   255
      Left            =   6840
      TabIndex        =   9
      Top             =   2100
      Width           =   1815
   End
   Begin Label Label5
      Alignment       =   2  'Center
      BackColor       =   &H008080000&
      Caption         =   "Presencia"
      ForeColor       =   &H00E0E0E0&
      Height          =   375
      Left            =   7155
      TabIndex        =   8
      Top             =   465
      Width           =   1815
   End
   Begin Label Label4
      Alignment       =   2  'Center
      BackColor       =   &H008080000&
      Caption         =   "Label4"
      ForeColor       =   &H00E0E0E0&
      Height          =   252
      Left            =   4020
      TabIndex        =   6
      Top             =   1956
      Width           =   1812
   End
   Begin Label Label3
      Alignment       =   2  'Center
      BackColor       =   &H008080000&
      Caption         =   "Humedad Relativa, en %"
      ForeColor       =   &H00E0E0E0&
      Height          =   375
      Left            =   4185
      TabIndex        =   4
      Top             =   375
      Width           =   1815
   End
   Begin Label Label2
      Alignment       =   2  'Center
      BackColor       =   &H008080000&
      Caption         =   "Temperatura,     en ºC"
      ForeColor       =   &H00E0E0E0&
      Height          =   375
      Left            =   1140
      TabIndex        =   3
      Top             =   435
      Width           =   1695
   End
   Begin Label Label1
      Alignment       =   2  'Center
      BackColor       =   &H008080000&
```

```
Sub Gauge6_Change ()
Label8.Caption = Value
End Sub

Sub Command1_Click ()
Menuform.Show
moni.Hide
Menuform.Enabled = True
Menuform.ModUno.Checked = M1.Checked
Menuform.ModDos.Checked = M2.Checked
Menuform.ModTres.Checked = M3.Checked
Menuform.ModCua.Checked = M4.Checked

End Sub

Sub Form_GotFocus ()
M1.Checked = Menuform.ModUno.Checked
M2.Checked = Menuform.ModDos.Checked
M3.Checked = Menuform.ModTres.Checked
M4.Checked = Menuform.ModCua.Checked


End Sub

Sub Form_Load ()
 On Error Resume Next
 If Menuform.ModUno.Checked = True Then
 M1.Checked = True
 M2.Checked = False
 M3.Checked = False
 M4.Checked = False
 a = 1
 End If
 If Menuform.ModDos.Checked = True Then
 M1.Checked = False
 M2.Checked = True
 M3.Checked = False
 M4.Checked = False
 a = 2
 End If
 If Menuform.ModTres.Checked = True Then
 M3.Checked = True
 a = 3
 End If
 If Menuform.ModCua.Checked = True Then
 M4.Checked = True
 a = 4
 End If
donde$ = "c:\moni.tmp"

        Open donde$ For Input As #1
        Do While Not EOF(1)
        ModAdhoc = 0
        For i = 0 To 4
           Input #1, datos
           If i = 0 And datos = a Then ModAdhoc = 1
           If i = 1 Then ParAdhoc = datos
           If i >= 4 Then
                If ModAdhoc = 1 Then
                 If ParAdhoc = 1 Then Gauge1.Value = datos
                 If ParAdhoc = 2 Then Gauge2.Value = datos
                 If ParAdhoc = 3 Then Gauge3.Value = datos
                 If ParAdhoc = 4 Then Gauge4.Value = datos
                 If ParAdhoc = 5 Then Gauge5.Tag = datos
                Else
                Gauge1.Value = 0
                Gauge2.Value = 0
                Gauge3.Value = 0
                Gauge4.Value = 0
                Gauge5.Tag = 0
                End If
           End If
      Next i
```

```
MONI.FRM - 4

      Caption         =   "Label1"
      ForeColor       =   &H00E0E0E0&
      Height          =   252
      Left            =   768
      TabIndex        =   2
      Top             =   2004
      Width           =   1932
   End
   Begin Menu Mod
      Caption         =   "&Modulo"
      Begin Menu M1
         Caption          =   "&1"
         Checked          =   -1  'True
      End
      Begin Menu M2
         Caption          =   "&2"
      End
      Begin Menu M3
         Caption          =   "&3"
      End
      Begin Menu M4
         Caption          =   "&4"
      End
   End
End
```

```
        Loop
        Close #1

Gauge5.Value = 50
Label1.Caption = Gauge1.Value
Label4.Caption = Gauge2.Value
Label8.Caption = Gauge4.Value
Label10.Caption = Gauge5.Value
If Gauge3.Value > 50 Then
    Gauge3.ForeColor = &HC0&
    Label6.Caption = "Hay alguien"
Else
    Gauge3.ForeColor = &HE0FFFF
    Label6.Caption = "No hay nadie"
End If

If Gauge5.Tag >= 50 And Gauge5.Tag < 75 Then
    Gauge5.ForeColor = &HE0FFFF
    Label10.Caption = "Regular"
ElseIf Gauge5.Tag > 24 And Gauge5.Tag < 50 Then
    Label10.Caption = "Mala"
    Gauge5.ForeColor = &H40C0&
ElseIf Gauge5.Tag < 25 Then
Label10.Caption = "Pésima"
    Gauge5.ForeColor = &H404040
Else
    Label10.Caption = "Buena"
    Gauge5.ForeColor = &HC0FFC0
End If

End Sub

Sub Form_Unload (Cancel As Integer)
Menuform.Show
moni.Hide
Menuform.Enabled = True
Menuform.ModUno.Checked = M1.Checked
Menuform.ModDos.Checked = M2.Checked
Menuform.ModTres.Checked = M3.Checked
Menuform.ModCua.Checked = M4.Checked



End Sub

Sub Gauge1_Change ()
Label1.Caption = Value
End Sub

Sub Gauge2_Change ()
Label4.Caption = Value
End Sub

Sub Gauge3_Change ()
Label6.Caption = Value
End Sub

Sub Gauge5_Change ()
Label10.Caption = Value
End Sub

Sub M1_Click ()
M1.Checked = True
Menuform.ModUno.Checked = True
M2.Checked = False
Menuform.ModDos.Checked = False
M3.Checked = False
Menuform.ModTres.Checked = False
M4.Checked = False
Menuform.ModCua.Checked = False
moni.Caption = "Monitor de Perámetros - Módulo 1"


End Sub
```

MONI.FRM - 3

```
Sub M2_Click ()
M1.Checked = False
Menuform.ModUno.Checked = False
M2.Checked = True
Menuform.ModDos.Checked = True
M3.Checked = False
Menuform.ModTres.Checked = False
M4.Checked = False
Menuform.ModCua.Checked = False
moni.Caption = "Monitor de Parámetros - Módulo 2"


End Sub

Sub M3_Click ()
M1.Checked = False
M2.Checked = False
M3.Checked = True
M4.Checked = False
moni.Caption = "Monitor de Parámetros - Módulo 3"


End Sub

Sub M4_Click ()
M1.Checked = False
M2.Checked = False
M3.Checked = False
M4.Checked = True

moni.Caption = "Monitor de Parámetros - Módulo 4"
End Sub

Sub Timer1_Timer ()
 On Error Resume Next
 If Menuform.ModUno.Checked = True Then
 M1.Checked = True
 M2.Checked = False
 M3.Checked = False
 M4.Checked = False
 a = 1
 End If
 If Menuform.ModDos.Checked = True Then
 M1.Checked = False
 M2.Checked = True
 M3.Checked = False
 M4.Checked = False
 a = 2
 End If
 If Menuform.ModTres.Checked = True Then
 M3.Checked = True
 a = 3
 End If
 If Menuform.ModCua.Checked = True Then
 M4.Checked = True
 a = 4
 End If
donde$ = "c:\moni.tmp"

        Open donde$ For Input As #1
        Do While Not EOF(1)
        ModAdhoc = 0
        For i = 0 To 4
           Input #1, datos
           If i = 0 And datos = a Then ModAdhoc = 1
           If i = 1 Then ParAdhoc = datos
           If i >= 4 Then
               If ModAdhoc = 1 Then
                If ParAdhoc = 1 Then Gauge1.Value = datos
                If ParAdhoc = 2 Then Gauge2.Value = datos
                If ParAdhoc = 3 Then Gauge3.Value = datos
                If ParAdhoc = 4 Then Gauge4.Value = datos
                If ParAdhoc = 5 Then Gauge5.Tag = datos
               Else
```

```
                Gauge1.Value = 0
                Gauge2.Value = 0
                Gauge3.Value = 0
                Gauge4.Value = 0
                Gauge5.Tag = 0
                End If
            End If
        Next i
        Loop
        Close #1

Gauge5.Value = 50
Label1.Caption = Gauge1.Value
Label4.Caption = Gauge2.Value
Label8.Caption = Gauge4.Value
Label10.Caption = Gauge5.Value
If Gauge3.Value > 50 Then
    Gauge3.ForeColor = &HC0&
    Label6.Caption = "Hay alguien"
Else
    Gauge3.ForeColor = &HE0FFFF
    Label6.Caption = "No hay nadie"
End If

If Gauge5.Tag >= 50 And Gauge5.Tag < 75 Then
    Gauge5.ForeColor = &HE0FFFF
    Label10.Caption = "Regular"
ElseIf Gauge5.Tag > 24 And Gauge5.Tag < 50 Then
    Label10.Caption = "Mala"
    Gauge5.ForeColor = &H40C0&
ElseIf Gauge5.Tag < 25 Then
Label10.Caption = "Pésima"
    Gauge5.ForeColor = &H404040
Else
    Label10.Caption = "Buena"
    Gauge5.ForeColor = &HC0FFC0
End If

End Sub
```

```
RESFORM.FRM - 1

VERSION 2.00
Begin Form ResForm
   BackColor       =   &H00808000&
   Caption         =   "Resultados"
   ClientHeight    =   6240
   ClientLeft      =   192
   ClientTop       =   648
   ClientWidth     =   8916
   Height          =   6564
   Left            =   144
   LinkTopic       =   "Form1"
   ScaleHeight     =   6240
   ScaleWidth      =   8916
   Top             =   372
   Width           =   9012
   Begin CommandButton Command3
      Caption         =   "Actualizar"
      Height          =   372
      Left            =   5040
      TabIndex        =   18
      Top             =   4560
      Width           =   1092
   End
   Begin PictureBox Picture3
      Height          =   288
      Left            =   516
      Picture         =   (Bitmap)
      ScaleHeight     =   264
      ScaleWidth      =   288
      TabIndex        =   17
      Top             =   4656
      Visible         =   0   'False
      Width           =   312
   End
   Begin PictureBox Picture2
      Height          =   288
      Left            =   504
      Picture         =   (Bitmap)
      ScaleHeight     =   264
      ScaleWidth      =   276
      TabIndex        =   16
      Top             =   4644
      Visible         =   0   'False
      Width           =   300
   End
   Begin CommandButton Cambio
      Caption         =   "Command3"
      Height          =   204
      Left            =   576
      TabIndex        =   15
      Top             =   4680
      Width           =   168
   End
   Begin PictureBox Picture1
      Height          =   288
      Left            =   156
      Picture         =   (Bitmap)
      ScaleHeight     =   264
      ScaleWidth      =   288
      TabIndex        =   14
      Top             =   4644
      Width           =   312
   End
   Begin SpinButton Spin2
      BackColor       =   &H00C0C0C0&
      ForeColor       =   &H00C0C0C0&
      Height          =   372
      Left            =   4212
      LightColor      =   &H00808080&
      ShadeColor      =   &H00C0C0C0&
      ShadowBackColor =   &H00C0C0C0&
      ShadowForeColor =   &H00C0C0C0&
      SpinForeColor   =   &H00404040&
      TdThickness     =   4
      Top             =   3900
```

RESFORM.FRM - 1

```
VERSION 2.00
Begin Form ResForm
   BackColor       =   &H00808000&
   Caption         =   "Resultados"
   ClientHeight    =   6240
   ClientLeft      =   192
   ClientTop       =   648
   ClientWidth     =   8916
   Height          =   6564
   Left            =   144
   LinkTopic       =   "Form1"
   ScaleHeight     =   6240
   ScaleWidth      =   8916
   Top             =   372
   Width           =   9012
   Begin CommandButton Command3
      Caption         =   "Actualizar"
      Height          =   372
      Left            =   5040
      TabIndex        =   18
      Top             =   4560
      Width           =   1092
   End
   Begin PictureBox Picture3
      Height          =   288
      Left            =   516
      Picture         =   (Bitmap)
      ScaleHeight     =   264
      ScaleWidth      =   288
      TabIndex        =   17
      Top             =   4656
      Visible         =   0   'False
      Width           =   312
   End
   Begin PictureBox Picture2
      Height          =   288
      Left            =   504
      Picture         =   (Bitmap)
      ScaleHeight     =   264
      ScaleWidth      =   276
      TabIndex        =   16
      Top             =   4644
      Visible         =   0   'False
      Width           =   300
   End
   Begin CommandButton Cambio
      Caption         =   "Command3"
      Height          =   204
      Left            =   576
      TabIndex        =   15
      Top             =   4680
      Width           =   168
   End
   Begin PictureBox Picture1
      Height          =   288
      Left            =   156
      Picture         =   (Bitmap)
      ScaleHeight     =   264
      ScaleWidth      =   288
      TabIndex        =   14
      Top             =   4644
      Width           =   312
   End
   Begin SpinButton Spin2
      BackColor       =   &H00C0C0C0&
      ForeColor       =   &H00C0C0C0&
      Height          =   372
      Left            =   4212
      LightColor      =   &H00808080&
      ShadeColor      =   &H00C0C0C0&
      ShadowBackColor =   &H00C0C0C0&
      ShadowForeColor =   &H00C0C0C0&
      SpinForeColor   =   &H00404040&
      TdThickness     =   4
      Top             =   3900
```

```
                Gauge1.Value = 0
                Gauge2.Value = 0
                Gauge3.Value = 0
                Gauge4.Value = 0
                Gauge5.Tag = 0
                End If
            End If
        Next i
        Loop
        Close #1

Gauge5.Value = 50
Label1.Caption = Gauge1.Value
Label4.Caption = Gauge2.Value
Label8.Caption = Gauge4.Value
Label10.Caption = Gauge5.Value
If Gauge3.Value > 50 Then
    Gauge3.ForeColor = &HC0&
    Label6.Caption = "Hay alguien"
Else
    Gauge3.ForeColor = &HE0FFFF
    Label6.Caption = "No hay nadie"
End If

If Gauge5.Tag >= 50 And Gauge5.Tag < 75 Then
    Gauge5.ForeColor = &HE0FFFF
    Label10.Caption = "Regular"
ElseIf Gauge5.Tag > 24 And Gauge5.Tag < 50 Then
    Label10.Caption = "Mala"
    Gauge5.ForeColor = &H40C0&
ElseIf Gauge5.Tag < 25 Then
Label10.Caption = "Pésima"
    Gauge5.ForeColor = &H404040
Else
    Label10.Caption = "Buena"
    Gauge5.ForeColor = &HC0FFC0
End If

End Sub
```

```
      Width           =   252
   End
   Begin SpinButton Spin1
      BackColor       =   &H00C0C0C0&
      ForeColor       =   &H00C0C0C0&
      Height          =   372
      Left            =   5370
      LightColor      =   &H00808080&
      ShadeColor      =   &H00C0C0C0&
      ShadowBackColor =   &H00C0C0C0&
      ShadowForeColor =   &H00C0C0C0&
      SpinForeColor   =   &H00404040&
      TdThickness     =   4
      Top             =   1800
      Width           =   252
   End
   Begin CommandButton Command2
      Caption         =   "Imprimir"
      Height          =   228
      Left            =   228
      TabIndex        =   7
      Top             =   4668
      Width           =   144
   End
   Begin GRAPH Graph1

      AsciiData       =   "0~0~0~0~0~0~1"




      AsciiSymbol     =   "1"
      AsciiXPos       =   "6"
      AutoInc         =   0   'Off
      Background      =   7   'Light Gray




      GraphType       =   9   'Scatter
      Height          =   3012
      LabelEvery      =   3
      Labels          =   3   'Y Labels only

      Left            =   132

      NumPoints       =   24

      RandomData      =   0   'Off

      TabIndex        =   0
      ThickLines      =   0   'Off
      TickEvery       =   3
      Ticks           =   3   'Y Ticks only
      Top             =   240
      Visible         =   0   'False
      Width           =   4812

   End
   Begin CommandButton Command1
      Caption         =   "Aceptar"
      Height          =   372
      Left            =   6360
      TabIndex        =   4
      Top             =   4548
      Width           =   972
   End
   Begin Grid Grid1
      BackColor       =   &H00FFFFFF&
      Cols            =   3
```

```
      FixedCols       =   0
      ForeColor       =   &H00000000&
      Height          =   3972
      Left            =   360
      Rows            =   5
      ScrollBars      =   2  'Vertical
      TabIndex        =   3
      Top             =   480
      Visible         =   0   'False
      Width           =   2532
   End
   Begin Frame pic
      BackColor       =   &H00C0C0C0&
      ForeColor       =   &H80C0C0C0&
      Height          =   1548
      Left            =   5640
      TabIndex        =   8
      Top             =   144
      Visible         =   0   'False
      Width           =   1692
      Begin PictureBox nube
         BackColor       =   &H00808000&
         BorderStyle     =   0  'None
         Height          =   1308
         Left            =   108
         Picture         =   (Bitmap)
         ScaleHeight     =   1308
         ScaleWidth      =   1368
         TabIndex        =   9
         Top             =   216
         Visible         =   0   'False
         Width           =   1368
      End
      Begin PictureBox Imecas
         BackColor       =   &H00C0C0C0&
         BorderStyle     =   0  'None
         FillColor       =   &H00E0E0E0&
         Height          =   732
         Left            =   156
         Picture         =   (Bitmap)
         ScaleHeight     =   732
         ScaleWidth      =   1356
         TabIndex        =   13
         Top             =   516
         Visible         =   0   'False
         Width           =   1356
      End
      Begin PictureBox pres
         BackColor       =   &H00C0C0C0&
         BorderStyle     =   0  'None
         Height          =   1368
         Left            =   216
         Picture         =   (Bitmap)
         ScaleHeight     =   1368
         ScaleWidth      =   1356
         TabIndex        =   11
         Top             =   132
         Visible         =   0   'False
         Width           =   1356
      End
      Begin PictureBox Ilum
         BackColor       =   &H00C0C0C0&
         BorderStyle     =   0  'None
         Height          =   1224
         Left            =   408
         Picture         =   (Bitmap)
         ScaleHeight     =   1224
         ScaleWidth      =   852
         TabIndex        =   12
         Top             =   288
         Visible         =   0   'False
         Width           =   852
      End
      Begin PictureBox hot
         BackColor       =   &H00808000&
```

```
         BorderStyle     =   0  'None
         Height          =   1428
         Left            =   156
         Picture         =   (Bitmap)
         ScaleHeight     =   1428
         ScaleWidth      =   1440
         TabIndex        =   10
         Top             =   96
         Visible         =   0   'False
         Width           =   1440
      End
   End
   Begin Label Label5
      Alignment       =   2  'Center
      BackColor       =   &H008080000&
      Caption         =   "Label5"
      ForeColor       =   &H00C0C0C0&
      Height          =   1092
      Left            =   5628
      TabIndex        =   6
      Top             =   2760
      Width           =   1092
   End
   Begin Label Label3
      BackColor       =   &H008080000&
      Caption         =   "Módulo"
      FontBold        =   -1  'True
      FontItalic      =   0   'False
      FontName        =   "Arial"
      FontSize        =   21
      FontStrikethru  =   0   'False
      FontUnderline   =   0   'False
      ForeColor       =   &H00C0C0C0&
      Height          =   492
      Left            =   3120
      TabIndex        =   5
      Top             =   3360
      Width           =   1572
   End
   Begin Label Label2
      BackColor       =   &H008080000&
      Caption         =   "Label2"
      FontBold        =   -1  'True
      FontItalic      =   0   'False
      FontName        =   "MS Sans Serif"
      FontSize        =   12
      FontStrikethru  =   0   'False
      FontUnderline   =   0   'False
      ForeColor       =   &H00C0C0C0&
      Height          =   735
      Left            =   5640
      TabIndex        =   2
      Top             =   1800
      Visible         =   0   'False
      Width           =   2415
   End
   Begin Label Label1
      Alignment       =   2  'Center
      BackColor       =   &H008080000&
      Caption         =   "Label1"
      FontBold        =   -1  'True
      FontItalic      =   0   'False
      FontName        =   "Arial"
      FontSize        =   21
      FontStrikethru  =   0   'False
      FontUnderline   =   0   'False
      ForeColor       =   &H00C0C0C0&
      Height          =   612
      Left            =   3216
      TabIndex        =   1
      Top             =   3804
      Visible         =   0   'False
      Width           =   1332
   End
End
```

RESFORM.FRM - 1

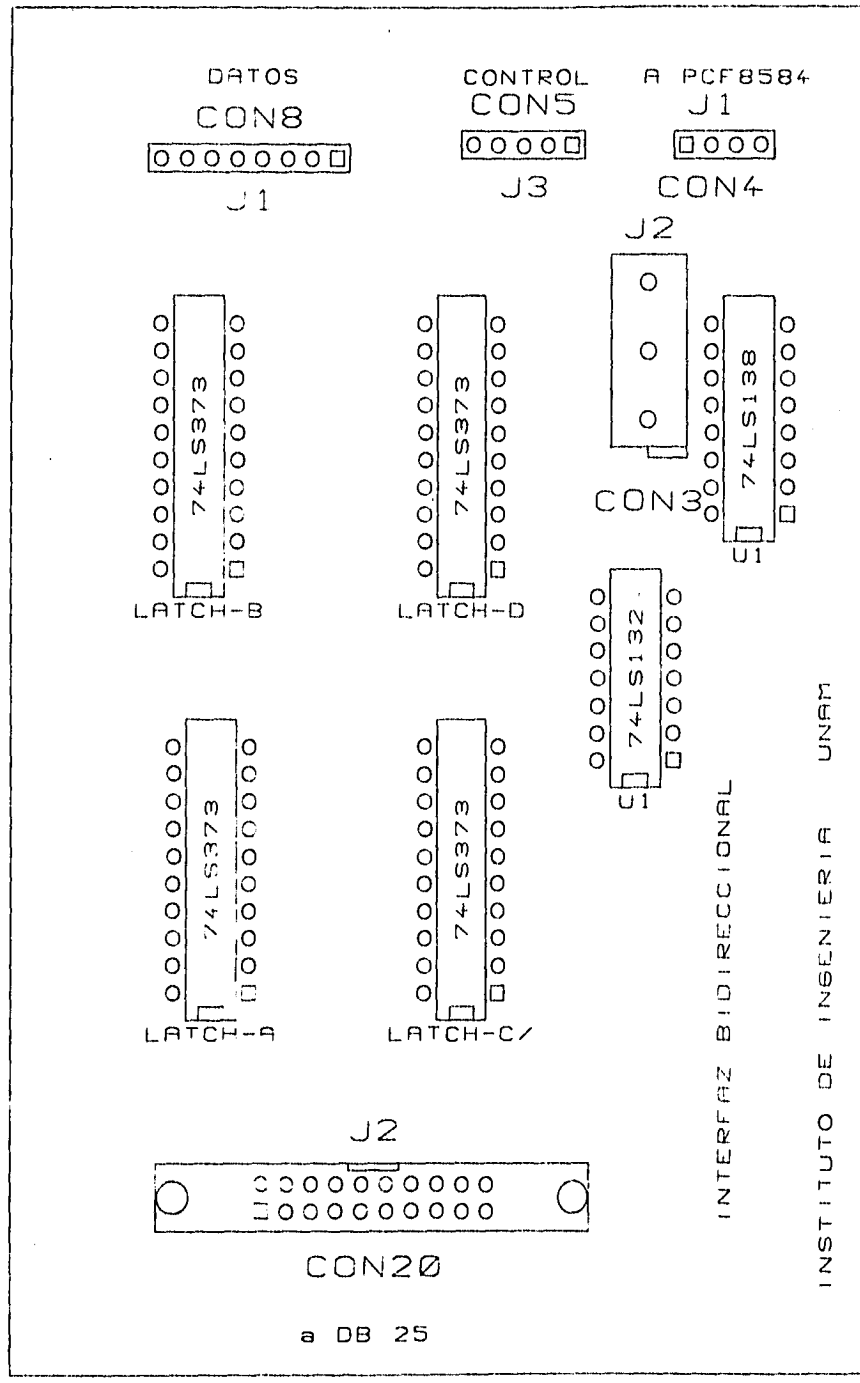
```
Sub CAmbio_Click ()
If Graph1.Visible = True Then
MenuForm.Tag = "G2R"
Picture2.Visible = True
Picture3.Visible = False
End If
If Grid1.Visible = True Then
Picture3.Visible = True
Picture2.Visible = False
MenuForm.Tag = "R2G"
End If
Command1_Click
End Sub

Sub Command1_Click ()
MenuForm.Show
MenuForm.Enabled = True
ResForm.Hide
Graph1.Visible = False
Grid1.Visible = False
pic.Visible = False
Nube.Visible = False
Hot.Visible = False
Ilum.Visible = False
Pres.Visible = False
Imecas.Visible = False
End Sub

Sub Command2_Click ()
If Grid1.Visible = True Then
 Printer.FontSize = 16
 Printer.Print Label3.Caption; "  "; label1.Caption
 Printer.Print
 Printer.Print Label2.Caption
 Printer.Print Label5
 Printer.Print

 Printer.FontSize = 12
 For i = 0 To Grid1.Rows - 1
    For j = 0 To Grid1.Cols - 1
    Grid1.Col = j
    Grid1.Row = i
    Printer.Print Grid1.Text;
    Printer.Print "     ";
    Next j
    Printer.Print
 Next i


   Printer.EndDoc

ElseIf Graph1.Visible = True Then
Graph1.BottomTitle = Label2.Caption + "  -Módulo " + label1.Caption

Graph1.DrawMode = 5
Graph1.BottomTitle = ""
Graph1.DrawMode = 2
End If
End Sub

Sub Command3_Click ()
If Dir$("c:\tmp.tmp") <> "" Then FileCopy "c:\tmp.tmp", "c:\j&j1.tmp"
If Graph1.Visible = True Then MenuForm.Tag = "REFG"
If Grid1.Visible = True Then MenuForm.Tag = "REFR"
Command1_Click
End Sub

Sub Form_Unload (Cancel As Integer)
MenuForm.Show
MenuForm.Enabled = True
ResForm.Hide
Graph1.Visible = False
Grid1.Visible = False
```

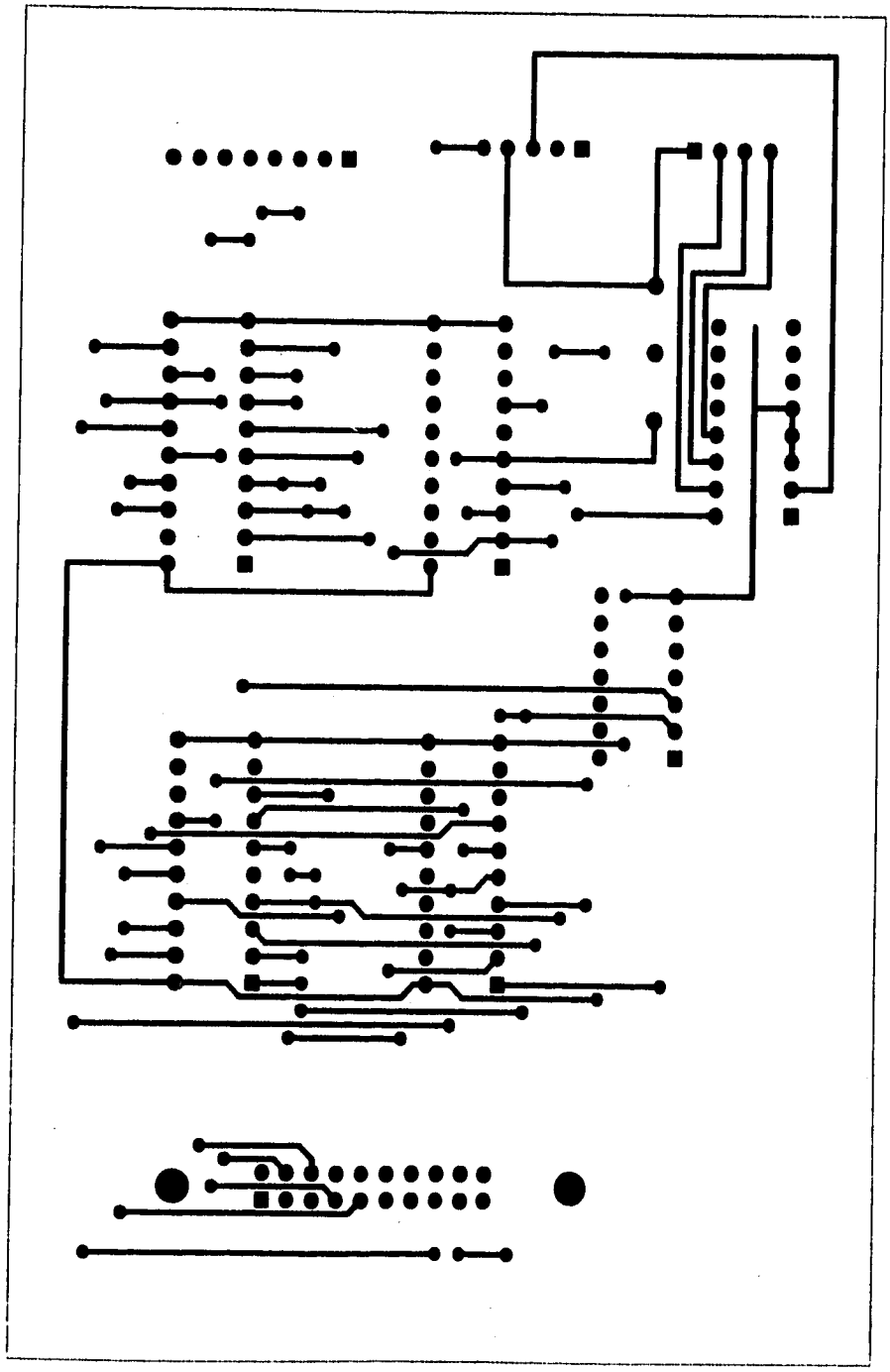
```
RESFORM.FRM - 2

pic.Visible = False
Nube.Visible = False
Hot.Visible = False
Ilum.Visible = False
Pres.Visible = False
Imecas.Visible = False
MenuForm.Grafica.Tag = 0

End Sub

Sub Picture1_Click ()
Command2_Click
End Sub

Sub Picture2_Click ()
CAmbio_Click
End Sub

Sub Picture3_Click ()
CAmbio_Click
End Sub

Sub Spin1_SpinDown ()
If Graph1.Visible = True Then MenuForm.Tag = "GPD"
If Grid1.Visible = True Then MenuForm.Tag = "RPD"
Command1_Click
End Sub

Sub Spin1_SpinUp ()
If Graph1.Visible = True Then MenuForm.Tag = "GPU"
If Grid1.Visible = True Then MenuForm.Tag = "RPU"
Command1_Click
End Sub

Sub Spin2_SpinDown ()
If Graph1.Visible = True Then MenuForm.Tag = "GD"
If Grid1.Visible = True Then MenuForm.Tag = "RMD"
Command1_Click

End Sub

Sub Spin2_SpinUp ()
If Graph1.Visible = True Then MenuForm.Tag = "GU"
If Grid1.Visible = True Then MenuForm.Tag = "RMU"
Command1_Click
End Sub
```

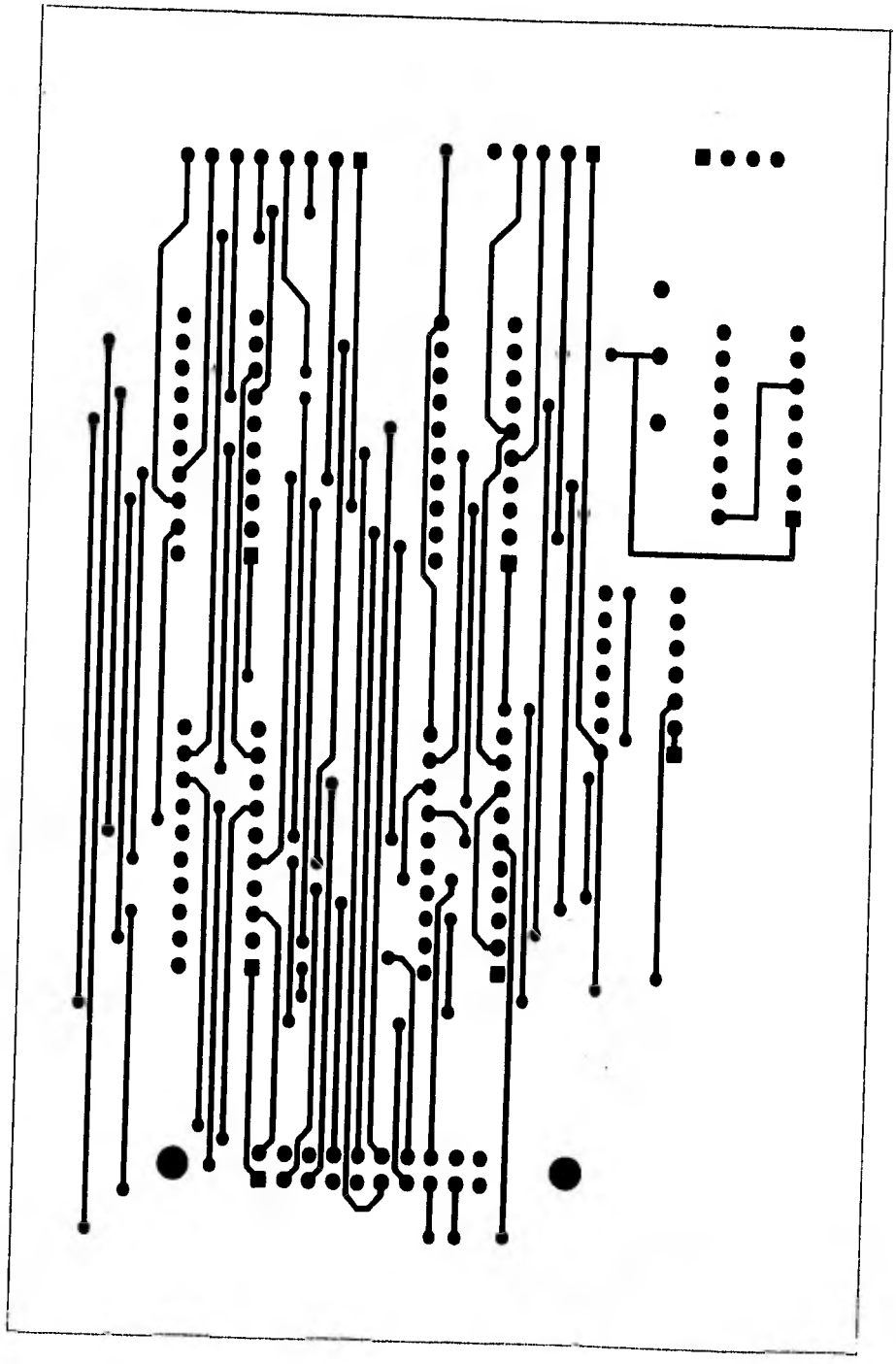
MODULE1.BAS - 1

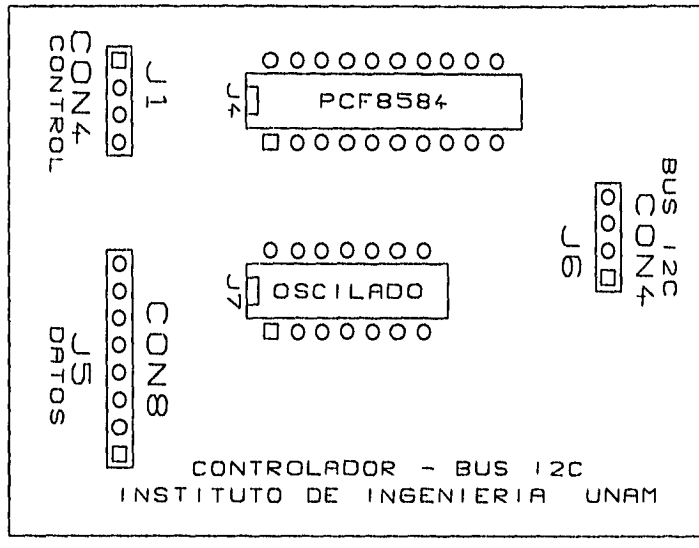
```
Global Archivo$
```

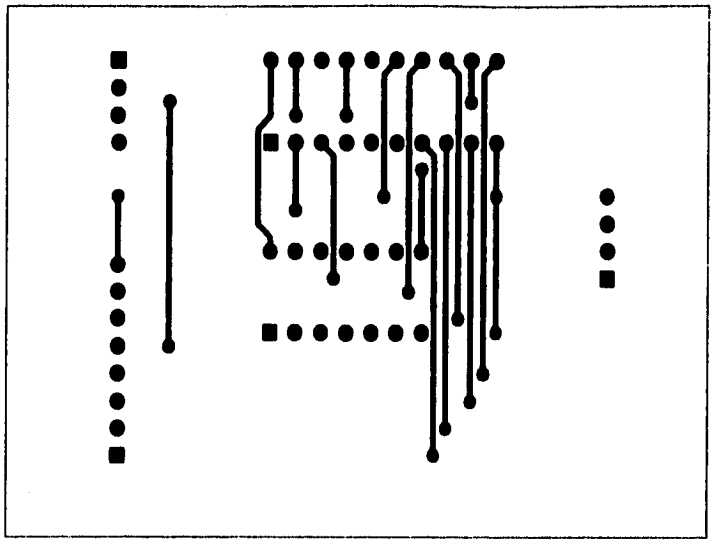
Apéndice D: Circuitos Impresos

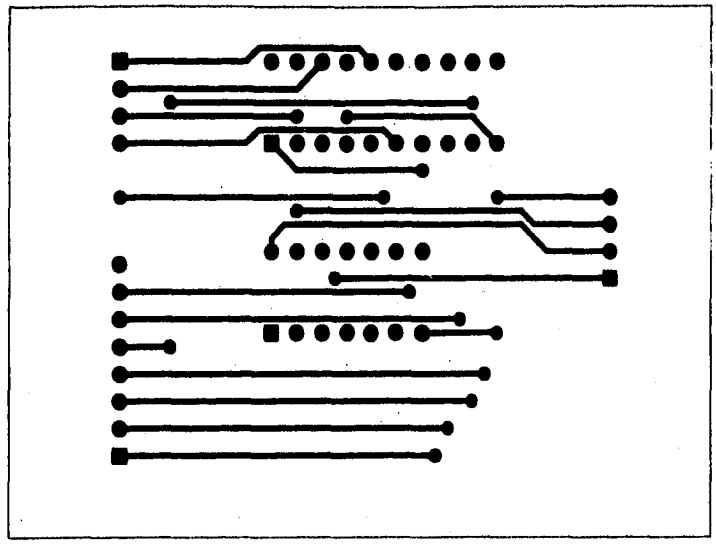

DATOS
CON8
J1
CONTROL
CON5
J3
A PCF8584
J1
CON4
J2
CON3
74LS373
LATCH-B
74LS373
LATCH-D
74LS138
U1
74LS132
U1
74LS373
LATCH-A
74LS373
LATCH-C
J2
CON20
a DB 25
INTERFAZ BIDIRECCIONAL
INSTITUTO DE INGENIERIA UNAM

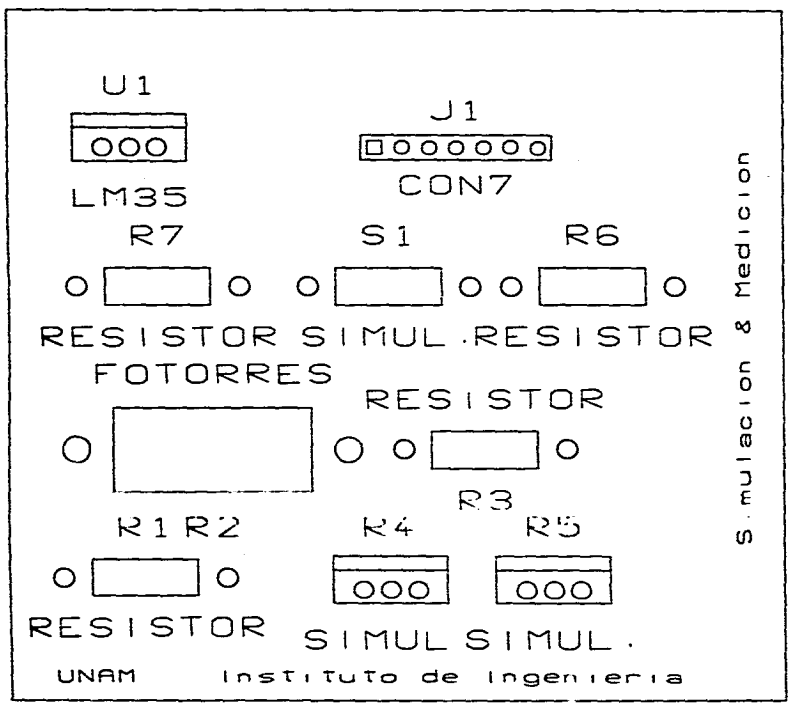





CON4
CONTROL
J1
PCF8584
J4
BUS I2C
CON4
J6
OSCILADO
J7
CON8
J5
DATOS
CONTROLADOR - BUS I2C
INSTITUTO DE INGENIERIA UNAM






U1
J1
LM35
CON7
R7
S1
R6
RESISTOR SIMUL. RESISTOR
FOTORRES
RESISTOR
R3
R1 R2
R4
R5
RESISTOR
SIMUL SIMUL.
UNAM Instituto de Ingenieria
Simulacion & Medicion