UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

MICROCOMPUTADORA DE 16 BITS PARA SISTEMAS DE MULTIPROCESAMIENTO

T E S I S :

QUE PARA OBTENER EL TITULO DE:

INGENIERO MECANICO ELECTRICISTA

P R E S E N T A N :

MIGUEL AGUILAR SIERRA

VICTOR MANUEL CASTRO GOMEZ

MEXICO, D. F. JUNIO DE 1987

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

CAPITULO I

INTRODUCCIÓN

CAPITULO I

INTRODUCCION

1.1 ANTECEDENTES

El desarrollo nacional en el área de control de procesos no se está dando tanto en la construcción de sistemas analógicos tipo servomecanismo, sino en el área de control por computadora. Hay labor intensa tanto en la industria como en las universidades con proyectos en esta área.

La tendencia actual de desarrollo es hacia el control distribuido utilizando sistemas de multiprocesamiento comunicados por una red, con cierto grado de redundancia y que puedan cambiar, adoptar o heredar funciones ajenas en circunstancias criticas [4].

En la actualidad la tecnología VLSI se ha desarrollado a tal grado que es posible tener una potente microcomputadora en un solo chip, gran capacidad de memoria en pocos componentes y chips de soporte que facilitan en gran medida la interfaz del

procesador con periféricos, dispositivos aledaños y con el mundo real en general. Parece mucho más factible desarrollar sistemas de control en una serie de microcomputadoras diseñadas exprofeso para tal efecto, interconectarlas entre sí e incrementar el número de éstas según los requerimientos, que usar una sola computadora de proposito general, por muy grande que ésta sea.

Con estas ventajas, día con día se están desarrollando en el mercado sistemas con una gran versatilidad, en donde se pueden tener controlados diversos procesos a la vez, ofreciendo confiabilidad en el sistema, en adición a velocidad de respuesta.

Con esta preocupación, se desarrolló el presente trabajo con el objeto de diseñar un sistema que además de contar con las características antes mencionadas, tuviera sus bases sentadas en arquitecturas comerciales, con la finalidad de ofrecer la ventaja de contar con el apoyo de diversas compañías comerciales, y así tener la posibilidad real de crecimiento del sistema.

1.2 CARACTERISTICAS REQUERIDAS PARA EL SISTEMA

A continuación se describen las principales características que debe reunir el sistema y se presenta la aplicación del mismo a sistemas de control de procesos.

1.2.1 TIEMPO REAL

Este término se refiere al tiempo que tarda en responder un sistema a un evento externo [3].

Algunos autores consideran de tiempo real a aquéllos sistemas que proporcionan respuestas comparables con los tiempos

de reacción humana, sin embargo, durante el control de un proceso se pueden presentar eventos críticos síncronos (si se sabe cuándo se van a presentar) o asíncronos (impredecibles), que deben atenderse en el orden de microsegundos en algunos casos (cortes de alimentación de energía eléctrica del propio sistema) o milisegundos en otros (al sobrepasar los límites de temperatura permisibles), lo que ocasiona que los sistemas deban ser capaces de reaccionar a una velocidad mucho mayor de la que podría realizar el hombre. Esta velocidad en la atención de eventos críticos se ve seriamente amenazada cuando en determinado momento se presentan una gran cantidad de alarmas o eventos críticos que atender. En tal caso el sistema deberá organizar una cola de eventos críticos siguiendo una cierta estructura de prioridades y atenderá a los mismos de acuerdo con dicha cola. El resultado es que algunos eventos serán atendidos no en tiempo real sino de acuerdo con la demanda de atención del sistema.

Un nuevo concepto en la literatura de los sistemas de control es lo que se conoce como "Sistemas de estricto tiempo real"[3], es decir, son sistemas en los que no importando las condiciones en las cuales se presentan los eventos críticos, estos serán atendidos en un cierto tiempo garantizado en todo momento. Cada evento tiene su tiempo garantizable particular, por lo que es responsabilidad del diseñador organizar la atención de los eventos en función a la velocidad de operación del sistema de tal manera que no importando cual sea el orden y circunstancias en que se presenten estos eventos todos serán atendidos dentro de su respectivo tiempo garantizado.

El término de tiempo real es usado para distinguír los sistemas mencionados de los sistemas de procesamiento de datos en los que los tiempos de respuesta no son el punto central.

Una de las características del sistema que se propone es atender a los eventos de la misma manera que lo llevan a

efecto los sistemas de estricto tiempo real.

1.2.2 TOLERANCIA A FALLAS

En la actualidad existen sistemas de cómputo complejos, que tienen a su cargo la responsabilidad de controlar procesos delicados, en los cuales es de suma importancia no detener la ejecución de los mismos a la mitad. Los procesos de producción en línea no deben detenerse durante la operación normal ya que se pierden insumos a medio procesar, sincronía y tiempo. Reiniciar la operación normal puede ser muy complejo y tardado. Asimismo no se puede detener la operación de sistemas de adquisición de datos porque los eventos que se esperan pueden presentarse en cualquier momento. Por estos motivos los equipos de cómputo que gobiernan estos sitemas no deben fallar nunca, por el contrario deben garantizar, en lo posible, un correcto funcionamiento en forma ininterrumpida.

La habilidad de un sistema de operar correctamente en presencia de fallas es lo que se llama tolerancia a fallas [1].

1.2.2.1 Definición de falla

Una falla (de hardware o software) se define como cualquier estado erróneo de un sistema.

1.2.2.2 Causas de las fallas

Clasificaremos las causas de las fallas en 4 niveles [2]:

Las de más alto nivel son las debidas a errores de especificación, que contienen errores en algoritmos y/o en la arquitectura, y los más frecuentes, que son los errores de diseño

de hardware y software.

En el siguiente nivel estan los errores de implementación, entendiendo como implementación el proceso de la transformación de las especificaciones del hardware y software en el modelo físico. Estos errores son frecuentemente introducidos por un mal diseño, o por errores en la codificación del software.

La siguiente causa de fallas son las fallas en los componentes. Errores o defectos de manufactura, fallas en los dispositivos semiconductores, o componentes que se queman, son ejemplos típicos. Las fallas en los componentes son quizá las más frecuentemente consideradas como la causa de fallas en sistemas digitales.

La última causa de fallas esta categorizada como disturbios externos, como radiación, interferencia electromagnética, y condiciones ambientales externas.

En la figura 1.1 se ilustran los conceptos anteriores.

1.2.2.3 Técnicas para el tratamiento de fallas

Existen tres técnicas para intentar mejorar o mantener el funcionamiento normal de un sistema: Evitar la falla, enmascarar la falla y tolerar la falla.

Evitar la falla es la prevención de las causas de fallas, por lo que se debe llevar a cabo un estricto mantenimiento, ya sea del tipo preventivo o predictivo.

Enmascarar la falla es cualquier proceso de tiempo real que previene la propagación de las fallas de un sistema en los datos del sistema.

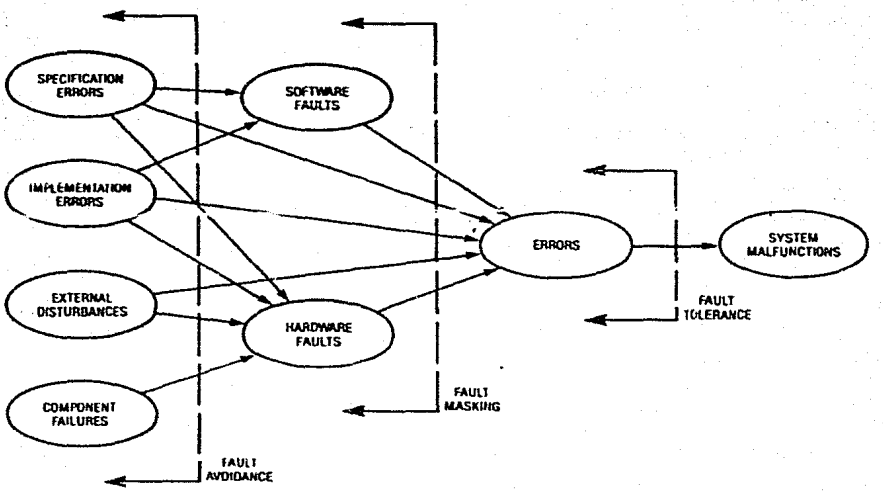


Figura 1.1 Causas de las fallas.

Tolerar la falla es la habilidad de un sistema para continuar ejecutando sus tareas especificadas después de que ha ocurrido un error. En este trabajo se considera de interés únicamente esta opción.

1.2.2.4 Técnicas de tolerancia de fallas

La tolerancia de fallas involucra cuatro aspectos: detección de fallas, localización de fallas, contención de fallas, y recuperación de fallas. La detección de fallas es la habilidad de un sistema para reconocer que una falla ha ocurrido. La detección de fallas es frecuentemente requerida antes de que cualquier mecanismo de recuperación pueda ser implementado. Localización de

fallas en un sistema es la habilidad para determinar en donde ha ocurrido la falla, para que una recuperación apropiada sea llevada a efecto. Contención de fallas es el proceso de aislar una falla, previniendo que sus efectos se propaguen a través del sistema. Finalmente recuperación de fallas es la habilidad de un sistema para permanecer operando, o de recobrar el estado de operación por medio de la reconfiguración, aún en presencia de fallas.

La clave en todas las técnicas de tolerancia de fallas es la redundancia. La redundancia es simplemente la adición de información, recursos, o tiempo para la operación normal del sistema.

La redundancia puede tomar diversas formas: redundancia de información, redundancia de hardware, redundancia de software , y redundancia de tiempo. La redundancia de información es la adición de información más allá de la requerida para llevar a cabo una transferencia de datos. La redundancia de hardware es la réplica física de hardware con el propósito de detectar y tolerar fallas. La redundancia de software es la adición de software más allá de lo necesario para realizar una función dada, con el propósito de detectar y tolerar fallas en un sistema. Finalmente la redundancia de tiempo usa tiempo adicional para la detección de la falla y algunas veces para tolerarla.

A continuación se dará una breve explicación de la redundancia de hardware, ya que el presente trabajo, está abocado sobre todo a tolerar fallas de hardware.

1.2.2.5 Redundancia de hardware

Quizá la técnica más comunmente utilizada en sistemas tolerantes a fallas es la réplica de hardware. Debido a que los componentes semiconductores han llegado a ser menos caros, ésta técnica se ha hecho la más económica y práctica.

Existen básicamente tres formas de replicar el hardware. La primera son los métodos de réplica pasiva, que enmascaran la ocurrencia de fallas y no ofrecen detección, aislamiento o reparación del módulo que está fallando. La segunda son los métodos de réplica activa, que no enmascaran las fallas, pero detectan y localizan la falla para que un componente de repuesto pueda ser conmutado para reemplazar el componente que está fallando. La tercera son los métodos híbridos, que combinan las características de las técnicas pasiva y activas. Los métodos de réplica híbrida utilizan el enmascaramiento de la falla para prevenir que la falla afecte al sistema, y la detección de la falla para permitir que un módulo de repuesto sea conmutado para reemplazar el módulo que está fallando.

Por lo anterior, se concluye que para que un sistema sea tolerante de fallas debe tener:

Redundancia de Información.
Redundancia de Softwarwe.
Redundancia de Hardware.
Redundancia de Tiempo.

1.2.3 MULTIPROCESADORES

Antes de poder definir a un multiprocesador y establecer las diferencias entre éste y otros conceptos existentes en la literatura especializada y que se prestan a confusión, es necesario definir primero los conceptos generales que se utilizarán en el resto del trabajo.

-Tarea. Es una "entidad" de un programa que puede ejecutarse independientemente de cualquier otro programa. Es la unidad básica de software en un sistema operativo. Pueden existir tareas

de usuario o del sistema, y pueden ser suspendidas por una interrupcion de hardware para ser continuadas después.

-**Trabajo.** Tarea o grupo de tareas a ser realizadas por un sistema de cómputo.

-**Sistema multitarea.** Sistema en el que dos o más tareas pueden estar "activadas" en cualquier momento dado.

-**Proceso.** Este es un concepto muy general que debe ser usado con cuidado. El concepto "proceso" se utiliza para describir una actividad que provee una función determinada para el usuario o para el sistema. En otras palabras, se puede pensar en un proceso como una secuencia de acciones, realizadas ejecutando una secuencia de instrucciones (un programa), cuyo resultado neto es proveer de alguna función al sistema o al usuario. Debe notarse que en esta definición no se asume nada sobre el nivel al que se implementa dicha función, es decir, el programa o programas asociados con un proceso no tienen que estar implementados como software; el acceso a memoria, la transferencia de datos en un canal serie o el acceso a un periférico pueden también considerarse procesos cuyos programas están alambrados en el hardware del sistema.

-**Simultáneo.** Ocurrencia de dos o más eventos en el mismo instante.

-**Concurrente.** Ocurrencia de dos o más eventos dentro de un mismo intervalo de tiempo. La concurrencia puede entenderse como el estado activo de varios procesos (por ejemplo, la ejecución de varios programas) en un mismo lapso de tiempo.

-**Procesamiento en paralelo (o concurrente).** Básicamente, éste concepto quiere decir que en un instante dado, tomando al sistema como un todo, varios procesos pueden encontrarse en algún lugar

entre sus puntos de inicio y sus puntos de término. Cuando existen tantos procesadores como procesos en ejecución, esto no presenta ninguna dificultad lógica y se tendrá paralelismo (o concurrencia) verdadero, es decir, la ejecución simultánea de procesos. Cuando no es así, se tiene paralelismo (o concurrencia) aparente, que es lograda conmutando el procesador único de un proceso a otro.

-Multiprocesamiento. Se refiere a la ejecución simultánea de procesos en diferentes procesadores de un sistema de cómputo [6]. En otras palabras, el multiprocesamiento consiste en paralelismo verdadero. El concepto de multiprocesamiento ha sido presentado en diversas formas como: sistemas de multiprocesadores, sistemas de cómputo distribuido, sistemas de procesador múltiple, multicomputadoras y redes de computadoras. La diferencia entre estos conceptos se explicará en el siguiente capítulo, al describir las arquitecturas de los sistemas de cómputo. Por el momento, solo nos ocuparemos de los multiprocesadores, pues es esta arquitectura el tema fundamental del presente trabajo.

-Multiprocesadores. Durante unos años, los multiprocesadores se han llevado a la práctica principalmente en los llamados procesadores de arreglos en los que varias unidades de procesamiento ejecutan simultáneamente la misma función, y en procesadores "pipeline" en los que una tarea se subdivide en una secuencia de subtareas que se ejecutan concurrentemente. Posteriormente, el aumento de aplicaciones en tiempo real, tanto militares como industriales y comerciales, que demandan sistemas que no fallen o si lo hacen, pierdan cuando mucho una transacción recuperable, ha provocado que los investigadores diseñen e implementen mejores sistemas de cómputo, buscando arquitecturas de multiprocesadores innovadoras y sofisticadas como solución a dicha demanda.

1.2.3.1 Definición de un Multiprocesador

El multiprocesador es un sistema que tiene las siguientes características [7]:

1.- Contiene dos ó más procesadores de aproximadamente la misma capacidad.
2.- Todos los procesadores comparten acceso a una memoria común.
3.- Todos los procesadores comparten acceso a canales de E/S, unidades de control y otros dispositivos.
4.- El sistema está controlado por un solo sistema operativo que permite la interacción entre los procesadores y sus programas en los niveles de trabajo, tarea, transferencia y manejo de datos.

Esta lista de características será fundamental a lo largo de todo el trabajo para evitar confusiones y poder distinguir entre la gran variedad de arquitecturas que existen a la fecha, por lo que se hará referencia a la misma cada vez que sea necesario.

1.2.3.2 Características de los procesadores para un multiprocesador.

A continuación se presentan algunas características necesarias que deben tener los procesadores que pretendan utilizarse en un sistema de multiprocesador.

-**Recuperación de falla.** Si un procesador falla, deberá ser posible para algún otro procesador recuperar la información del proceso interrumpido de manera que su ejecución pueda ser continuada. Para esto es necesario que los registros internos del procesador donde se guarda el estado del proceso en ejecución, sean enviados a memoria al presentarse una falla. También es

importante que el procesador pueda manejar códigos de operación no reconocidos, esto es, que no pertenezcan a su juego de instrucciones.

-Espacio de direcciones grande, virtual y físico. Las aplicaciones para las cuales se enfocan los sistemas de multiprocesamiento requieren del manejo de extensos volúmenes de información. Aunque los algoritmos se puedan implementar con un mínimo de instrucciones, en medios ambientes de multiprocesamiento es necesario almacenar muchos datos sobre los procesos que se estén ejecutando para un análisis posterior con fines estadísticos ó para mejorar el rendimiento de los procesos. El manejo de gráficas, muy común hoy en día, requiere también dé un espacio amplio de memoria, sobre todo si se desea una buena resolución. Por estas razones, el procesador que se pretenda utilizar en la construccion de un multiprocesador deberá ser capaz de soportar un gran espacio físico de direccionamiento. También es deseable que se tenga espacio de direccionamiento virtual que pueda segmentarse de manera que permita compartir modularmente la memoria y facilite la supervisión de los rangos de direcciones para la protección de dicha memoria.

-Mecanismos para la comunicación entre procesadores. Los procesadores utilizados en un multiprocesador deberán tener medios eficientes para su intercomunicación. Estos medios deben estar implementados a nivel de hardware para facilitar la sincronización entre los procesos. Dicho mecanismo puede ser utilizado cuando se presenta una falla en un procesador para emitir una señal (por hardware) a los demás procesadores en el canal de comunicaciones. De esta manera algún otro procesador que se percate de la falla podrá iniciar una rutina de sustitución ó un procedimiento de diagnóstico.

-Conjunto de instrucciones. El juego de instrucciones del procesador deberá tener las facilidades para implementar

lenguajes de alto nivel y que permitan manejar eficientemente estructuras de datos. Deberá incluir instrucciones que soporten el ligado de procedimientos, manipulaciones de parámetros, instrucciones anidadas y el chequeo de rangos de direcciones. De las características más importantes es la existencia de instrucciones que faciliten la ejecucución de procesos paralelos ó concurrentes, lo cual implica instrucciones de lectura-modifica-escritura en un ciclo indivisible que permite el manejo de semáforos en secciones críticas de procesos concurrentes (esto se detallará en el cap. 4).

1.2.4 CONTROL DE PROCESOS

Todos los procesos industriales son afectados por perturbaciones externas y deben ser controlados para mantener la calidad del producto dentro de límites aceptables [4]. La complejidad del sistema de control que se utilice para este fin depende de la complejidad del proceso.

Las computadoras digitales han sido aplicadas al control de procesos desde fines de los cincuentas, y desde entonces, la tecnología de estas aplicaciones se ha desarrollado rápidamente, tanto en hardware como en software.

Las computadoras se usan de muchas maneras diferentes para propósitos de control de procesos. Al nivel más bajo, se utilizan para control digital directo (DDC), es decir, la sustitución de controladores analógicos convencionales. También se aplican en control supervisorio para calcular los puntos óptimos de operación, y en control jerárquico ó distribuido horizontal. Los sofisticados sistemas de control adaptable por computadora intentan optimizar el funcionamiento global de todo el proceso. Procedimientos de encendido óptimo por computadora son

implementados para minimizar el tiempo en el que el proceso alcanza el estado estable. De manera similar, la computadora se usa para apagar una planta de una manera segura.

1.2.4.1 Sistemas de Control Distribuido

La mayoría de los sistemas de control implementados durante los sesentas y setentas eran centralizados, es decir, eran sistemas en los que una sola computadora grande se usaba para adquirir datos de muchos procesos y controlar un gran número de lazos de control. El costo de estos sistemas era relativamente alto, ya que las computadoras de gran capacidad y minicomputadoras grandes usadas en estos sistemas, así como sus periféricos asociados, son relativamente caros. Además existía la necesidad de largos cables blindados entre los actuadores y sensores y el cuarto de control central. El costo se eleva aún más en los casos en los que se desea un sistema de alta confiabilidad ya que es necesaria otra computadora lista para ser usada en casos de falla en la unidad principal (tolerancia a fallas). Una interrupción de la producción y el apagado completo del proceso puede ser extremadamente costoso.

La disponibilidad de poderosas micro y minicomputadoras ha hecho posible implementar sistemas de control por computadora que no requieren la infraestructura de un poderoso sistema de cómputo con costosos periféricos. Estos sistemas de proceso distribuido son adecuados tanto para control de procesos como para aplicaciones generales en administración. Así mismo, es posible combinar sistemas de control con el manejo de manufactura y sistemas de cómputo administrativos. En estos sistemas, en lugar de usar una computadora de gran capacidad para realizar muchas tareas no relacionadas, se utilizan varias mini o microcomputadoras para realizar las tareas requeridas. En el caso de control de procesos, se llega al límite cuando se usa una

microcomputadora para controlar cada lazo individual del proceso. Las microcomputadoras deberán estar ligadas entre sí o a un sistema que controle su operación en un nivel más alto. En sistemas complejos, puede haber varios niveles en los que se usen diferentes tipos de sistemas de cómputo dependiendo del tipo de función a realizar.

Los pequeños sistemas de cómputo utilizados en procesamiento distribuido son relativamente baratos, y al ligarlos entre sí, ofrecen un poder de cómputo mucho mayor al disponible de una computadora de gran capacidad que cuesta mucho más.

1.2.4.2 Sistemas de Multiprocesadores

Una forma alternativa de procesamiento distribuido es el uso de sistemas de multiprocesadores.

En algunas aplicaciones de sistemas de control de procesos y adquisición de datos en tiempo real, la velocidad de muchos sistemas individuales de microcomputación es inadecuado. Muchas aplicaciones también requieren que algunas manipulaciones de datos sean realizadas simultáneamente o en paralelo. Las computadoras digitales son dispositivos secuenciales que solo pueden ejecutar una instrucción a la vez, por lo que estos problemas se tienen que solucionar usando sistemas de multiprocesadores que comparten un bus común a distancias relativamente cortas. Cada microprocesador del sistema puede realizar simultáneamente las tareas requeridas, como por ejemplo, entrada/salida y manipulación de datos. Los resultados de las operaciones individuales pueden ser combinadas, si es necesario, para cumplir con la función total.

1.3 FACTORES DE DECISION.

Antes de iniciar el diseño del sistema, es necesario decidir el tipo de procesador que se utilizará en base a las necesidades del usuario y a los distintos factores que intervienen en el diseño.

1.3.1 MICROPROCESADORES MODERNOS

Tomando en cuenta las características mencionadas en la 2a. parte de este capítulo, los factores que intervienen para la elección del procesador son, entre otros: recursos para manejo de buses de tiempo compartido, rango de direccionamiento grande, y operación a alta velocidad para poder atender eventos en tiempo real.

Por otro lado, debe tomarse en cuenta el costo del sistema, además de la cantidad de espacio que ocuparán los módulos sobre la tarjeta, ya que ésta es de dimensiones específicas.

Con base en los aspectos anteriores se llegó a la conclusión de que se requiere de un procesador moderno de 16 bits que cuente con un procesador de 32 bits en la misma familia para facilitar la expansión futura del sistema.

Algunas de las compañías que satisfacen estas necesidades son:

Compañía:	Microprocesador:	
Intel	8086 (16 bits)	80386 (32 bits)
National Semic.	16000 (16 bits)	32000 (32 bits)
Zilog	Z8001 (16 bits)	Z80000 (32 bits)
Motorola	68010 (16 bits)	68020 (32 bits)

A continuación se presenta una tabla comparativa de los espacios de direccionamiento de cada uno de ellos, y la forma de manejar los canales de datos y direcciones.

Micro	Espacio de direccionamiento directo	Espacio de direccionamiento Virtual	Canal de datos y direcciones Multiplexado
8086	1Mbyte	No tiene	SI
16000	16Mbytes	No tiene	SI
Z8001	48Mbytes	96Mbytes*	SI
68010	16Mbytes	64Mbytes*	NO

* Se requiere de circuitería externa

La tabla anterior muestra que el Z8001 es el que tiene mayor espacio de direccionamiento directo y virtual, y en segundo lugar el 68010, ya que los otros dos no cuentan con espacio de direccionamiento virtual. También se observa que el único que no multiplexa sus canales de datos y direcciones es el 68010, por lo que no necesita dispositivos externos para atrapar los datos y direcciones, esto representa un ahorro de espacio en la tarjeta y costo de componentes.

En lo que se refiere a la velocidad de operación, ésta se puede incrementar en dos formas: indirecta, al contar con un código compacto y amplia variedad de instrucciones y directa al aumentar la velocidad de ejecución de las instrucciones, es decir, al operar con relojes cada vez más rápidos. La siguiente tabla compara las diferentes versiones de velocidad a las que pueden trabajar los procesadores:

Micro	Velocidad del Reloj
8086	4, 5 y 8 MHz
16000	4, 6 y 8 MHz
Z8001	10 MHz
68010	8, 10 y 12.5 MHz

Todos ellos cuentan con un código compacto y diferentes velocidades de operación, siendo el 68010 el que puede operar a la más alta velocidad.

Para sistemas de multiprocesamiento, el Z8001 y el 68010 cuentan con pins que facilitan el manejo de buses de tiempo compartido. Ninguno de los otros dos micros tienen señales de hardware ni instrucciones de software que los ayuden para su operación en medios ambientes de múltiples procesadores.

También se debe considerar que para implementar el sistema, es necesario tener la seguridad de poder adquirir los componentes y manuales de operación con relativa facilidad. De las cuatro compañías mencionadas, Motorola es la que cuenta con mayores facilidades en nuestro país. Este factor es de suma importancia, ya que es muy problemático, costoso y tardado conseguir componentes y documentación directamente del extranjero. Por todas las comparaciones anteriormente expuestas, se llegó a la conclusión que el diseño de la tarjeta de procesamiento debería realizarse con el procesador 68010 de Motorola.

1.3.2 ARQUITECTURAS PARA SISTEMAS DE MULTIPROCESAMIENTO

Para lograr la comunicación entre microprocesadores y

recursos del sistema, es necesario un canal de comunicaciones moderno y de alta velocidad.

Debe tomarse en cuenta que en 1986 la primera generación de microprocesadores de 32 bits ha hecho impacto a gran escala en el mercado [8], por lo que un gran número de diseñadores han estado construyendo sofisticados sistemas de control de procesos en tarjetas diseñadas para 32 bits. Con miras a una futura expansión del sistema, se decidió utilizar un canal que soportara comunicación de datos hasta de 32 bits.

La siguiente tabla muestra los canales de comunicación con esta característica que se están utilizando actualmente, las compañías que los desarrollaron y las que los apoyan en el mercado.

CANAL	DESARROLLO	SOPORTE
FUTUREBUS	IEEE	Natnl. Semicond.
MULTIBUSII	INTEL	21 Vendedores
NUBUS	TI, MIT, WESTERN DIGITAL	TI, Máq. Lisp
VMEBUS	MOTOROLA, MOSTEK, SIGNETICS	147 Vendedores

El VMEbus fué el primer canal de 32 bits que se dió a conocer, adelántandose casi tres años al anuncio de su más cercano competidor, el MULTIBUS II de Intel. En 1981, al convertirse en un estandard del IEEE, el VMEbus ganó fuerte apoyo de aquellos que buscaban capacidad de datos y direcciones de 32 bits y un sistema que ofreciera clara migración de 16 a 32 bits. Actualmente los productos VME son los más fuertes en el mercado con ventas estimadas en $120 millones de dólares en 1985, cifra que se espera crezca al doble cada año hasta 1990.

En la tabla de la siguiente página se muestran las diferencias y similitudes entre las diferentes arquitecturas.

En esta tabla se puede observar que el VMEBUS es el único que no multiplexa su canal de datos, además de que mantiene protocolos de comunicación asíncrona. Esto último lo hace el más adecuado para las necesidades de manejo de interrupciones en ambientes asíncronos como el control de procesos y el control numérico en una fábrica, sin embargo, esta característica complica la transferencia de bloques de datos muy grandes, tarea para la cual el MULTIBUS II es el mejor, brillando en aplicaciones como sistemas paralelos grandes.

	FUTUREBUS	MULTIBUSII	NUBUS	VMEBUS
Arquitectura de comunicación	asíncrona	síncrona	síncrona	asíncrona
Canal de datos	multiplex.	multiplex.	multiplex.	nomultiplex.
Primario(bits)	32	32 a 10Mhz	32 a 10Mhz	16
Secundario(bits)	32,24,16,8	34,24,16,8	32,16,8	32,24,16,8
Transferencias no alineadas	SI	SI	NO	SI
Direccionamiento:				
Bytes Primarios	4Gbytes	4Gbytes	4Gbytes	16Mbytes
Bytes Secundarios	expandible	NO	NO	4Gbytes
Operaciones Emisoras	SI	SI	NO	NO
Velocidad de transferencia de datos(Mb/s)	117.7	40	37.5	57(máx.) 24(típica)

Una característica común entre los buses, es el rango de direccionamiento con capacidad de hasta 4Gbytes. Por otro lado, tanto el VMEBUS como el MULTIBUS II tienen una estructura de canal múltiple, en la que el VME utiliza tres trayectorias mientras que el MULTIBUSII utiliza cinco para manejar: datos, direcciones, mensajes, puertos de E/S y expansión del sistema.

La siguiente tabla muestra la aceptación de los buses en el mercado, así como su soporte en nuestro país.

	FUTUREBUS	MUTLIBUSII	NUBUS	VMEBUS
Aceptación	Pobre	Grande	Pobre	Grande
Soporte en México	NO	NO	NO	SI

Al comparar las características mencionadas hasta aquí, podemos decir que el punto más fuerte del VMEbus es su ventaja de tres años en el mercado. El los últimos seis meses se han añadido 40 vendedores más de VME. Para un usuario de este canal esto representa una gran ventaja pues existe una amplia oferta de productos VME en gran variedad de aplicaciones, desde memoria RAM, controladores de dispositivos de almacenamiento masivo, interfaces analógicas y relevadores hasta sofisticados sistemas de graficas, video y aplicaciones biomédicas. Por todo lo anterior, se eligió la arquitectura VMEBUS para este proyecto.

CAPITULO II

ESPECIFICACIONES DEL SISTEMA

CAPITULO II

ESPECIFICACIONES DEL SISTEMA

2.1 INTRODUCCION

En este capítulo se clasifican las diferentes arquitecturas de sistemas de cómputo para ubicar el sistema que se diseñó y se define el medio ambiente de multiprocesamiento sobre el que trabajará.

2.2 UBICACION DE LOS MULTIPROCESADORES EN LOS SISTEMAS DE COMPUTO

En esta parte se define al multiprocesamiento partiendo del esquema general de los sistemas de cómputo para distinguirlo de los diferentes conceptos que podrían causar confusión al hablar de una arquitectura específica.

2.2.1 ARQUITECTURAS DE LOS SISTEMAS COMPUTACIONALES

Los sistemas computacionales se han clasificado[9] según la secuencia de instrucciones y adquisición de datos, en cuatro categorías:

SISD: Secuencia única de instrucciones , secuencia única de datos.

SIMD: Secuencia única de instrucciones , secuencia múltiple de datos.

MISD: Secuencia múltiple de instrucciones , secuencia única de datos.

MIMD: Secuencia múltiple de instrucciones , secuencia múltiple de datos.

Esta clasificación se ilustra en la figura 2.1

Debido a que **MISD** es prácticamente una clasificación vacía, no se considera en este estudio.

A la categoría SISD pertenecen las computadoras de un solo procesador que ejecutan los programas en forma secuencial. A esta categoría también pertenecen las computadoras de un solo procesador que pueden ejecutar varios programas paralelamente pero no en un mismo instante (paralelismo aparente), por ejemplo los sistemas multiusuario de uniprocesador y las máquinas que cuentan con lenguajes de programación concurrente con un solo procesador. Esto no debe confundirse con las arquitecturas de procesamiento en paralelo que involucran dos o más procesadores trabajando en forma simultánea.

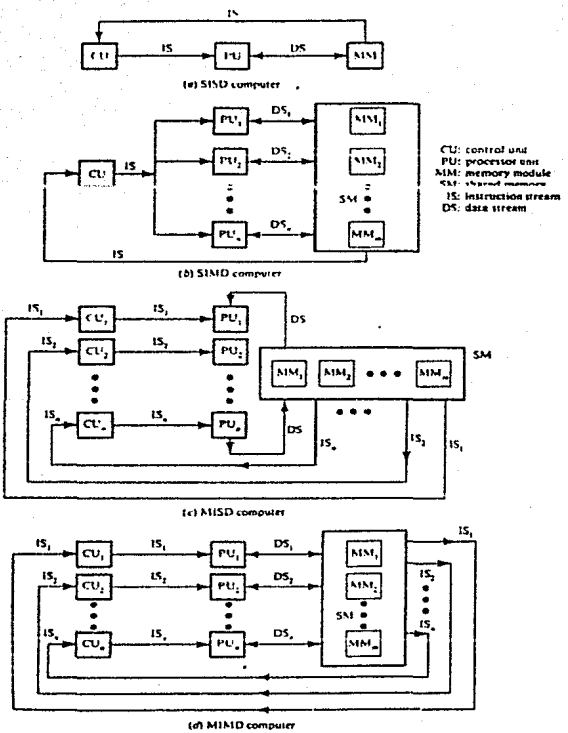


Figura 2.1 Clasificación de los Sistemas de Cómputo

A la categoría SIMD pertenecen los llamados sistemas de procesamiento paralelo (paralelismo verdadero); la unidad de control central toma y decodifica las instrucciones que son ejecutadas por la misma unidad de control (saltos condicionales e incondicionales) o transmitidas para su ejecución a otros

elementos de procesamiento. De acuerdo con la complejidad de las unidades de control, de los modos de direccionamiento y de la interacción entre los elementos de procesamiento, SIMD se puede dividir en las siguientes subcategorías:

-Procesadores vectoriales: Sistema computacional con múltiples unidades aritméticas y lógicas, su organización es tal que las operaciones se ejecutan en forma simultánea sobre un arreglo de datos (datos arreglados en forma de vectores).

-Procesadores asociativos: Los procesadores de este tipo accesan y operan sobre datos más por su contenido que por su dirección.

-Procesamiento por ensamble: Estos sistemas utilizan una combinación de procesadores vectoriales y asociativos para obtener una velocidad alta de salida. Tienen una computadora de carácter general que gobierna todos los elementos del procesamiento.

-Procesamiento pipeline: En este tipo de procesamiento una operación puede ser iniciada antes de que se temine la anterior. Es difícil de clasificar, porque según su funcionamiento pueden pertenecer a diferentes categorías. Se considera SIMD cuando su unidad aritmética de varias etapas ejecuta una misma operación sobre datos diferentes. El número de etapas es igual al de operaciones simultáneas que pueden realizar.

Por último, en la categoría MIMD el paralelismo resulta de ejecutar tareas independientes por diferentes procesadores sobre conjuntos de datos separados. La característica más importante de los sistemas MIMD es que las tareas que están ejecutando los procesadores pertenecen a un trabajo común que eventualmente se sincroniza para llegar a un resultado final. Es la gran diferencia de tener un conjunto de computadoras aisladas o interconectadas donde cada una ejecuta un trabajo totalmente

independiente de las demás. A esta categoría pertenecen los sitemas de multiprocesamiento, con las características mencionadas en el capítulo I.

2.2.2 CLASIFICACION DE LOS SISTEMAS DE MULTIPROCESADOR

Dependiendo del grado de interacción que exista entre los procesadores, los sistemas de multiprocesador se pueden clasificar en débilmente acoplados y fuertemente acoplados.

Respecto a esta clasificación, existe abundante literatura en la que se pueden encontrar definiciones ambiguas e inclusive contradictorias, creando confusiones, por lo que se decidió establecer las siguientes definiciones basadas en las características de la arquitectura del sistema sin tomar en cuenta la manera en que éste pudiera programarse. Yu-cheng y Glenn A. Gibson [13] definen:

- Sistemas de Multiprocesadores fuertemente acoplados.

En estos sistemas existe un procesador maestro y procesadores de apoyo trabajando como esclavos. Ambos procesadores maestro y esclavos no sólo comparten memoria y puertos de E/S sino que también comparten el mismo control lógico del bus y el reloj generador como se muestra en la figura 2.2. En esta configuración el procesador maestro está a cargo del control del acceso al bus.

La comunicación entre procesadores se realiza mediante un espacio de memoria compartido, en donde el procesador maestro deja un mensaje, y despierta al procesador esclavo mediante una señal, para que éste accese a la localidad de memoria compartida y ejecute la tarea que le sea asignada por el procesador maestro. Después de completar la tarea, el procesador esclavo debe avisar al maestro que ha terminado, haciendo uso de un bit de estado o

de una interrupción. El formato del mensaje varía dependiendo de la aplicación. Típicamente un mensaje debe de especificar que operación se debe realizar, los parámetros de entrada y la dirección de las localidades en la que se almacenarán los resultados.

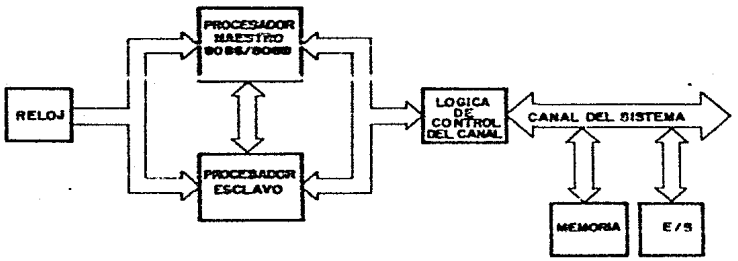


Figura 2.2 Sistema de multiprocesador fuertemente acoplado.

Cuando un procesador está utilizando el bus el otro procesador coloca sus líneas de salida en estado de alta impedancia.

Las configuraciones fuertemente acopladas, son recomendadas para sistemas de multiprocesador que no requieran un gran crecimiento. Este tipo de configuraciones permite al procesador maestro concentrarse en tareas de alto nivel, mientras los procesadores esclavos están dedicados a procesos específicos (por ejemplo un procesador de punto flotante), logrando así simplificar la programación y desarrollo de hardware, mejorando la flexibilidad y funcionamiento del sistema.

-Sistemas de Multiprocesadores débilmente acoplados.

Las configuraciones débilmente acopladas, consisten de varios módulos, en donde cada uno de ellos puede ser una tarjeta

de procesamiento capaz de ser el maestro del bus. Los diferentes módulos pueden compartir los recursos del sistema (por ejemplo: puertos de E/S y memoria), por lo que es necesario el uso de un sistema de sincronización de acceso a memoria que resuelva los problemas de concurrencia de los procesadores en la memoria compartida, ya que varios procesadores pueden utilizar una variable común.

Como se muestra en la figura 2.3 , no existe conexión directa entre los procesadores, y cada maestro del bus puede trabajar en forma independiente. La comunicación entre los procesadores se hace posible a través de los recursos compartidos. En adición a los recursos compartidos cada módulo puede tener su propia memoria privada y puertos de E/S. Los procesadores en los módulos separados pueden accesar simultáneamente sus recursos privados a través de un bus local y realizar búsquedas de instrucciones y referencias de datos locales en forma independiente. En este tipo de configuraciones debe existir un árbitro que esté encargado del control de el bus sobre el que se conectan los distintos procesadores.

Las configuraciones débilmente acopladas presentan las siguientes ventajas:

1.- Gran número de operaciones pueden ser realizadas teniendo más de un procesador.
2.- El sistema se puede expander en forma modular. Cada módulo maestro del bus es una unidad independiente y normalmente reside en una tarjeta separada, por lo tanto un módulo maestro del bus puede ser añadido o retirado del sistema sin afectar a los otros módulos del sistema.
3.- Una falla en un módulo, normalmente no causa un malfuncionamiento en todo el sistema, ya que éste puede ser fácilmente localizado y reemplazado.
4.- Cada maestro del bus tiene un bus local para accesar a

memoria privada o dispositivos de E/S, por lo que un mayor grado de procesamiento en paralelo puede ser llevado a cabo.

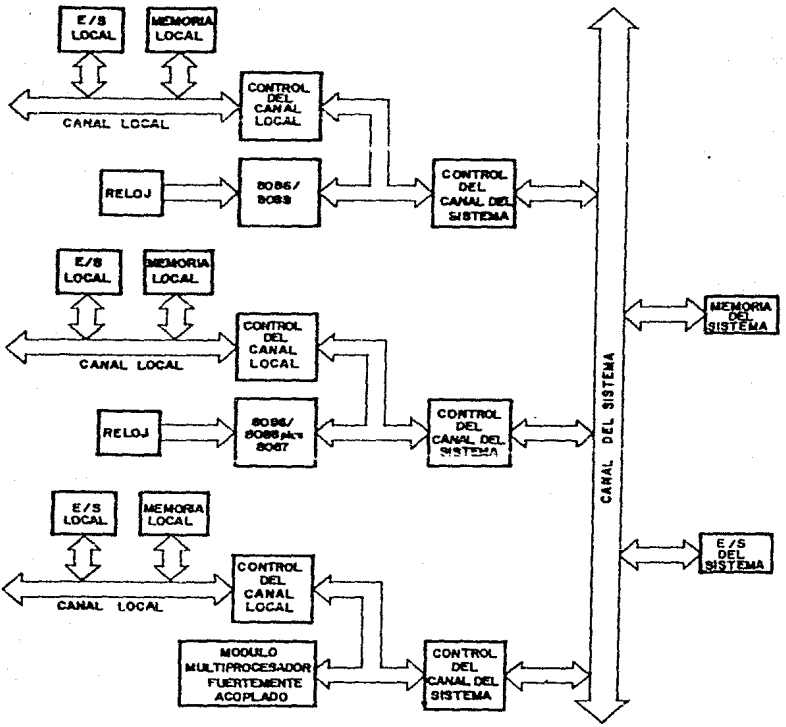


Figura 2.3 Sistema de Multiprocesdor débilmente acoplado.

Con base en los conceptos desarrollados en esta sección podemos ubicar el sistema diseñado en este trabajo dentro de la

categoría de sistemas de multiprocesadores debilmente acoplados, ya que como se observará en las siguientes secciones el diseño presenta las características de estos sistemas, contando adicionalmente con un bus serial.

2.3 MEDIO AMBIENTE DE MULTIPROCESAMIENTO

Como ya se expuso, la arquitectura de multiprocesamiento que posee el bus VME tiene varias ventajas para la aplicación que se desarrollará en este trabajo. A continuación se describen las características y se definen las especificaciones de esta arquitectura para poder proceder posteriormente al diseño.

Como se muestra en la figura 2.4, la arquitectura del sistema es de triple bus. El bus principal (VME) se utiliza para comunicar entre sí todas las distintas tarjetas que se encuentren conectadas en el chasis. Sobre este bus se realizan la mayor cantidad de transferencias de datos en el sistema. El bus privado (VMX) cumple la función de comunicar a cada procesador con un conjunto de recursos propios de ese procesador. Estos recursos no son compartidos directamente con otros procesadores, sin embargo, pueden ser accesados vía el bus principal. Finalmente se dispone de un bus serial (VMS) que comunica a todas las tarjetas (comunes, e inclusive privadas si es necesario). En este bus se pueden transmitir mensajes rápidamente ya sea entre dos tarjetas o a varias tarjetas simultáneamente.

2.3.1 BUS PRINCIPAL VME

Las especificaciones del bus VME definen un sistema de interface para interconectar dispositivos de procesamiento de

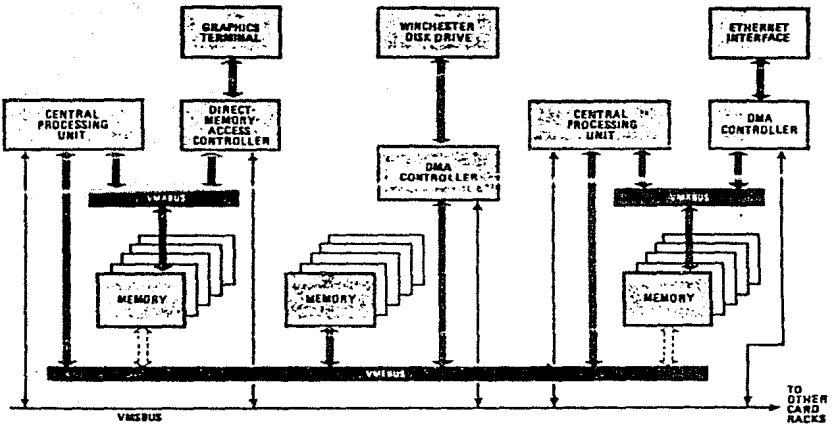


Figura 2.4 Arquitectura del bus VME

datos, de almacenamiento de datos, o de control de periféricos en una configuración eficiente. El sistema se ha concebido con los siguientes propósitos:

- Comunicación entre dos dispositivos en el bus sin afectar las actividades internas de otros dispositivos conectados al bus.
- Especificar las características eléctricas y mecánicas para el diseño de dispositivos que se comuniquen eficientemente y sin confusiones.
- Especificar protocolos que definan precisamente las interacciones entre el bus y los dispositivos conectados en él.
- Proporcionar una amplia latitud de diseño para optimizar costos y eficiencias.
- Proporcionar un sistema en el que la eficiencia esté

limitada por los dispositivos más que por la interface que los une.

Los módulos (parte de un circuito con un mismo propósito, figura 2.5) que pueden existir en el sistema son: Maestro, Solicitante, Manejador de interrupción, Esclavo, Interruptor, Subsistema Maestro, Subsistema Esclavo y Subsistema Controlador (Master, Requester, Interrupt handler, Slave, Interrupter, Master Subsistem, Slave Subsistem y Controler Subsistem).

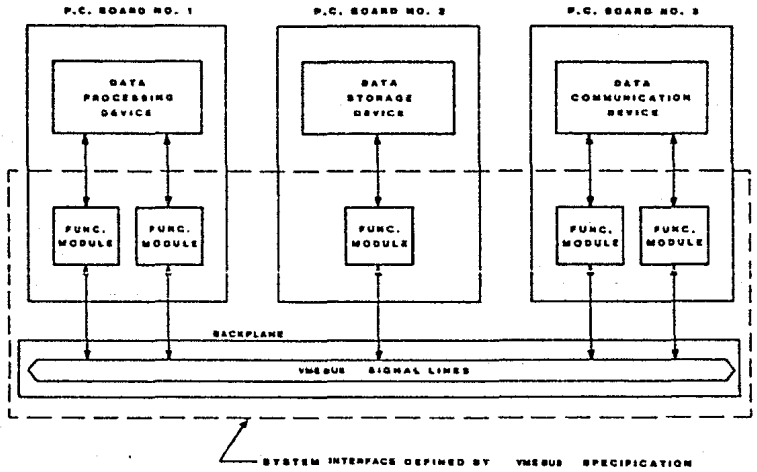


Figura 2.5 Modulos Funcionales.

2.3.1.1 Estructura básica del bus.

El bus consiste en cuatro grupos de líneas de señales

(buses) y un conjunto de módulos funcionales configurados según se requiera. Debe notarse que la definición de estos módulos no es un limitante para el diseño lógico, pues se pueden seguir los protocolos del bus y usar diferentes señales dentro de la tarjeta.

Las funciones de interface del bus se han dividido en cuatro categorías, cada una consistente en un bus y sus módulos funcionales asociados que realizan tareas específicas. Estas categorías se describen brevemente como sigue:

a. Transferencia de datos. Contiene las líneas de datos y direcciones y las líneas de control asociadas.

b. Arbitraje de transferencia de datos. Controla de manera ordenada las transferencias entre Maestros y garantiza que sólo un Maestro a la vez use el bus.

c. Prioridad de interrupciones. Contiene al bus de interrupciones y los módulos Interruptor y manejador de interrupciones para controlar un máximo de 7 niveles de interrupción.

d. Utilerías. Contiene en reloj del sistema, líneas de reset o inicialización, y detección de falla del sistema y de la alimentación.

En la figura 2.6 se muestra un ejemplo que utiliza los módulos antes mencionados conectados al bus correspondiente.

Las especificaciones eléctricas del bus son compatibles con los niveles TTL, es decir, el nivel alto será mayor o igual a 2.0V y el nivel bajo menor o igual a 0.8V; una línea estará en "transición" cuando su voltaje se mueva entre +0.8V y +2.0V.

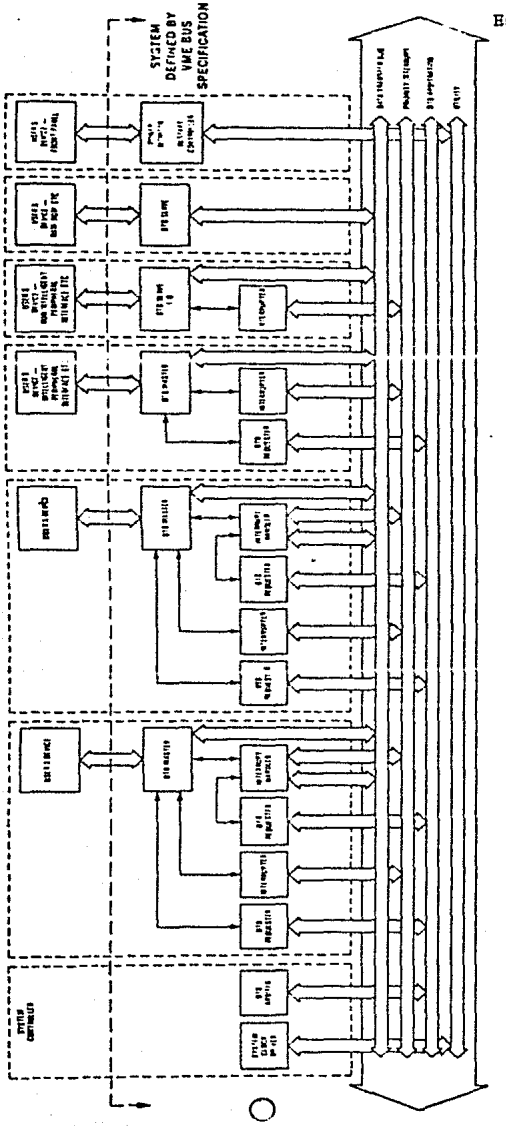


Figura 2.6

Módulos Conectados al bus VME

El protocolo determina cuando un módulo puede manejar y cambiar el nivel de las señales del bus, y cuando y cómo un módulo debe responder a una señal del bus.

Las señales del bus se pueden dividir en 2 clasificaciones generales:

-Señales entrelazadas, que requieren de cierto dialogo o "handshake", para coordinar la transferencia de datos, o para solicitar interrupción, por ejemplo.

-Señales emitidas, que permanecen en el bus durante cierto tiempo, anunciando condiciones especiales como una inicialización.

2.3.1.2 Transferencia de datos

El bus VME contiene un bus de transferencia de datos de alta velocidad, paralelo y asíncrono con las siguientes opciones:

D8, D16, D32	Tamaño del canal de datos
A16, A24, A32	Tamaño del canal de direcciones
BTO(n)	Bus Time Out
SEQ	Acceso secuencial ascendente

Operación:

La transferencia de datos siempre es iniciada por un Maestro. El Esclavo direccionado debe reconocer la transferencia después de que ya la ha realizado, ya sea lectura o escritura. La naturaleza asíncrona del bus permite que de esta manera sea el esclavo quien controle el tiempo que toma la transferencia. Cuando el Maestro recibe el reconocimiento de transferencia, termina el ciclo.

Líneas de direcciones:

La unidad de almacenamiento direccionable más pequeña es capaz de almacenar 8 bits de datos binarios, es decir, un byte. Dos localidades consecutivas de bytes (dirección par y la dirección impar consecutiva mayor) forman una localidad de una "palabra". Las líneas de dirección del bus comienzan con A01, para hacer énfasis en que las localidades de byte se direccionan con las líneas de "data strobe" 1 y 2 en lugar de la línea A00.

La opción BTO(n) especifica que el módulo abortará una transferencia de datos después de "n" microsegundos si no se ha recibido respuesta de un Esclavo. Esto protege contra bloqueos del bus por causa de una dirección errónea o un Esclavo descompuesto.

Modificadores de dirección:

Estas líneas permiten al Maestro pasar información adicional al esclavo durante la transferencia de datos, como por ejemplo:

- Partición del sistema. Se puede asignar un código diferente a cada maestro.

- Selección de mapa de memoria. El esclavo puede responder a diferentes direcciones.

- Acceso privilegiado. Los esclavos pueden responder a algunos maestros y a otros no.

- Tipo de ciclo. Pueden especificarse diferentes tipos de ciclo aparte del secuencial.

- Manejo distribuido de memoria. Pueden cambiarse dinámicamente el conjunto de registros de segmentos de memoria según la

tarea que se esté realizando.

- Rango de direccionamiento. El bus vme define tres rangos de direccionamiento: Corto (64K bytes, 16 líneas), estandar (16M bytes, 24 líneas) y extendido (4G bytes, 32 líneas). El esclavo deberá responder sólo a los rangos de direccionamiento que sea capaz de decodificar.

Líneas de control de transferencia de datos:

a. El Maestro controlará las siguientes líneas:

AS*	Dirección	(En toda transferencia)
DS0*	Byte impar	(Depende de la operación,
DS1*	Byte par	pero al menos una se activa)
LWORD*	Palabra larga	(Depende de la operación)
WRITE*	Lee o escribe	(Depende de la operación)

b. El Esclavo controlará las siguientes líneas:

BERR*	Error del bus	(Si la transf. no es posible)
DTACK*	Reconocimiento	(Si la transf. tuvo éxito)

- AS* indica que las líneas de dirección son estables
- DS0* y DS1* indica que los datos están listos en transferencias de escritura, y sustituyen la función que cumpliría A00 y permiten además la transferencia simultánea de 2 bytes, lo que no sería posible con solo A00.
- LWORD* indica que se transferirán 32 bits en las líneas de datos.

En la tabla 2.1 se resume el significado de estas últimas tres líneas.

- DTACK* es activada por el Esclavo para indicar que los datos se

recibieron correctamente en ciclo de escritura y que los datos están estables en el bus en ciclo de lectura.

- BERR* indica que los datos no se escribieron o que no se pudieron leer.

LWORD*	A01	DS0*	DS1*	CONSTRAINT/ACTION
L	H	X	X	Illegal (not on proper boundary)
L	L	L	L	Long word transfer
L	L	L	H	Illegal (only one data strobe)
L	L	H	L	Illegal (only one data strobe)
L	L	H	H	Transition State (no data strobe yet)
H	H	L	L	Transfer both bytes of odd word
H	H	L	H	Transfer odd byte of odd word
H	H	H	L	Transfer even byte of odd word
H	H	H	H	Transition State (no data strobe yet)
H	L	L	L	Transfer both bytes of even word
H	L	L	H	Transfer odd byte of even word
H	L	H	L	Transfer even byte of even word
H	L	H	H	Transition State (no data strobe yet)

Tabla 2.1

2.3.1.3 Mecanismos de arbitraje

El subsistema de arbitraje del bus VME está diseñado para:

- Prevenir el acceso simultáneo del bus por 2 Maestros.
- Control de petición de múltiples Maestros para uso óptimo de los recursos.

Opciones:

Se puede elegir una de las 3 opciones que define el bus para controlar el sistema. La opción PRI (priority) asigna siempre el bus con una prioridad fija desde la mayor (BR3) hasta la menor (BR0). La opción RRS (round robin select) consiste en una asignación rotatoria de prioridad. La opción ONE (nivel único) sólo atiende a peticiones en BR3 y depende de una estructura tipo

"cadena de margarita" (daisy-chain) (ver fig. 2.7) para determinar la prioridad. Por esta razón el subsistema de arbitraje deberá estar en la primera posición del bus (slot 1).

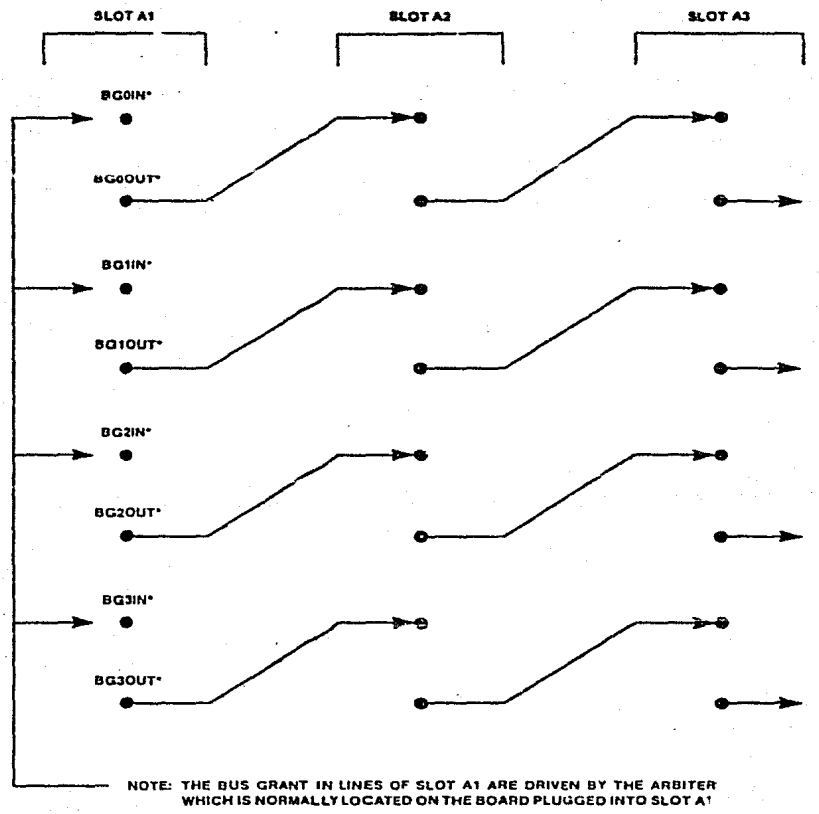


Figura 2.7 Cadena de Margarita.

Operación del Arbitro:

Excepto para la opción ONE, el árbitro acepta peticiones del bus en cuatro líneas (BR0-BR3), las cuales pueden ser manejadas por dispositivos colector abierto de manera que varios Maestros puedan compartir una línea de petición común. Cada línea de petición tiene una línea de concesión correspondiente en cadena de margarita (BG3IN/OUT a BG0IN/OUT). Si el bus está disponible cuando se recibe una petición, el árbitro responderá inmediatamente en la línea de concesión correspondiente al nivel de petición pendiente. Cuando el Maestro libera el bus, el árbitro responderá concediendo el control a la petición pendiente de más alto nivel.

En el caso de que no existan peticiones pendientes, el árbitro esperará en estado ocioso hasta que una petición del bus se reciba. El hecho de que las señales de concesión del bus estén en cadena de margarita representa un segundo nivel de prioritización intrínseco del bus, ya que los Maestros que compartan una línea de petición común tendrán diferentes prioridades según la posición del slot (ranura) que ocupen. La prioridad más alta la tendrá el Maestro más cercano al slot uno.
Líneas del bus de arbitraje:

El bus de arbitraje consiste en 6 líneas del bus VME y cuatro líneas interrumpidas o en cadena de margarita. A estas últimas se les dan nombres especiales; las señales entrando a los Maestros son llamadas "Entrada de concesión del bus" (BGxIN*), y las que salen del Maestro son "Salida de concesión del bus" (BGxOUT*), donde x es un número entre 0 y 3 inclusive. Las líneas están distribuidas como sigue:

a. El Maestro maneja las siguientes líneas:

1 línea de petición	(una de BR0* a BR3*)
1 línea salida de concesión	(una de BG0OUT a BG3OUT)
La línea de "ocupado"	(BBSY*)

b. El Arbitro maneja las siguientes líneas:

La línea de "limpia"	(BCLR*) (Opción PRI solamente, sirve para quitarle el bus a un Maestro cuando otro de mayor prioridad lo necesita)
4 líneas entrada de concesión	(BG0IN* a BG3IN*)

c. Las otras dos líneas están relacionadas con la secuencia de encendido y apagado del sistema de arbitraje:

SYSRESET*	línea de reinicialización del sistema
ACFAIL*	Falla en la alimentación

La línea BBSY* es activada por el Maestro una vez que se le ha concedido el control del bus de transferencia de datos por medio de una línea de concesión en cadena de margarita. Hasta que no suelte esta línea, no se le puede quitar el control a este Maestro.

La línea BCLR* es usada por un árbitro de opción PRI para informar al Maestro que tiene el control del bus que está pendiente una petición de mayor prioridad. El Maestro no tiene un límite de tiempo para soltar el bus, pero generalmente continuará hasta alcanzar un punto apropiado para interrumpir su transferencia y desactivar la línea BBSY*.

Secuencias de encendido y apagado:

A fin de interrumpir la operación del sistema de una manera ordenada cuando se presenta una falla de alimentación y para re-acomodar los datos antes de operar el sistema al recuperar la energía, se cuenta con las líneas ACFAIL* y SYSRESET*.

Al perder la energía se deben seguir las siguientes reglas:

a. Después de 200 microsegundos de la activación de ACFAIL*, los

Maestros no deberán pedir el bus para ninguna actividad excepto en respuesta a la falla de energía.

b. Cualquier Maestro con una petición pendiente debe limitar su actividad a 200 microsegundos.
c. Los Esclavos deben ignorar las peticiones de transferencia de datos iniciadas mas de 30 nanoseg. después de SYSRESET*

Al encender el sistema, los Maestros no pueden iniciar transferencias ni los esclavos responder a ellas hasta 30 nanoseg. después de que SYSRESET* se haga inactiva. Estas condiciones se muestran en la figura 2.8.

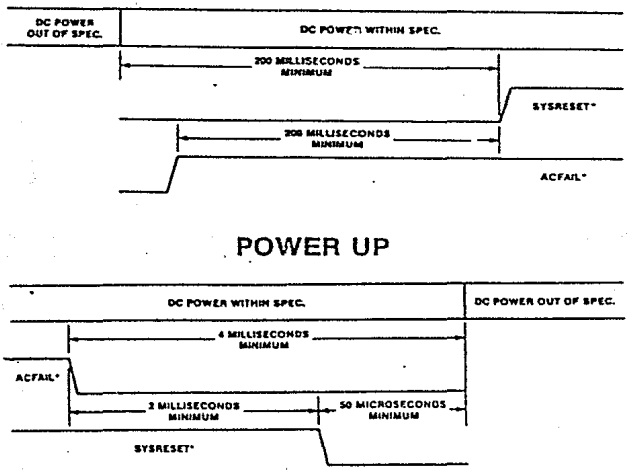


Figura 2.8 Condiciones para el SYSRESET*

2.3.1.4 Manejo de prioridad de interrupciones

Filosofía de la interrupción:

Los subsistemas de interrupción pueden dividirse en 2 grupos:

1. Sistemas de manejador único.- Tienen un procesador supervisorio que recibe y da servicio a todas las interrupciones.

2. Sistemas distribuidos.- Tienen dos o más procesadores que reciben y dan servicio a las interrrupciones del bus.

Cualquier sistema con capacidad para manejar interrupciones debe tener un conjunto de rutinas de servicio de interrupción en su software ejecutivo (sistema operativo). Se puede pensar en estas rutinas como tareas que son activadas por una interrupción.

Si se tiene un sistema operativo de tiempo real, estas rutinas de interrupción deben operar como tareas bajo este mismo sistema operativo.

Sistemas de manejador único:

En el dibujo 2.9 se ve la estructura de interrupción de un sistema de manejador único. Este tipo de arquitectura es muy conveniente para aplicaciones de control de máquinas o procesos.

Los procesadores dedicados son típicamente la interface con el proceso o máquina bajo control, por lo que su procesamiento debe ser interrumpido lo menos posible, ya que las tareas que realizan para el control pueden consistir en varias subtareas, algunas de las cuales son ininterrumpibles (puede perderse el control si una rutina no es terminada después de cierto tiempo de iniciada, por ejemplo). En resumen, el procesador supervisor es

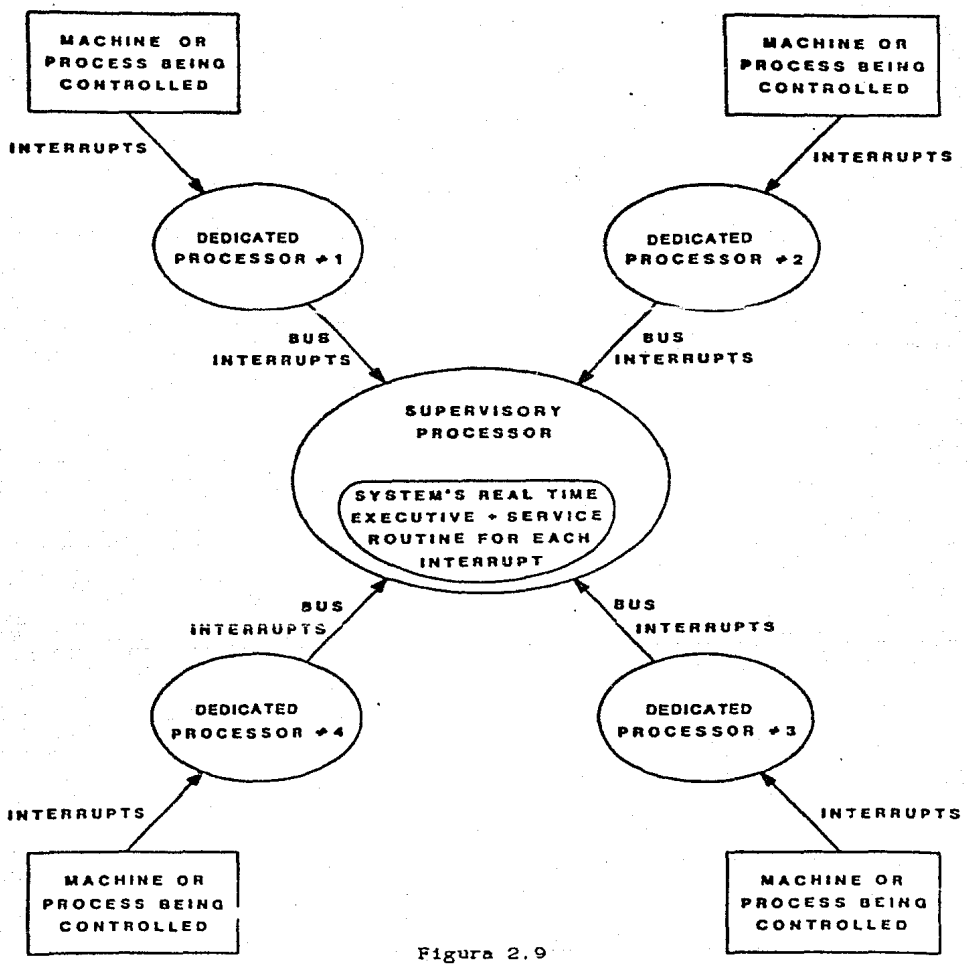


Figura 2.9

Interrupciones con manejador único.

el receptor de todas las interrupciones del bus, lo que permite dar servicio a las mismas siguiendo cierta prioridad. Los procesadores dedicados no son distraídos para dar servicio a interrupciones del bus sino que dan atención principal a las interrupciones de la máquina o proceso al que controlan.

Sistemas de interrupciones distribuidas:

El dibujo 2.10 muestra la estructura de interrupciones de un sistema distribuido. Este tipo de arquitectura es apropiado para aplicaciones en las que las tareas nuevas puedan asignarse al siguiente procesador disponible. Cada coprocesador ejecuta parte del sistema operativo, y da servicio sólo a las interrupciones dirigidas a ellos por otros procesadores en el sistema. Como el servicio a algunas interrupciones puede requerir el acceso a recursos del sistema, las partes del sistema operativo deben comunicarse por medio de memoria de acceso común para localizar recursos.

Señales usadas para manejar interrupciones:

Para generar y manejar interrupciones se usan el bus de transferencia de datos, el bus de arbitraje, y el bus de interrupciones. Los dos primeros ya han sido descritos.

Señales del bus de interrupciones:

Este bus consiste de siete líneas de petición de interrupción y dos líneas de reconocimiento de interrupción, una de ellas tipo cadena de margarita:

IRQ1* a IRQ7*
IACK*
IACKIN*/IACKOUT*

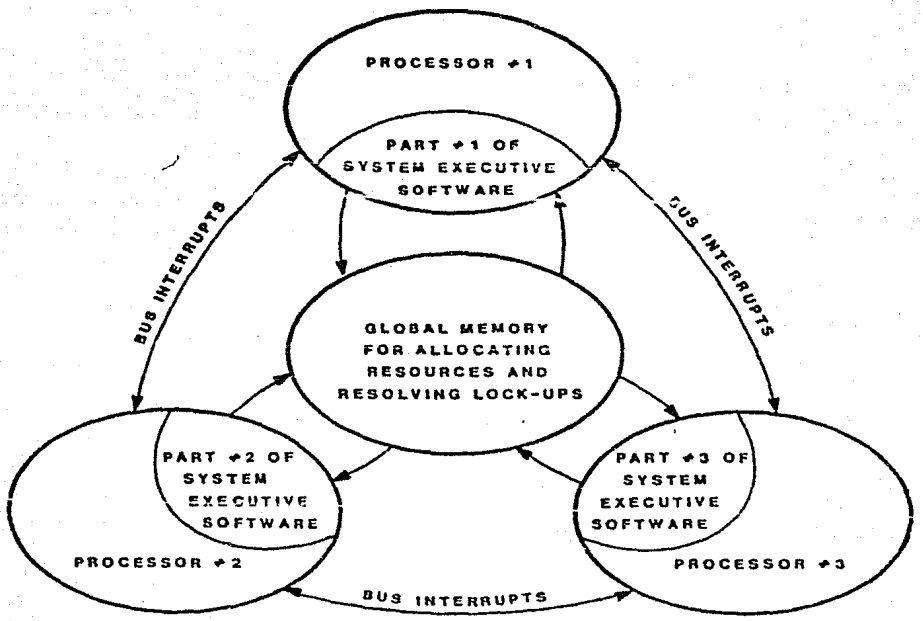


Figura 2.10 Interrupciones distribuidas.

Cada línea de petición puede ser activada por un interruptor para solicitar una interrupción. En un sistema de manejador único estas líneas tienen prioridad, IRQ7* tiene la más alta prioridad.

La línea IACK* está en todos los slots del bus y está conectada a IACKIN* en el primer slot. Cuando es activada, inicia una transición negativa a través de la cadena de margarita.

Como las 7 líneas de petición de interrupción pueden ser compartidas por dos o más Interruptores, es necesario asegurar que sólo uno de ellos sea reconocido. Esto se logra con la cadena de margarita de reconocimiento de interrupción que pasa por todos los conectores del bus VME representando un segundo nivel de prioridad de interrupción igual al que se describió en el arbitraje del bus.

2.3.1.5 Utilerías del bus

Las líneas de utilería proporcionan señales periódicas de tiempo y proveen de capacidad de inicialización y diagnóstico al bus. Estas líneas son:

SYSCLK*	Reloj del sistema
ACFAIL*	Falla de AC
SYSRESET*	Reset del sistema
SYSFAIL*	Falla del sistema

Reloj del sistema:

Es una señal independiente, de frecuencia fija a 16 Mhz, y ciclo de trabajo de 50%. Puede ser usada para generar retardos o funciones de temporización en una tarjeta. No tiene una relación fija de fase con otras señales de tiempo del bus.

Inicialización y diagnóstico del sistema:

La línea SYSRESET* es activada por un módulo supervisor de alimentación o por un interruptor manual. Cada que sea activada, debe permanecer así durante mínimo 200 milisegundos. Cuando es desactivada, el software del sistema ejecuta una secuencia de prueba, después de la cual, todos los módulos desactivarán la línea de SYSFAIL* que debieron activar al recibir el reset, y

pasarán al modo de operación normal. Si algún módulo detecta una falla, no debe desactivar SYSFAIL*.

Finalmente, en la figura 2.11 se muestra un dibujo del chasis para un sistema VME.

2.3.2 BUS PRIVADO VMX

Este es un bus subsistema diseñado para usarse con el bus VME. El bus privado es un camino secundario de alta velocidad sobre el que se pueden conectar hasta seis tarjetas que pueden transferir datos entre sí sobre el bus VMX, sin tener que esperar y sobrecargar al bus principal VME. Las objetivos del bus VMX son los siguientes:

a) Especificar las características eléctricas requeridas para una transferencia de datos confiable.

b) Especificar las características mecánicas requeridas compatibles con el bus VME.

c) Especificar protocolos que definan de una manera precisa la interacción entre las tarjetas conectadas al bus VMX.

d) Especificar la terminología usada para describir subsistemas basados en el bus VMX.

2.3.2.1 Transferencia de datos.

El bus VMX contiene un bus de transferencia de datos en paralelo de alta velocidad y asíncrono (VMX-DTB). El VMX-DTB es usado por un maestro primario y por uno secundario, para seleccionar esclavos y transferir datos hacia ellos, o para recibirlos.

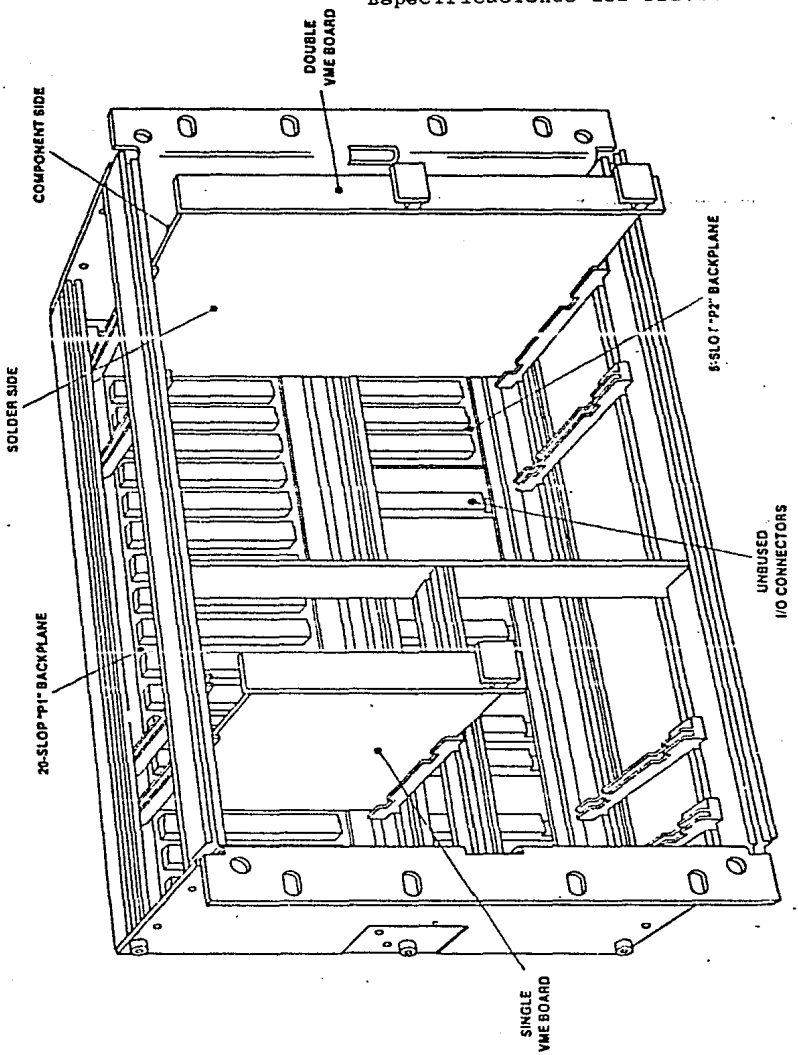


Figura 2.11 Chasis para un sistema VME.

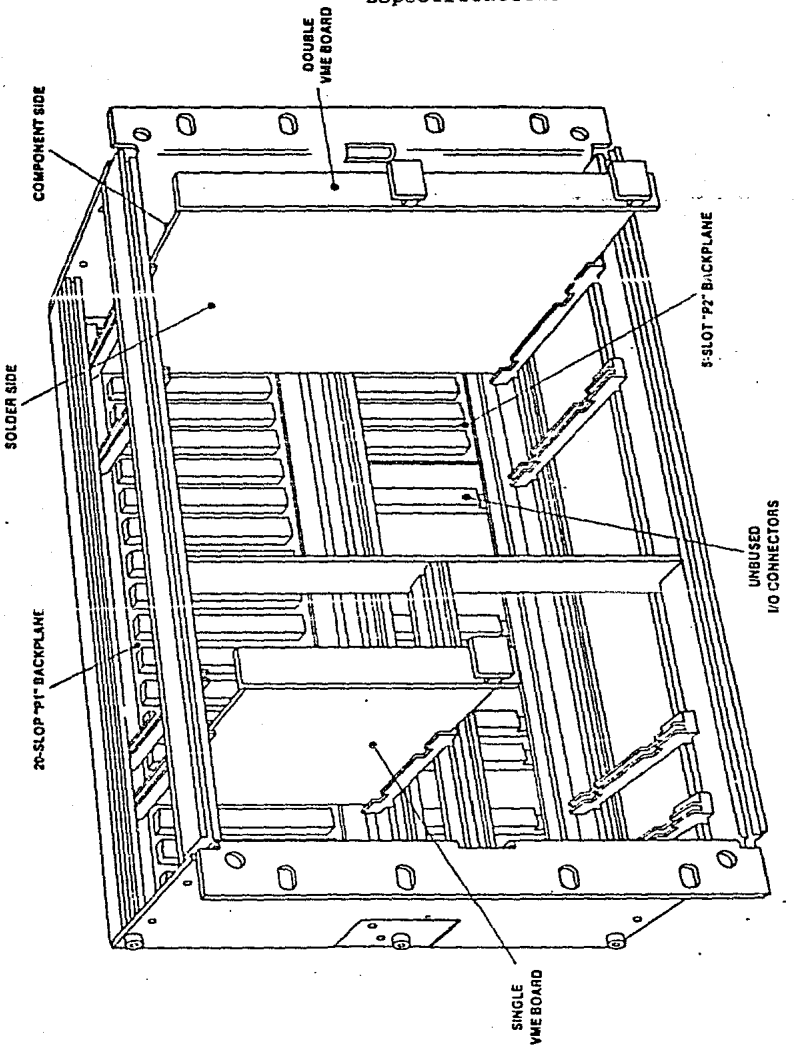


Figura 2.11 Chasis para un sistema VME.

El maestro primario es típicamente un procesador, y el secundario un controlador inteligente de E/S, el esclavo es típicamente un arreglo de memoria.

El bus de transferencia de datos se divide en direcciones, datos y líneas de control:

LWORD/A12-A11/A23	(23) Líneas de direcciones multiplexadas + palabra larga (LDS* + UDS* proveen el bit 24).
D00-D31	(32) Líneas de Datos.
LAS*	Habilitación de direcciones bajas
UAS*	Habilitación de direcciones altas
LDS*	Habilitación de datos altos (byte par)
UDS*	Habilitación de datos bajos (byte impar)
ACK*	Reconocimiento
DERR*	Error de dato
READ	Control de Lectura/Escritura

Las transferencias de datos pueden ser de 8, 16 o 32 bits. Transferencias de bytes hacia localidades de byte impares son transferidas sobre D00-D07, y las pares sobre D08-D15.

Las tranferencias de 32 bits se llevan a cabo sobre las líneas D00-D31. Cuando se realizan este tipo de transferencias, los 16 bits más significativos son transferidos sobre D00-D15 en vez de D16-D31 que es como se esperaba. Esto significa que las tarjetas de memoria de 32 bits deben tener un arreglo lógico para trasladar los 16 bits más significativos, a las 16 líneas de datos bajas (D00-D15).

El formato utilizado por el bus VMX para las tranferencias de datos es el mismo que se especificó en el bus VME (Ver tabla 2.1).

Todos los maestros y esclavos pueden hacer transferencias de datos de 8 y 16 bits. Algunos maestros y esclavos también pueden hacer transferencias de 32 bits. Maestros y esclavos que sólo pueden realizar transferencias de 8 y 16 bits son llamados Maestros D16 y Esclavos D16 respectivamente, de igual forma maestros y esclavos que pueden realizar transferencias de 32 bits son llamados Maestros D32 y Esclavos D32.

La definición de Maestros D16 permite conectar procesadores de 16 bits y DMA de 16 bits al bus VMX. Los Esclavos D16 proveen memorias de 16 bits.

Los Maestros D32 pueden hacer transferencias de palabras largas sobre D00-D31, pero también pueden actuar como Maestros D16. Esto significa que pueden accesar a Esclavos D16 así como a Esclavos D32. Si un Maestro D32 intenta leer o escribir un dato de 32 bits en un Esclavo D16, éste debe responder con un DERR*, indicando un error.

Los Maestros IMA son capaces de hacer ciclos indivisibles de dirección multiple, con lo que excluye a todos los otros maestros de hacer un acceso a la misma localidad del esclavo.

2.3.2.2 Protocolo de arbitraje.

El bus VMX cuenta con dos líneas de arbitraje las cuales son usadas para transferir el control del VMX-DTB entre el maestro primario y el macstro secundario:

SMRQ*	Petición del bus por el maestro secundario
SMACK*	Reconocimiento de petición del maestro secundario

El maestro secundario activa la línea de SMRQ* cuando necesita hacer uso del bus VMX para realizar una lectura o

escritura en una tarjeta de memoria conectada a este bus. El maestro primario puede activar la línea SMACK* en respuesta a SMRQ* para permitir que el maestro secundario haga uso del VMX-DTB.

Siempre que el sistema se encuentre en reset (línea de SYSRESET* activa) el maestro primario debe mantener inactiva la línea de SMACK* y mantenerla en ese estado hasta que el maestro secundario active la línea de SMRQ*. Por su parte el maestro secundario debe de mantener inactiva la línea de SMRQ* hasta que la línea de SYSRESET* sea inactiva.

Se debe aclarar que no existe una restricción en el tiempo que debe tardar el maestro primario en otorgar el bus, por lo que puede existir un maestro primario que nunca lo otorgue, en cuyo caso el maestro secundario nunca será capaz de realizar transferencias sobre el bus VMX.

Debido a que sólo se permite tener dos maestros en cada bus VMX (el maestro primario y el maestro secundario) existen sólo dos secuencias básicas de arbitraje. La primera es cuando el maestro primario otorga el control de VMX-DTB al maestro secundario, y la segunda cuando el maestro secundario regresa el control del VMX-DTB al maestro primario.

2.3.3 BUS SERIAL VMS

El bus serial constituye hardware adicional con la intención de ayudar a satisfacer las necesidades de un sistema de multiprocesadores. Algúnos ejemplos de la manera en que el bus serial cumple con estos propósitos son: comunicación de eventos de manera más eficiente que a través del arbitraje del bus principal, pues el bus serie está específicamente diseñado para esto y comunica a los procesadores entre sí o con cualquier

dispositivo en el sistema; ayuda al diseño de sistemas tolerantes a fallas proporcionando un camino adicional entre las tarjetas que puede usarse en la detección, localización y deshabilitación del módulo malfuncionante; permite la implementación de semáforos inteligentes para el acceso de recursos compartidos evitando el continuo acceso al bus principal para ver el estado del semáforo; permite esquemas de "pase de mensajes" (token passing) consistentes en una especie de cadena de margarita por software para el acceso a dispositivos compartidos; puede llegar a ser una alternativa al bus paralelo para tarjetas que no requieran una alta transferencia de datos.

2.3.3.1 Operación

El sistema de interface del bus serial consiste en dos líneas de señales (SERCLK y SERDAT*) y seis módulos llamados HEADER SENDERS (HS), HEADER RECEIVERS (HR), DATA SENDERS (DS), DATA RECEIVERS (DR), FRAME MONITORS (FM) y SERIAL CLOCK. La línea SERCLK es generada por el módulo SERIAL CLOCK. La línea SERDAT* puede ser mandada a nivel bajo por los seis módulos usando dispositivos de colector abierto; si ningún módulo activa esta línea, una resistencia la mantendrá en nivel alto. Esta línea es una señal activa baja (e.d. un uno es un voltaje bajo) por lo que con los dispositivos de colector abierto se tiene una operación O lógica cuando más de un módulo manda datos en SERDAT*.

La comunicación entre módulos se realiza enviando conjuntos de datos llamados "marcos" por la línea SERDAT*. Estos marcos se componen a su vez en "submarcos" que son enviados por varios módulos. Un marco se inicia cuando un módulo HEADER SENDER envía un "submarco de encabezado". Los otros módulos responden enviando submarcos de acuerdo a un protocolo preestablecido hasta llegar al final del marco.

Durante la transmisión del submarco de encabezado, el HEADER SENDER muestrea la línea SERDAT* y si detecta que otro módulo está activándola, suspende el envío. Este método de arbitraje permite el inicio simultáneo del envío de varios marcos sin afectarse entre sí: una transmisión se llevará a cabo y las otras serán intentadas después.

En una tarjeta, los módulos del bus serie se encuentran por grupos. Los más comunes de estos grupos son:

TIPO 1) Un HS y un FM
TIPO 2) Un HR y un flip flop
TIPO 3) Un HR y un DS
TIPO 4) Un HR y un DR
TIPO 5) Un HR, un DS y un DR

2.3.3.2 Protocolos de comunicación

Para realizar una transmisión de datos, el HS determina a través de su FM si se está ejecutando un marco. Si no, puede iniciar uno enviando un submarco de encabezado. Este submarco tiene un campo "S" y un campo "R", cada uno de diez bits. El HS pone en el campo S un código que corresponda a algún grupo tipo 3 o 4 en el bus, y un código de selección en el campo R que corresponda a un grupo tipo 4 o 5.

Cada HR en el bus serie compara estos códigos contra su propio código. Uno o más HR tipo 3 o 5 reconoce el código en el campo S y ordena a su DS que envíe datos. De manera similar, uno o más HR tipo 4 o 5 reconoce el código en el campo R y ordena a su DR que reciba datos. Después, el DS manda un submarco de tres bits indicando el número de bytes que pretende enviar, seguidos por los bytes de datos. El DR responde posteriormente con una indicación de que ha recibido los datos.

Cuando un HS se usa para escribir o borrar un flip flop, también envía un submarco de encabezado; sin embargo, mandará el código de un grupo tipo 2 en uno de los campos S o R y un código "vacío" (todos unos) en el otro, dependiendo de si quiere escribir o borrar el flip flop. El o los HR que reconozcan su código efectuarán la operación indicada en su flip flop.

De manera similar, grupos de módulos como los descritos se utilizan para reinicializar una o varias tarjetas, para accesar localidades de memoria en otra tarjeta, desconectar del bus una tarjeta en estado de falla, implementar semáforos o "pase de mensajes".

2.3.3.3 Marcos y submarcos del bus serial

Como ya se mencionó, un marco está compuesto por submarcos que son enviados por los diferentes módulos en secuencia después de que un HS ha enviado un submarco de encabezado. Dependiendo de que grupos de módulos se seleccionen, se pueden distinguir once clases de marcos:

1) Marco de escritura a flip flop.
2) Marco de borrado de flip flop.
3) Marco de semáforo.
4) Marco de pase de mensaje.
5) Marco de transferencia de 1 byte.
6) Marco de transferencia de 2 bytes.
7) Marco de transferencia de 4 bytes.
8) Marco de transferencia de 8 bytes.
9) Marco de transferencia de 16 bytes.
10) Marco de transferencia de 32 bytes.
11) Marco cancelado.

Todos los marcos contienen mínimamente tres de los

siguientes submarcos (el numero entre paréntesis son los bits que ocupan):

Encabezado(26) - tipo(3) - estado(3) - datos - interferencia(1)

Todos los marcos comienzan con un submarco de encabezado y un submarco de tipo de marco, y terminan con el submarco de bit de interferencia. El primero determina los módulos que participarán en el resto del marco, lo que a su vez determina el tipo de marco que se envía. Este submarco consiste en un bit de inicio, tres bits de prioridad del mensaje, un campo S de diez bits con el código de un enviador de datos o de un flip flop o de un semáforo o con "unos", un campo R de diez bits con el código de un receptor de datos o de un flip flop o con "unos", un bit de habilitación de arbitraje usado cuando dos o más DS comparten un código de selección, y un bit de confirmación que debe ser siempre uno para evitar que los módulos ignoren el cuadro. El submarco de tipo de marco indica si no se seleccionó ningún DS o el número de bytes que planea enviar el (los) DS seleccionado(s) por el campo S o si el marco fue cancelado por un MR. El bit de interferencia únicamente puede ser un uno si alguno o varios módulos FM se salen de sincronía debido a ruido en el sistema. En este caso todos los módulos ignorarán este marco, mismo que será enviado nuevamente después de la resincronización del bus serie.

El submarco de estado del marco es enviado por un FM al final de todos los marcos excepto cuando es cancelado, y es usado para diagnosticar los problemas (condiciones excepcionales) que pudieron haber surgido durante la transmisión del marco.

El submarco de datos se incluye en los marcos de transmisión de datos y su longitud puede ser 2, 4, 8, 16 o 32 bytes, según lo especifique el DS.

2.3.3.4 Líneas y módulos

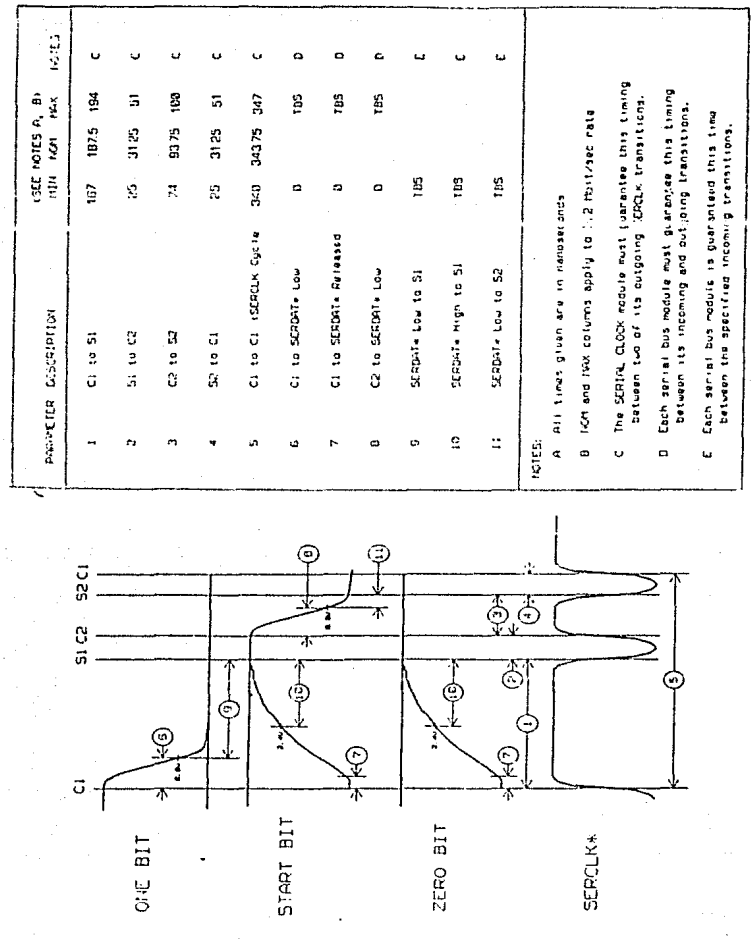
El bus serial incluye tres líneas: el reloj serie (SERCLK), los datos en serie (SERDAT*), y la reinicialización del sistema (SYSRESET*).

Un ciclo del reloj consiste en cuatro transiciones y se usa para enviar un bit en SERDAT*. Hay tres tipos de bits que pueden enviarse: un "uno", un "cero" y un "bit de inicio", los cuales se distinguen muestreando SERDAT* en ambas transiciones de bajada del reloj. En el primer flanco de subida, todos los módulos pueden cambiar el nivel de SERDAT*, mientras que en el segundo, sólo se cambiara SERDAT* (de alto a bajo) cuando un HS esté enviando un bit de inicio. En el siguiente diagrama de tiempos se ven claramente estas relaciones (figura 2.12a).

El módulo SERIAL CLOCK debe estar ya sea en la primera o en la última tarjeta del sistema. Cuando SYSRESET* sea verdadera, este módulo debe mantener SERDAT* en "uno" y soltarla con el primer flanco de subida del reloj serial (también generado por este módulo).

El módulo HEADER SENDER recibe de un maestro toda la información que enviará en en submarco de encabezado, que básicamente sirve para seleccionar uno o más HR del bus serie. Un HS siempre debe estar acoplado a un FM para avisarle si ganó el arbitraje del bus y comenzó un marco, o si intentó comenzar uno y perdió el uso del bus.

Un módulo HEADER RECEIVER puede estar solo o acoplado a un DS o/y un DR. Su función primaria es comparar los campos S y R de un submarco de encabezado y avisar a sus módulos acoplados en caso de que identifique su propio código de selección. Un maestro escribe en el HR el código que debe identificar y lo habilita para aceptar marcos con ese código en el campo S o R.



	PARAMETER DESCRIPTION	(SEE NOTES A, B) MIN	MAX	NOTES
1	C1 to S1	187	1875	194	C
2	S1 to C2	125	31.25	31	C
3	C2 to S2	74	93.75	188	C
4	S2 to C1	25	31.25	51	C
5	C1 to C1 (SERCLK Cycle	340	343.75	347	C
6	C1 to SERDATA Low	0		185	D
7	C1 to SERDATA Released	0		185	D
8	C2 to SERDATA Low	0		185	D
9	SERDATA Low to S1	185			E
10	SERDATA High to S1	185			C
11	SERDATA Low to S2	185			E

NOTES:

A All times given are in nanoseconds

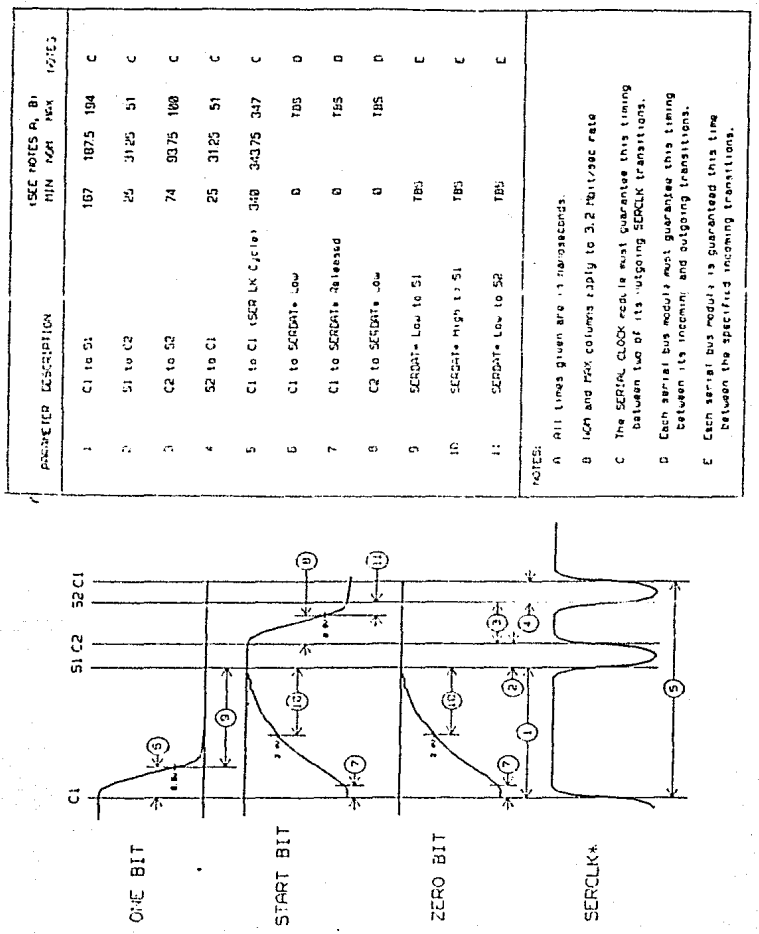
B MIN and MAX columns apply to 1.2 Mbit/sec rate

C The SERIAL CLOCK module must guarantee this timing between two of its outgoing SERCLK transitions.

D Each serial bus module must guarantee this timing between its incoming and outgoing transitions.

E Each serial bus module is guaranteed this time between the specified incoming transitions.

Figura 2.12 Diagrama de tiempos del bus serial.



PARAMETER	DESCRIPTION	(SEE NOTES A, B) MIN	MAX	MAX	NOTES
1	C1 to S1	167	1875	194	C
2	S1 to C2	25	3125	51	C
3	C2 to S2	74	9375	100	C
4	S2 to C1	25	3125	51	C
5	C1 to C1 (SER LK Cycle)	320	34375	347	C
6	C1 to SERDATA Low	0		TBS	D
7	C1 to SERDATA Released	0		TBS	D
8	C2 to SERDATA Low	0		TBS	D
9	SERDATA Low to S1	TBS			E
10	SERDATA High to S1	TBS			E
11	SERDATA Low to S2	TBS			E

NOTES:
A. All times given are in nanoseconds.
B. MIN and MAX columns apply to 3.2 Mbit/sec rate
C. The SERIAL CLOCK module must guarantee this timing between two of its outgoing SERCLK transitions.
D. Each serial bus module must guarantee this timing between its incoming and outgoing transitions.
E. Each serial bus module is guaranteed this time between the specified incoming transitions.

Figura 2.12 Diagrama de tiempos del bus serial.

Un módulo DATA SENDER debe estar acoplado a un HR. Su función es tomar datos de la lógica interna de la tarjeta y enviarlos por el bus serie cuando se lo indique su HR. De manera similar, un módulo DATA RECEIVER tomará datos del bus serie y los presentará a la lógica interna cuando su HR así se lo indique.

El módulo FRAME MONITOR completa los módulos que se han descrito, reportando el resultado de la transmisión de un marco a la lógica interna de la tarjeta con el HS que inició el marco. Cumple con dos importantes funciones. Primero sigue la transmisión de todo marco e informa a su HS si el bus está libre para iniciar un nuevo marco. En segundo lugar, continuamente observa si se envía un bit de inicio: si esto sucede mientras está siguiendo un marco, habrá ocurrido una desincronización y generará el bit de interferencia para obligar a los módulos a ignorar el marco transmitido y permitir que el bus se sincronice de nuevo.

CAPITULO III

DISEÑO LÓGICO

CAPITULO III

DISEÑO LOGICO

3.1 DIAGRAMA A BLOQUES

En la figura 3.1 se muestra el diagrama a bloques de la tarjeta que se utilizará como primer prototipo para el sistema que se describió en el capítulo precedente, por lo que no se incluyen las interfaces a VMX y VMS (estas se considerarán en una versión futura del proyecto). El objetivo de este capítulo es describir detalladamente el diseño lógico de cada uno de esos bloques.

Las características de esta microcomputadora son: CPU 68010 trabajando a 8MHz, un controlador de acceso directo a memoria de dos canales, 128K o 512K de RAM dinámica, dos bases para EPROM seleccionable desde 4K hasta 128K, reloj de tiempo real, un puerto serie RS232, dos puertos paralelos (uno de ellos con interfaz al estandard Centronics), un contador/temporizador de 24 bits, la opción ONE del árbitro del bus VME y un módulo REQUESTER que define a la tarjeta como maestro de este bus.

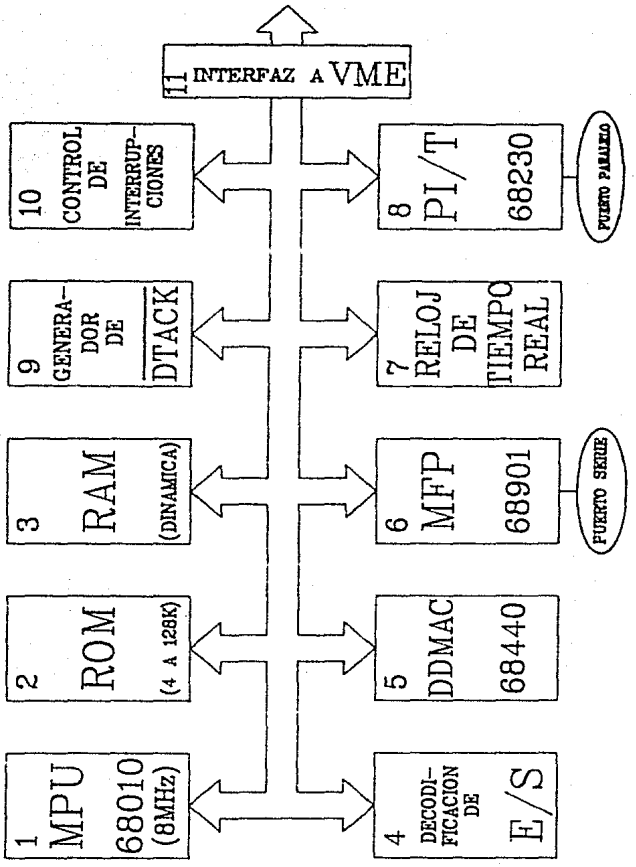


Figura 3.1 Diagrama a bloques

3.2 CPU

La figura 3.2 muestra los elementos principales de la tarjeta:

-La unidad microprocesadora de 16 bits, cuyas características principales son: registros internos de direcciones y datos de 32 bits, rango de direccionamiento directo de 16 Megabytes, entrada/salida mapeada en memoria, soporte para memoria/máquina virtual, 57 tipos de instrucción, operaciones de bucle de alta eficiencia, 14 modos de direccionamiento y 5 tipos principales de datos.

-Reloj: la señal de reloj se obtiene de un oscilador de cristal de 8 MHz.

-Circuitería de reset. Utilizando un circuito multivibrador monoestable se obtienen pulsos de un ancho adecuado para activar las líneas HALT y RESET simultáneamente y generar de esta manera una reinicialización general del sistema, ya sea durante el encendido o por medio de un interruptor de botón o a partir de la señal SYSRESET* del bus VME (misma que opcionalmente puede obtenerse del pulso de reinicio de encendido mediante un puente).

Con las dos últimas opciones de reinicialización (botón y señal SYSRESET*), el contenido de la memoria DRAM se conserva intacto ya que el ancho del pulso de reinicio (en estos casos de 1.3 *f*seg.) es mucho menor que el tiempo de refresco de la DRAM, que es de lo primero que se haría al programar el DDMAC en la rutina de reinicio.

-Circuito vigilante (watch-dog). Se cuenta con un circuito que activa automáticamente la señal BERR cuando después de aproximadamente 5 microsegundos de activada la línea AS no se ha recibido la señal DTACK, terminando de esta manera un intento de acceso a un área no utilizada de memoria.

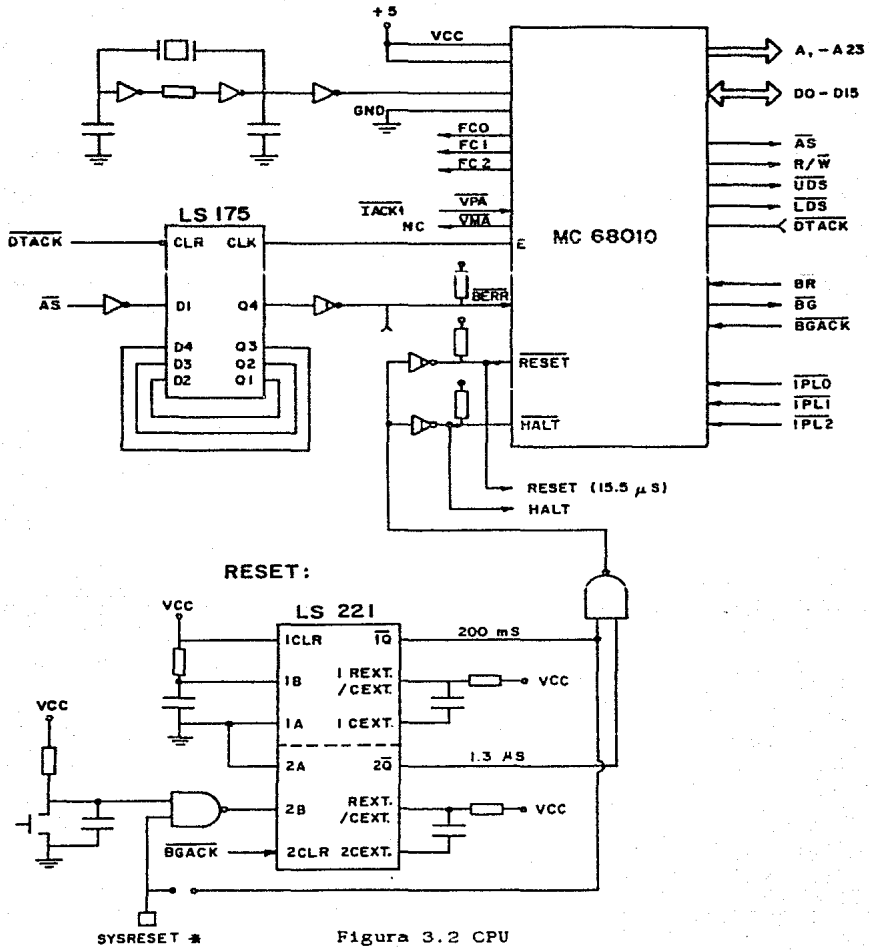


Figura 3.2 CPU

3.3 MEMORIA

3.3.1 MAPA DE MEMORIA

El mapa de memoria se distribuyó de manera que permitiera el crecimiento del sistema y al mismo tiempo se facilitara la decodificación.

La memoria EPROM se coloca al principio del mapa debido a que durante el ciclo de reinicialización el procesador busca el vector de reset en los primeros ocho bytes.

El mapa de memoria es el siguiente:

000000 - 07FFFF	EPROM 512K bytes
080000 - 0FFFFF	DRAM 512K bytes
100000 - FFEFFF	MEMORIA VME APROX. 16M bytes
FFF000 - FFF1FF	E/S EN TARJETA 512K bytes
FFF200 - FFFFFF	E/S VME APROX. 4K bytes

Dado que la totalidad del espacio de memoria direccionable por el procesador está asignada, es necesario decodificar todas las líneas de dirección.

Es evidente que utilizando decodificadores esto resultaría ineficiente; además que el diseño de la decodificación debe permitir programar la cantidad de memoria que se esté utilizando;

esto resulta más fácil con un arreglo lógico e interruptores programables.

3.3.2 MEMORIA EPROM

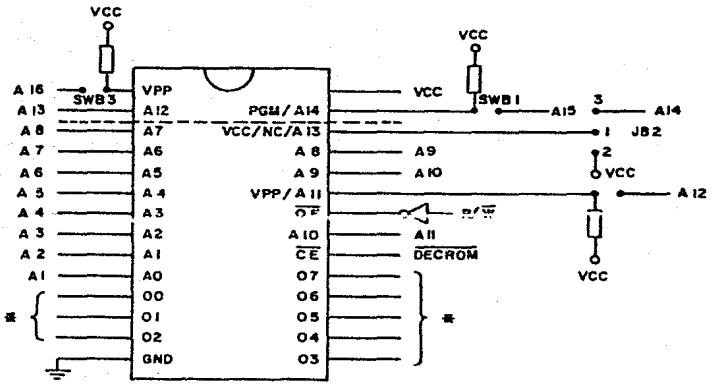
Debido a la gran variedad de chips de memoria ROM existentes y a la versatilidad que se le quiere dar al sistema, se realizó un diseño que permite al usuario tener la posibilidad de utilizar cualquier chip de memoria EPROM desde 2k*8, 4k*8, 8k*8, 16k*8, 32k*8 y 64k*8 bytes.

Cualquiera de estos circuitos (24 o 28 patas) se colocan en las bases de 28 patas mostradas en la figura 3.3 y se programa el tipo de memoria que se desee usar mediante puentes removibles e interruptores; la posición que estos deben de tener para los distintos tamaños de memoria se muestra en la misma figura. Los circuitos de 24 patas deben colocarse debajo de la línea punteada que se aprecia en el dibujo.

La decodificación de la memoria EPROM se lleva a cabo mediante compuertas NOR, y una compuerta NAND, contando además con 7 interruptores para programar el tamaño de memoria a utilizar.

En la figura 3.4 se ilustra la manera en la que se decodificó la memoria de acuerdo al mapa de memoria establecido anteriormente, así como la posición que deben tener los interruptores para decodificar los diferentes tamaños de memoria. Se debe tomar en cuenta que se proporciona el tamaño de la memoria considerando que se tienen dos bancos de memoria, y que se hace uso de las líneas UDS y LDS.

BASE EPROMS

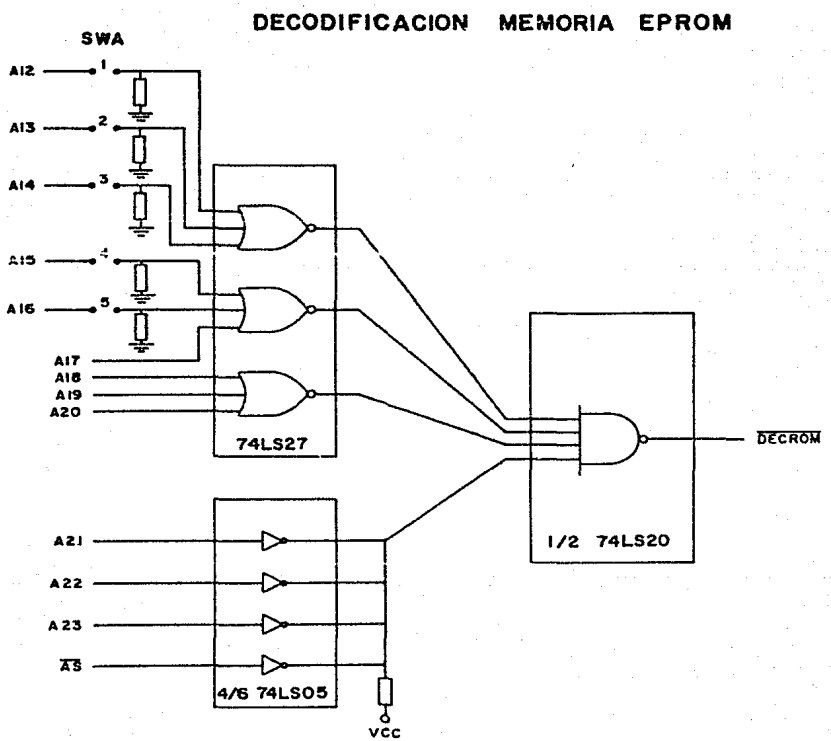


* DO – D7 EN UNA BASE
D8 – D15 EN OTRA BASE

TAMAÑO MEMORIA	SWB 1	SWB 2	SWB 3	JB2	
2K X 16	X	O	X	1-2	4K bytes
4K X 16	X	•	X	1-2	8K bytes
8K X 16	O	•	O	NC	16K bytes
16K X16	O	•	O	1-3	32K bytes
32K X 16	•	•	O	1-3	64K bytes
64K X 16	•	•	•	1-3	128K bytes

O ABIERTO
• CERRADO
X IRRELEVANTE
NC NO SE CONECTA

Figura 3.3 Base de los circuitos EPROM



TAMAÑO MEMORIA		SWA 1	2	3	4	5
4 K	bytes	●	●	●	●	●
8 K	bytes	o	●	●	●	●
16 K	bytes	o	o	●	●	●
32 K	bytes	o	o	o	●	●
64 K	bytes	o	o	o	o	●
128 K	bytes	o	o	o	o	o

● CERRADO
o ABIERTO

Figura 3.4 Decodificación de memoria ROM

3.3.3 MEMORIA RAM

Los requerimientos de memoria RAM en la misma tarjeta del procesador estaban determinados por los siguientes factores:

- Suficiente memoria para hacer eficiente el acceso a dispositivos de almacenamiento masivo.

- El espacio ocupado por los circuitos de memoria y soporte de la misma no deben exceder el espacio disponible dentro de la tarjeta.

- Económicos.

Las opciones principales consisten en memoria estática o memoria dinámica. Para ambas opciones se requieren dos bancos de memoria de 8 bits, ya que el procesador tiene un canal de datos de 16 bits. A continuación se presenta una comparación entre las dos opciones en base a la cantidad de memoria, su costo y el área que éstas ocupan (los precios son de Abril de 1987).

Memoria Estática.

Circuitos Int.	Area aprox.(cm2)	Cantidad de Mem.	Costo(dlls.)
2- 6116 (2k*8)	10	4k bytes	$3.90
2- 6264 (8k*8)	10	16k bytes	$7.50
16-6264 (8k*8)	100	128k bytes	$60.00

Memoria Dinámica

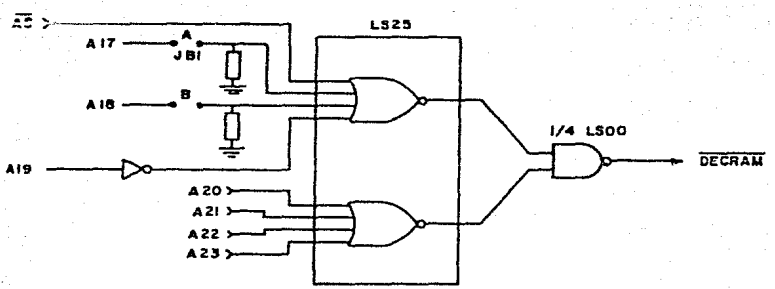
Circuitos Int.	Area aprox.(cm2)	Cantidad de Mem.	Costo(dlls.)
18-4164 (64k*1)	50	128k bytes	$20.60
18-41256 (256k*1)	50	512k bytes	$49.50

De acuerdo a este estudio concluimos que la opción que cumple satisfactoriamente con nuestros factores es la memoria dinámica, pues aunque requiere de circuiteria adicional para control de acceso y de refresco, la diferencia en precio y área para una misma cantidad de memoria justifica plenamente la elección.

El diseño de la memoria proporciona la opción de utilizar circuitos de 64 Kbits o 256 Kbits, para obtener un total de 128K o 512 Kbytes respectivamente de memoria dinámica con paridad (figura 3.5). Esta memoria está organizada en dos bancos de 8 bits más un bit de paridad que pueden accesarse individualmente o como palabras de 16 bits (figura 3.6). Dos circuitos especializados generan el bit de paridad en cada escritura y la verifican en cada lectura, generando un error de bus (activando BERR) si ocurre algún error durante estos procesos (figura 3.7).

Control del refresco. El conteo del tiempo de refresco se realiza en un canal del DMAC, lo que hace que la circuitería adicional necesaria sea muy sencilla. En cada ciclo de refresco la señal ACK0 del DMAC inhibe la generación de las señales SEL y CAS que durante un acceso normal a la RAM se producen 15 y 35 nanosegundos después de RAS, respectivamente (ver figura 3.8).

DECODIFICACION :



TAMANO	MEMORIA	PUENTE JBI A Y B
128 K	(4164'S)	CERRADOS
512 K	(41256'S)	ABIERTOS

Figura 3.5 Decodificación de DRAM

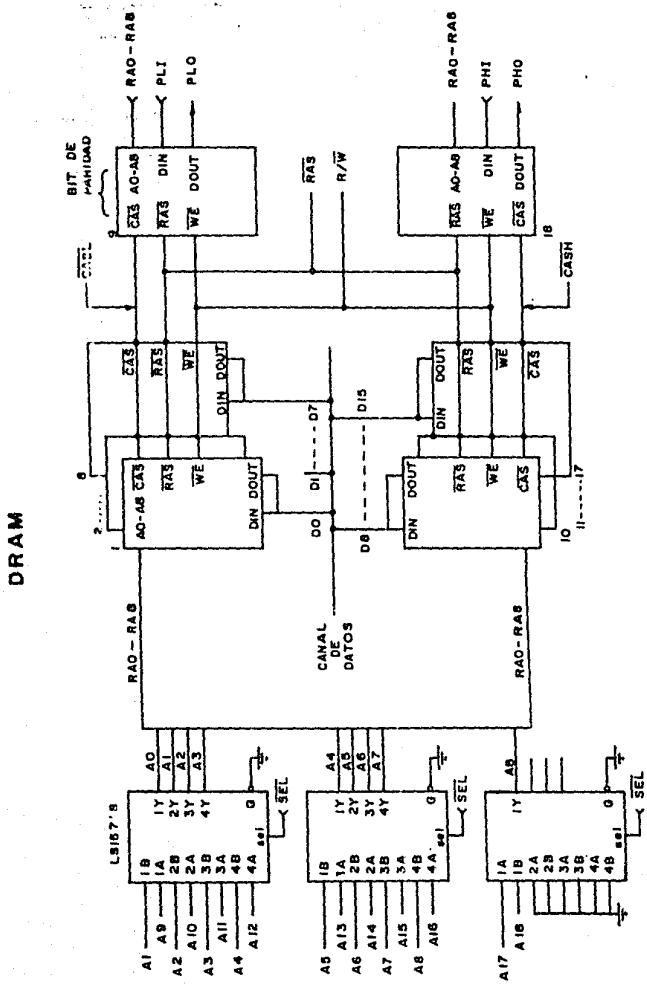


Figura 3.6 Organización de la DRAM

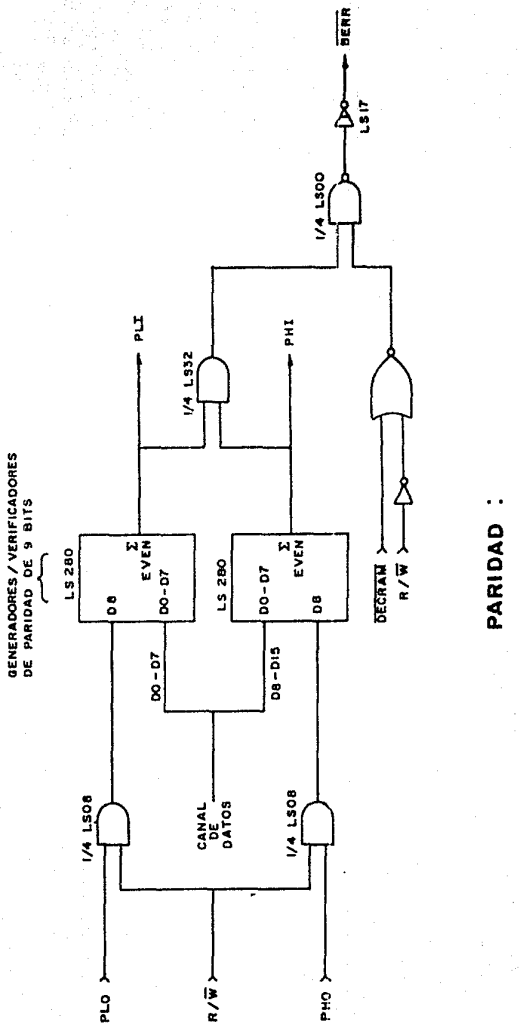


Figura 3.7 Control de paridad

SEÑALES DE CONTROL:

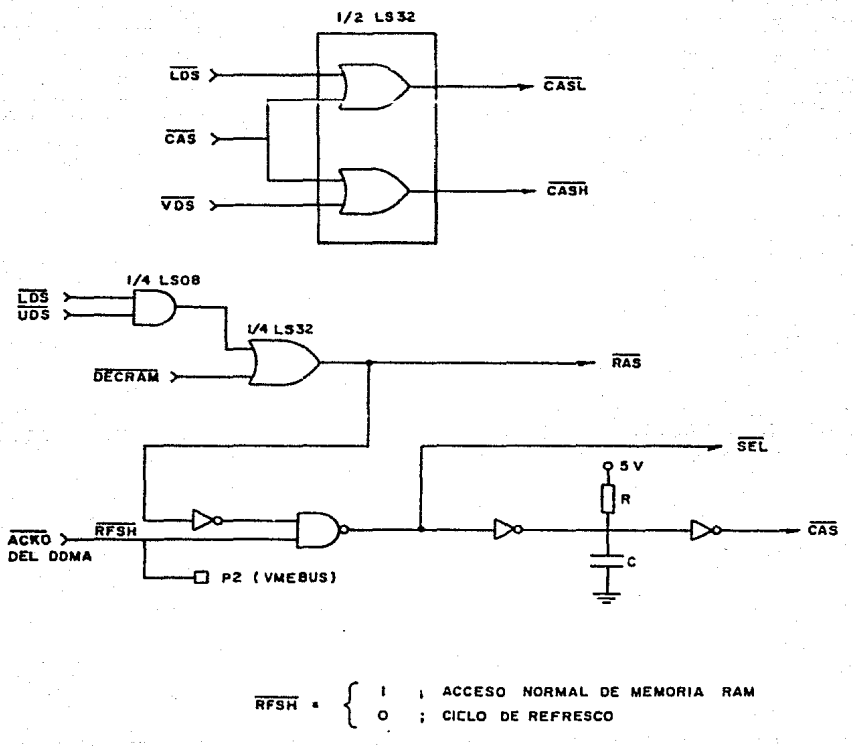


Figura 3.8 Señales de control de DRAM

Esto indica a los circuitos de memoria que se trata de un ciclo de refresco. Este ciclo dura 4 ciclos de reloj y se repite 256 veces seguidas refrescando todos los renglones de los dos bancos de memoria en un total de 128 microsegundos. Los circuitos de memoria requieren de ser refrescados cada 4 milisegundos, por lo que el porcentaje de utilización del canal de datos del sistema es de 96.8%.

3.4 PUERTOS

3.4.1 DECODIFICACION

En la decodificación de los puertos de entrada/salida se debe tomar en cuenta el número de registros que contienen los puertos, así como espacio disponible para el usuario, para que tenga la posibilidad de conectar más puertos o periféricos si lo desea.

El número de registros que utiliza cada uno de los puertos, el espacio de direccionamiento requerido así como el espacio de direccionamiento asignado se muestran en la tabla 2.1 de la página siguiente.

En esa tabla se observa que el MFP contiene 24 registros, pero debido a que los registros internos están organizados en bytes y su canal de datos es de 8 bits, el MFP sólo utiliza la pata de LDS de el procesador, por lo que debe escribirse en la parte baja del canal de datos, desperdiciándose 24 bytes que corresponderían a la parte alta (UDS), sumando en total 48 bytes, por lo que deben asignarse 64 bytes de espacio de direccionamiento.

DISPOSITIVO	No DE REG. INTERNOS (bytes)	ESPACIO DE DIR. REQUERIDO (bytes)	ESPACIO DE DIR ASIGNADO	No DE BYTES
DDMAC	128	128	FFF000-FFF07F	128
PI/T	32	64	FFF080-FFF0BF	64
MFP	24	64	FFF000-FFF0FF	64
RTR	16	32	FFF100-FFF13F	64
DISPONIBLE PARA EL USUARIO			FFF140-FFF1FF	192

Tabla 2.1

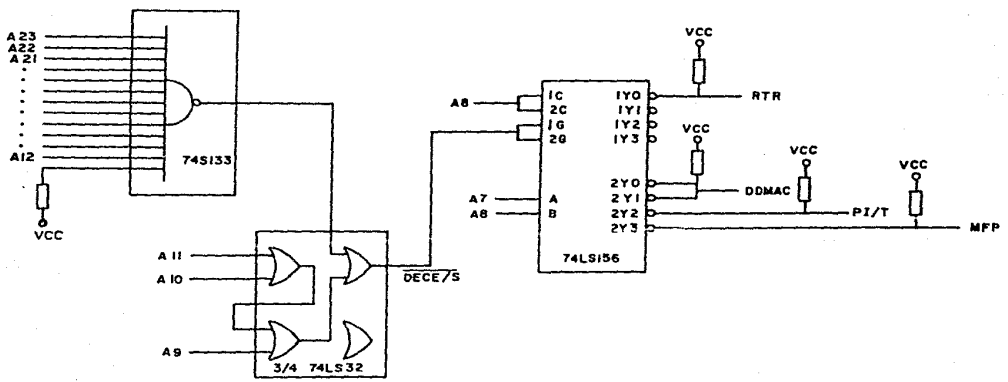
En el caso del PI/T y el RTR, ocurre lo mismo, sólo tienen canal de datos de 8 bits y se desperdician la misma cantidad de registros que contiene el puerto.

Para el DDMAC se necesitan 128 bytes y dado que éste cuenta con registro internos de 16 bits (utiliza las señales de LDS y UDS del procesador), se le asignó un espacio de direccionamiento de 128 bytes. Por último, se dejaron disponibles 192 bytes para el usuario.

Para llevar a cabo la decodificación se utilizaron compuertas OR y una NAND, además de un decodificador dual de 2 a 4 líneas 74LS156 que permite direccionar 64 bytes por línea de salida de acuerdo a la forma en que se conectó, (ya que es lo que requieren el PI/T y el MFP) y aprovechando la salida de colector abierto del decodificador, se OR alambraron dos salidas para direccionar los 128 bytes necesarios para el DDMAC; al RTR se le asignó una salida de 64 bytes, restando por último tres salidas de 64 bytes disponibles para el usuario (figura 3.9).

DECODIFICACION DE E/S

(FFF000 ⟵ FFF1FF)



FFF000	–	FFF07F	DDMAC
FFF080	–	FFF0BF	PI/T
FFF0C0	–	FFF0FF	MEP
FFF100	–	FFF13F	RTR
FFF140	–	FFF1FF	DISPONIBLE PARA EL USUARIO

Figura 3.9 Decodificación de E/S

3.4.2 DDMAC

Se cuenta con un controlador de acceso directo a memoria de dos canales (DDMAC), el cual tiene el propósito de transferir datos entre memoria y dispositivos, dispositivos y memoria, y de memoria a memoria, a altas velocidades, usualmente mucho más rápidas que el procesador. Esto permite al procesador realizar otras tareas mientras el DDMAC transfiere bloques de datos.

La operación del DDMAC se lleva a cabo en tres fases: fase de inicialización, fase de transferencia, y fase de terminación. En la fase de inicialización (modo de operación MPU), el procesador carga los registros del DDMAC con la información de control, y los apuntadores de dirección. Durante la fase de transferencia (modo de operación DMA) el DDMAC acepta peticiones para la transferencia de operandos y proporciona el direccionamiento y el control del bus para las transferencias. La fase de terminación ocurre después de completar la operación cuando el DDMAC reporta el estado de la operación. Si posteriormente no se requieren operaciones de transferencia, el DDMAC entra en el modo de operación de desocupado.

Existen dos modos de transferencia en la operación del DDMAC: transferencia de datos de direccionamiento implícito, y de direccionamiento explícito. En el direccionamiento implícito, los dispositivos no generan una dirección para la transferencia de datos. Las transferencias entre memoria y dispositivos son controladas por las líneas de petición y reconocimiento como se muestra en la figura 3.10a. Los dispositivos de direccionamiento explícito requieren de la dirección del dispositivo que se va a direccionar. Debido a que el bus de direcciones es utilizado para direccionar el periférico, el dato no puede ser transferido hacia/desde memoria, ya que ésta también requiere

3.4.2 DDMAC

Se cuenta con un controlador de acceso directo a memoria de dos canales (DDMAC), el cual tiene el propósito de transferir datos entre memoria y dispositivos, dispositivos y memoria, y de memoria a memoria, a altas velocidades, usualmente mucho más rápidas que el procesador. Esto permite al procesador realizar otras tareas mientras el DDMAC transfiere bloques de datos.

La operación del DDMAC se lleva a cabo en tres fases: fase de inicialización, fase de transferencia, y fase de terminación. En la fase de inicialización (modo de operación MPU), el procesador carga los registros del DDMAC con la información de control, y los apuntadores de dirección. Durante la fase de transferencia (modo de operación DMA) el DDMAC acepta peticiones para la transferencia de operandos y proporciona el direccionamiento y el control del bus para las transferencias. La fase de terminación ocurre después de completar la operación cuando el DDMAC reporta el estado de la operación. Si posteriormente no se requieren operaciones de transferencia, el DDMAC entra en el modo de operación de desocupado.

Existen dos modos de transferencia en la operación del DDMAC: transferencia de datos de direccionamiento implícito, y de direccionamiento explícito. En el direccionamiento implícito, los dispositivos no generan una dirección para la transferencia de datos. Las transferencias entre memoria y dispositivos son controladas por las líneas de petición y reconocimiento como se muestra en la figura 3.10a. Los dispositivos de direccionamiento explícito requieren de la dirección del dispositivo que se va a direccionar. Debido a que el bus de direcciones es utilizado para direccionar el periférico, el dato no puede ser transferido hacia/desde memoria, ya que ésta también requiere

direccionamiento. Por lo tanto el dato es transferido de la fuente a un registro interno en el DDMAC, y es transferido hacia/desde memoria durante una segunda transferencia de bus, como se muestra en la figura 3.10b.

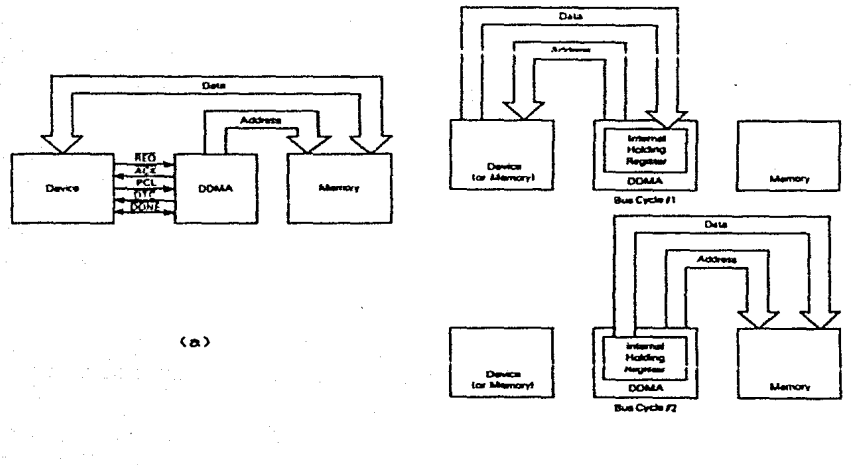


Figura 3.10 Modos de transferencia DDMAC

El DDMAC dependiendo de el tipo de transferencia, permite puertos de 8 o de 16 bits, ya que tiene un canal de datos de 16 bits.

Cuenta además con dos canales, y contiene 17 registros internos por canal, todos ellos completamente programables. Estos registros contienen información acerca de las tranferencias de datos, tales como: dirección fuente, dirección destino, número de transferencias, modo de transferencia, códigos de función, tamaño

del operando, tamaño del puerto, prioridad de canal, direccionamiento continuo, cuenta continua y uso de las líneas de control programable (PCL).

El DDMAC provee interrupciones vectorizadas al procesador cuando ocurren eventos tales como: la terminación de la operación del DDMAC, un error, o un requerimiento de un dispositivo periférico usando una línea PCL.

A cada canal se le puede asignar un nivel de prioridad de 0 o 1, siendo el 0 de más alta prioridad.

Las peticiones de DDMAC pueden ser generadas por un dispositivo, o internamente por el mecanismo de auto-petición del DDMAC. Las peticiones externas pueden ser peticiones en modo burst o en modo ciclo steal. El primer modo es utilizado por dispositivos que requieren de una alta velocidad de transferencia, y el segundo para dispositivos que generan una señal pulsante por cada operando a ser transferido. En el caso de las peticiones internas, éstas pueden ser generadas a máxima velocidad de 4Mbytes por segundo o a velocidades limitadas.

En nuestro diseño, un canal del DDMAC está dedicado a realizar el refresco de la DRAM, el cual se lleva a cabo cada 4mseg., refrescando de manera continua toda la memoria. El DDMAC trabaja en modo de operación de transferencia de dirección implicita, con generación de reconocimiento hacia el periférico. Se utiliza el canal cero, el cual tiene el nivel de más alta prioridad, con el propósito de que no se pierda la información de la DRAM en caso de que se estuviera utilizando el otro canal. La transferencia se realiza de memoria a dispositivo, para simular una lectura de la memoria y asi realizar el refresco, completando el ciclo escribiendo en un dispositivo imaginario. Los registros del DDMAC se cargan por medio del procesador durante la inicialización, proporcionándole la dirección de inicio de

refresco, número de direcciones a refrescar, canal a utilizar, así como el modo de transferencia. Una vez que se ha refrescado toda la memoria, se activa la línea de PCL0 para recargar nuevamente los registros y realizar de nuevo el refresco. Esto se lleva a cabo con un timer proveniente del MFP cada 4mseg. La línea que activa el refresco de la DRAM es la línea de ACK0 de el DDMAC, ya que se programó con generación de reconocimiento hacia el periférico, por lo que en cada ciclo de transferencia se activa esta línea y aparece la dirección de refresco, indicando así al circuito de control de refresco que se trata de un ciclo de refresco y no de lectura de la DRAM. En el diagrama de tiempo del ciclo de refresco se aprecia claramente el estado que tienen las líneas durante el refresco.

El DDMAC cuenta con tres líneas de entrada BCE0-BCE2 para detectar condiciones anormales de terminación de los ciclos de bus. De acuerdo a las especificaciones de el manual de Motorola para el MC68440, dependiendo de la manera en que se presenten las líneas de BERR y HALT, se debe tener en las líneas BCE0-BCE2 la siguiente decodificación:

BCE2	BCE1	BCE0	DEFINICION
H	H	H	No excepción
H	H	L	Detenido
H	L	H	Error de bus
H	L	L	Reintentar
L	H	H	Abandonar y Reintentar
L	H	L	Reservado
L	L	H	Reservado
L	L	L	Reinicializar

El circuito que propone Motorola para lograr esta decodificación es el mostrado en la figura 3.11.

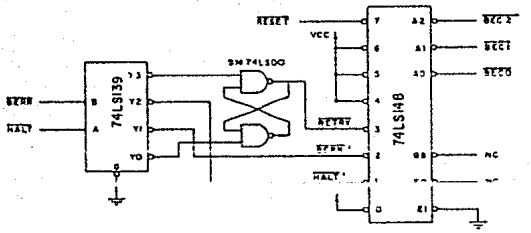


Figura 3.11 Generador de BEC para DDMAC

Por otro lado, debido a que las líneas de direcciones se encuentran multiplexadas con las de datos, se deben demultiplexar; el circuito de demultiplexaje lógico se muestra en la figura 3.12.

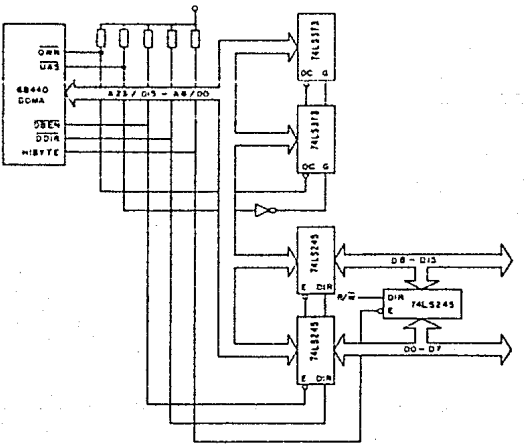


Figura 3.12 Demultiplexaje Lógico DDMAC

DDMAC (68440)

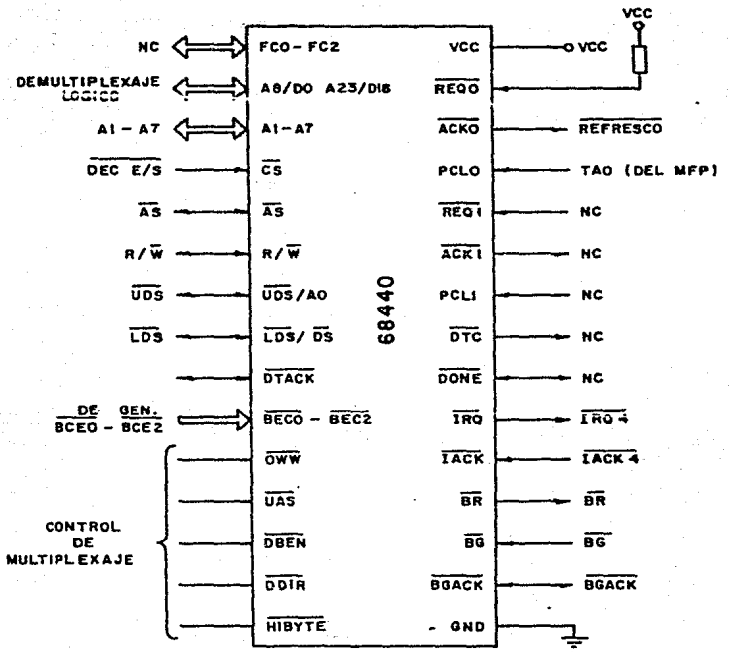


Figura 3.13 DDMAC

Finalmente la figura 3.13 de la página anterior muestra el alambrado completo del DDMAC.

3.4.3 MFP

El periférico multifunción MC68901 contiene 4 temporizadores, un puerto serie (USART) y un manejador de interrupciones de 8 niveles. Dos de los temporizadores y un cristal externo en conjunto con el USART conforman un puerto serie RS232 de velocidad programable para recepción y transmisión independientemente. Otro de los temporizadores está dedicado a generar la señal que ordena al DMAC realizar el refresco de RAM cada 4 milisegundos. Siete de las 8 líneas E/S son usadas como entradas de solicitudes de interrupción. El MFP se encarga de prioritizar estas líneas y las interrupciones internas de los temporizadores y el USART, y proporciona el vector de interrupción correspondiente (que es programable en sus 4 bits más significativos) durante un ciclo de reconocimiento de interrupción del CPU. El bit 5 de las líneas de E/S se usa como salida para manejar un LED que indique la ejecución y el resultado de una rutina de autoprueba que corre al encender o reinicializar el sistema.

Es necesario utilizar un flip/flop para dividir la frecuencia del reloj entre dos, ya que la frecuencia máxima de operación del MFP es 4 MHz (figura 3.14).

3.4.4 RTR

El circuito MM58174A cuenta con registros independientes para hora-fecha desde décimas de segundo hasta decenas de meses y día de la semana. La base de tiempo se genera a partir de un oscilador controlado por cristal de 32 KHz (figura 3.15). Puede

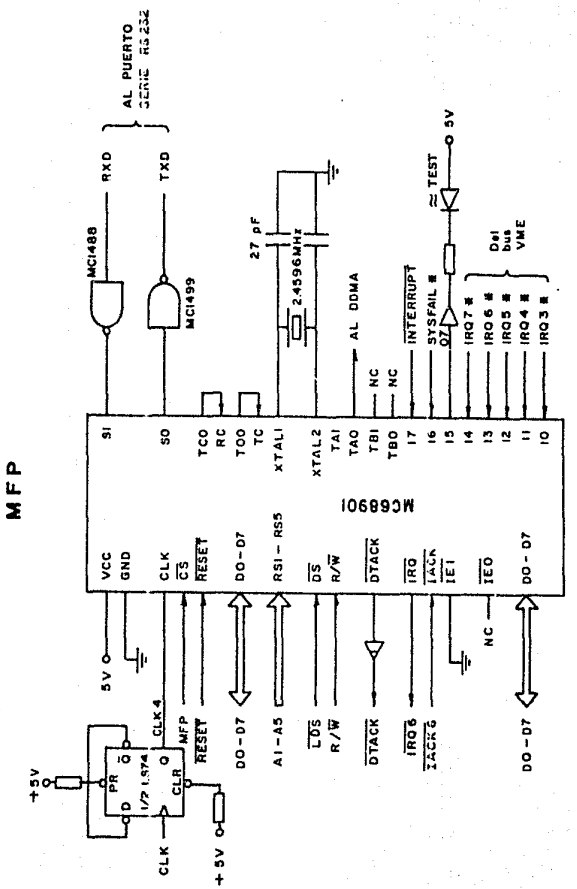


Figura 4.14 MFP

RELOJ DE TIEMPO REAL (MM58174A)

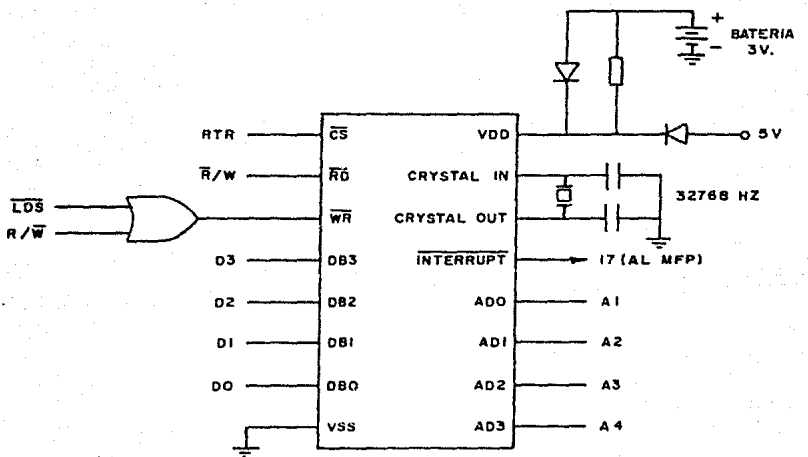


Figura 3.15 RTR

generar interrupciones sencillas o repetitivas cada 60, 5 o 0.5 segundos. Sólo utiliza los 4 bits menos significativos del canal de datos, con la ventaja de que ocupa muy poco espacio de la tarjeta, pues sólo tiene 16 patas.

3.4.5 PI/T

El circuito de interfaz paralela/temporizador MC68230 contiene dos puertos de 8 bits de propósito general con 4 líneas programables para diferentes protocolos, 8 líneas que pueden usarse como E/S o para funciones alternativas, un temporizador de 24 bits con preescalador de 5 bits y 5 vectores de interrupción separados.

El puerto A y dos líneas del C estan configurados para trabajar como interfaz paralela Centronics. La línea PC2 del puerto C permite introducir una señal externa de reloj para el temporizador. Las interrupciones de los puertos y del temporizador son manejadas con dos líneas diferentes. Estas líneas tienen salida de colector abierto, por lo que requieren de una resistencia de "pull-up" (figura 3.16).

3.5 GENERADORES DE DTACK

Los circuitos generadores de DTACK fueron incluídos para optimizar la duración de los ciclos de lectura y escritura de RAM y ROM y de acceso al reloj de tiempo real.

Se utilizan circuitos flip/flop D cuádruples para retrasar cada ciclo desde uno hasta cuatro ciclos de reloj, seleccionando mediante un puente el f/f del que se desea tomar la señal retrasada (ver figura 3.17). De esta manera es posible evitar el desperdicio de ciclos de reloj si en algún momento se utilizan dispositivos de memoria con tiempos de acceso diferentes a los utilizados inicialmente. Para el RTR el retraso es fijo de 4 ciclos de reloj.

PI / T

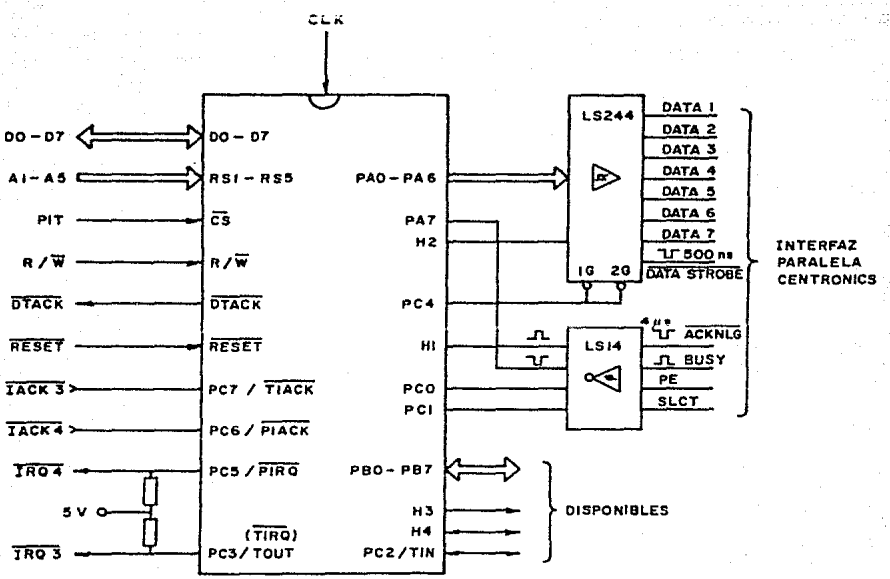


Figura 3.16 PI/T

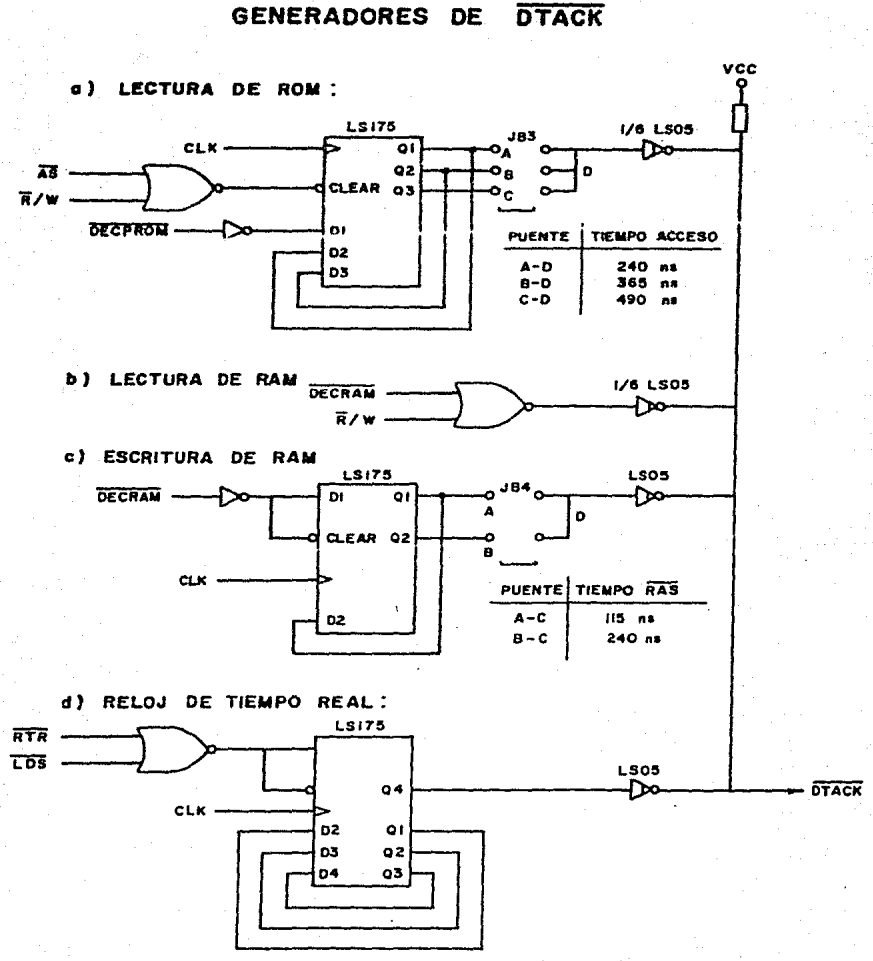


Figura 3.17 Generadores de DTACK

3.6 CONTROL DE INTERRUPCIONES

El microprocesador 68010 puede responder a 7 niveles de interrupción, los cuales tienen una prioridad decodificada. Existen dos métodos de interrupción vectorizada: el primero es el autovector, el cual tiene una localidad de memoria predefinida y en el segundo el dispositivo que solicita la interrupción proporciona la dirección de el vector de interrupción en 8 bits sobre el bus de datos.

Los 7 niveles de interrupción se encuentran asignados en el siguiente orden de prioridades: La interrupción no enmascarable (nivel 7) genera una interrupción autovectorizada a la localidad de memoria 7CH. Esta interrupción ocurre cuando se presenta el ACFAIL. En el nivel 6 se encuentra una petición de interrupción proveniente de el MFP, el nivel 5 corresponde a el DDMAC, los niveles 4 y 3 estan asignados a las salidas PC5/PIRQ y PC3/TOUT de el PI/T y por último los niveles 2 y 1 están a disposición de IRQ3 y IRQ2 de el bus VME, y son autovectorizados. El nivel cero indica que no existe petición de interrupción.

El manejador de petición de interrupciones se encarga de codificar las peticiones de interrupción, asegurando que sólo codificará la línea que tenga el más alto nivel de prioridad.

Por otra parte, el manejador de reconocimiento de interrupciones codifica las líneas A1 A2 y A3 de el procesador, las cuales proporcionan el nivel de reconocimiento de interrupción, este nivel es válido sólo cuando las líneas FC0 FC1 y FC2 se encuentran en un uno lógico y cuando AS es activa, por lo que las tres anteriores se encuentran alambradas a una NAND y la salida de ésta a una OR junto con el AS, de tal forma que cuando son válidas, se habilita el decodificador dual de 2 a 4 líneas. Se seleccionó el 74LS156 para aprovechar la salida de colector abierto, y OR alambrar las salidas de IACK7 IACK2 y IACK1. En la figura 3.18 de la página anterior se ilustran el

MANEJADOR DE INTERRUPCIONES

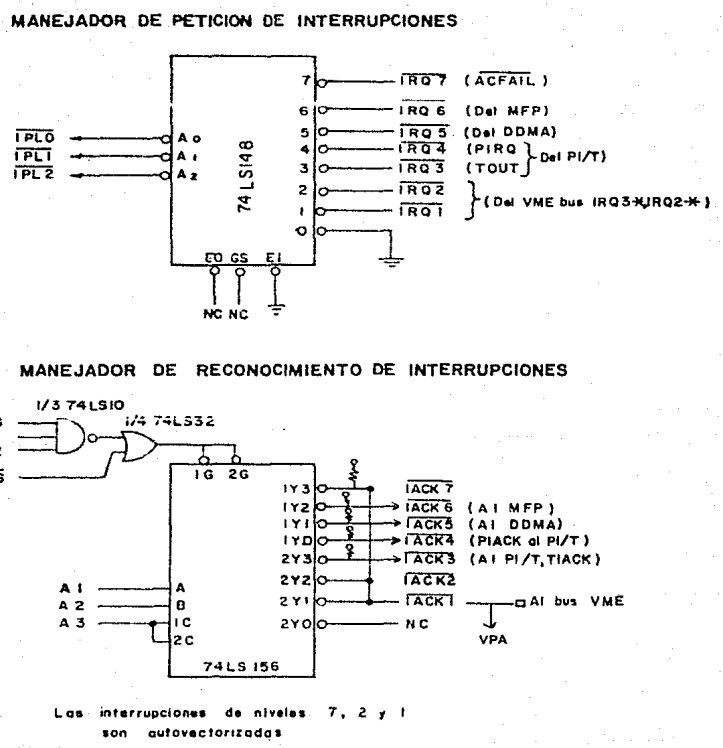


Figura 3.18 Manejo de interrupciones

manejador de peticiones y el manejador de reconocimiento de interrupciones.

3.7 Interfaz a VME

Los circuitos de interfaz al bus VME se seleccionaron en función de las especificaciones eléctricas requeridas que define el propio bus VME. Todos los circuitos buffers (exepto los de las líneas de control y falla del bus) son activados con la señal MSTR (Master) que indica que le ha sido otorgado al sistema el control del bus (figura 3.19).

3.7.1 Módulo Solicitante

Este circuito genera una petición del bus VME sobre la línea BRQ3* cuando durante un ciclo de acceso al canal del sistema los circuitos decodificadores NO reconocieron ninguna de las direcciones de memoria o puertos en la tarjeta. En otras palabras, cualquier intento de acceso a una dirección que no corresponda con los espacios de direcciones asignados a los dispositivos residentes en esta tarjeta, generará una solicitud de acceso al bus VME. Esta característica cumple con el mapa de memoria definido en el punto 3.3.1.

3.7.2 Arbitro del bus VME

Se escogió la opción ONE para el arbitraje del bus VME por ser ésta la de implementación más sencilla, pues sólo atiende a solicitudes de control del bus que se apliquen a la línea BRQ3* y concede este control en cuanto el maestro anterior lo libera (figura 3.20).

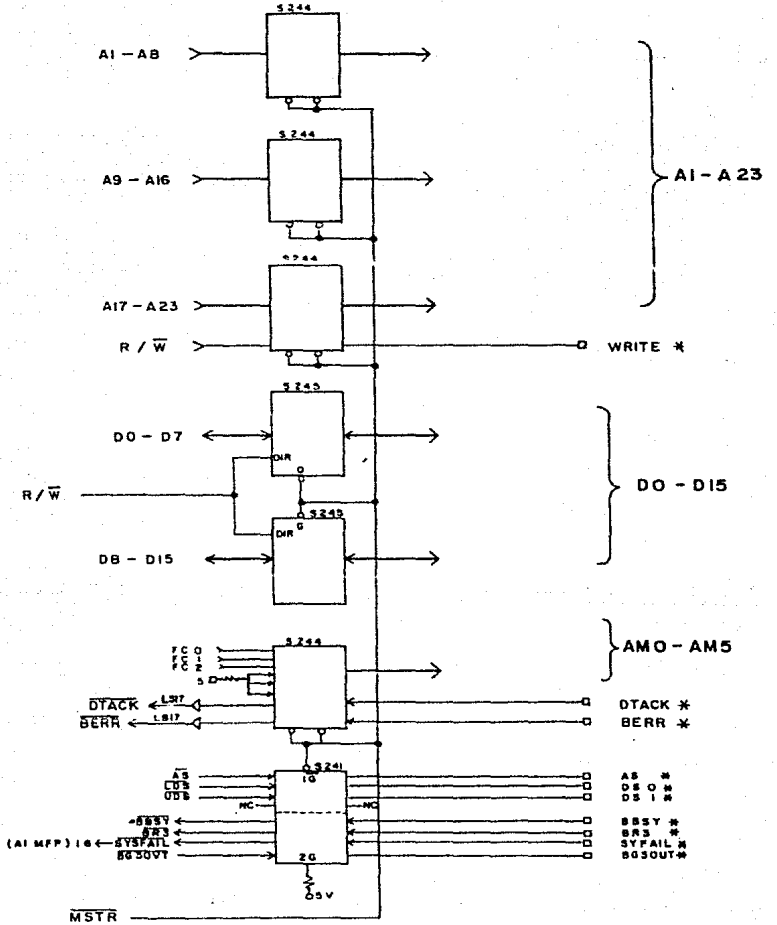


Figura 3.19 Interfaz a VME

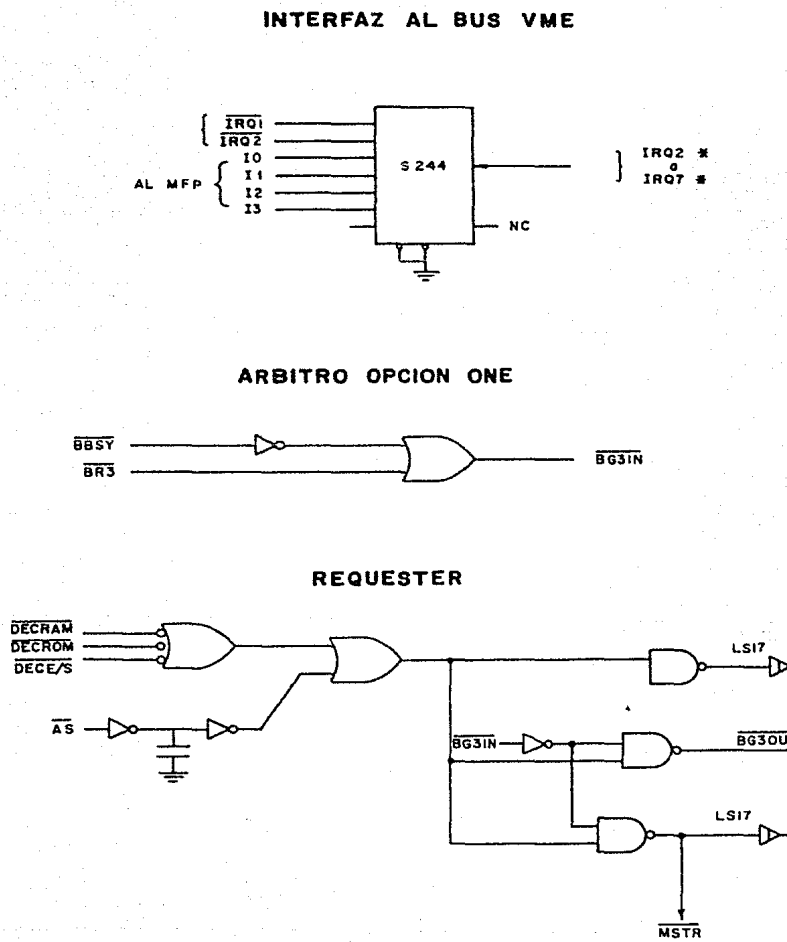


Figura 3.20 Interfaz a VME (solicitante y arbitro)

CAPITULO IV

APLICACIONES

CAPITULO IV

APLICACIONES

4.1 INTRODUCCION

En el presente capítulo se expondrán los problemas que se tienen al establecer una comunicación entre procesos, y la manera de resolverlos en la práctica. También se profundizará sobre el tema de tolerancia a fallas, explicando algunos de los métodos comunmente utilizados para soportar una falla.

Se explicará el concepto de memoria virtual y las ventajas que representa el uso de la misma. Por último se describirá una aplicación en robótica de los conceptos, arquitecturas y diseños propuestos en el desarrollo del presente trabajo.

4.2 COMUNICACION ENTRE PROCESOS

Los procesos dentro de un sistema de cómputo no actúan aisladamente. Por un lado deben cooperar para lograr el objetivo deseado y por el otro compiten entre sí por el uso de los recursos limitados como procesadores, memoria o archivos.

Esta cooperación y competencia implica la necesidad de una forma de comunicación entre los procesos. Se pueden distinguir tres áreas en las que esta comunicación es esencial:

-Exclusión mutua. Los recursos del sistema pueden clasificarse como "compartibles", cuando pueden ser utilizados por varios procesos concurrentemente (por ejemplo CPUs, discos, archivos sólo de lectura y áreas de memoria que contienen datos protegidos contra modificación), o "no compartible", cuando debe restringirse su uso a un proceso a la vez (por ejemplo: impresora, lectora de tarjetas, archivos de escritura y áreas de datos sujetas a modificación). La no compartibilidad de un recurso puede deberse a que su naturaleza física lo hace impráctico de compartir (impresora, por ejemplo), o a que si lo usaran diferentes procesos concurrentemente, las acciones de un proceso podrían interferir con las de otro, por ejemplo, si un proceso lee una variable mientras otro la está modificando el resultado sería impredecible. La exclusión mutua debe asegurar que los recursos no compartibles sean usados por un sólo proceso a la vez.

-Sincronización. En general la velocidad de un proceso relativa a otro es impredecible, es decir, los procesos corren de manera asíncrona. Sin embargo, para lograr la cooperación existen ciertos puntos en los que los procesos deben sincronizarse antes de poder continuar su actividad. Deben proveerse mecanismos para que la sincronización pueda efectuarse.

-Abrazo mortal (deadlock). Cuando varios procesos compiten por los recursos es posible que se dé una situación en la que ningún proceso pueda continuar porque los recursos que requiere cada uno están siendo utilizados por otro; en otras palabras, los procesos esperan por una condición o evento que no habrá de cumplirse. A esta situación se le llama "abrazo mortal" y obviamente debe evitarse, pues su ocurrencia puede tener consecuencias graves

sobre todo cuando se están controlando sistemas en tiempo real.

4.2.1 SEMAFOROS

La solución más ampliamente utilizada en los problemas expuestos anteriormente son los semáforos, concepto introducido por Dijkstra en 1965 [11] junto con las operaciones P y V que actúan sobre ellos. La idea básica de la solución de Dijkstra es la suspensión de los procesos que no puedan utilizar el recurso que necesitan, y su reactivación cuando algún proceso desocupa ese recurso, abriendo la posibilidad de aprovechar el tiempo de procesador en otras actividades. Un semáforo es un valor entero no negativo contenido en una localidad de memoria que todos los procesadores pueden accesar, pero no concurrentemente, y que excepto durante la inicialización, sólo puede ser afectado por las operaciones P o V que se definen como sigue:

V(s) Su efecto es incrementar el valor del semáforo s en uno de manera indivisible, es decir, sin que nadie pueda afectar el valor de s durante esta operación.

P(s) Su efecto es decrementar el valor del semáforo en uno siempre que el resultado no resulte negativo. Cuando ocurre esto, es decir, si se aplica a un semáforo que vale cero, el proceso ejecutando la operación no puede continuar hasta que algún otro proceso incremente el valor del semáforo en uno con una operación V. La operación P también es indivisible lo que significa que si varios procesos están esperando por el semáforo, sólo uno de ellos podrá completar su operación P cuando el semáforo se haga positivo, aunque no se supone nada sobre el método para elegir al proceso "ganador".

A continuación se describirá brevemente el uso de los semáforos para resolver los problemas de comunicación entre procesos.

-Exclusión mutua. Los recursos no compartibles pueden protegerse contra el acceso de varios procesos a la vez impidiendo a dichos procesos la ejecución concurrente de las partes del programa en las que se hace el acceso. Esas partes de programa son llamadas "secciones críticas". La exclusión mutua en el uso de los recursos puede verse como exclusión mutua de secciones críticas.

La exclusión se logra delimitando cada sección crítica con operaciones P y V sobre un solo semáforo cuyo valor inicial es unitario. Cada vez que un proceso desee hacer acceso a una variable o recurso compartido deberá efectuarlo vía una sección crítica, la que a su vez está protegida por un semáforo. Otra condición para resolver correctamente el problema de exclusión mutua con semáforos es no usar las operaciones P y V para otros propósitos.

-Sincronización. La forma más simple de sincronización es que un proceso A no pueda proceder más allá de un punto L1 hasta que otro proceso B haya alcanzado un punto L2. Esta situación surge por ejemplo cuando A requiere información en el punto L1 que es proporcionada por B al llegar a L2. La sincronización puede programarse como sigue:

```
Programa del proceso A          Programa del proceso B
          .                               .
          .                               .
   L1: P(procede);                 L2: V(procede);
```

donde "procede" es un semáforo con valor inicial cero.

Es claro que A no puede continuar más alla de L1 si B no ha ejecutado la operación V en L2.

Un ejemplo clásico de dos procesos en el que cada uno regula el progreso del otro es el problema del productor y el consumidor (ver ref. [14]).

-Abrazo mortal. Como se señaló antes, un abrazo mortal puede ocurrir siempre que un conjunto de procesos compitan por los recursos. También puede ocurrir cuando los procesos estén esperando que otro proceso complete ciertas acciones como por ejemplo la ejecución de una operación V. De hecho, esta situación es análoga al abrazo mortal que surge de la competencia por recursos si se considera a un semáforo como recurso, y las operaciones P y V como el pedir y soltar este recurso. La "compartibilidad" de un semáforo está determinada por su valor inicial: si el valor es n>1 el semáforo puede ser compartido por n procesos; si es 1 entonces el semáforo no es compartible.

Debe hacerse notar que un abrazo mortal puede ocurrir como resultado de una secuencia de operaciones P ordenada en forma incorrecta aún cuando no se esté haciendo una petición explícita de un recurso. Este problema sólo se puede evitar realizando un cuidadoso análisis de la programación de los semáforos y comprobar que no se haya omitido ninguna operación P y V necesaria y que estas estén en los lugares correctos.

4.2.2 REALIZACION DE SEMAFOROS

Para proponer una implementación de las operaciones P y V es necesario precisar un poco más que es lo que en realidad deben

hacer estas funciones cuya definición original es:

V(S): Si S>0, decrementa S.

P(S): Incrementa S

donde S es cualquier semáforo.

A continuación se describen los varios aspectos que involucra la implementación de semáforos en un medio ambiente de multiprocesamiento en general y para el caso particular que se propone en este trabajo.

-Bloqueo y desbloqueo.

La operación V implica que los procesos son bloqueados cuando un semáforo tiene un valor cero, y liberados o desbloqueados cuando una operación P incrementa este valor a uno. La forma natural de implementar esto es asociando a cada semáforo una "cola del semáforo". Cuando un proceso realiza una operación V y no tiene éxito (es decir, opera sobre un semáforo con valor cero) es agregado a la cola del semáforo y se suspende su ejecución. Así mismo, cuando una operación P se realiza sobre un semáforo, algún proceso puede sacarse de la cola (a menos que ésta esté vacía) y reanudar su ejecución. El semáforo debe por lo tanto ser implementado con dos componentes: un número entero y un apuntador de cola.

De esta manera, la implementación queda como sigue:

V(S): Si S>0, entonces S=S-1
si no, agregar el proceso a la cola de S y suspenderlo.

P(S): Si cola vacía, entonces S=S+1
si no, quitar algún proceso de la cola de S
y reanudar su ejecución.

Cabe hacer notar que el semáforo no necesita ser incrementado con la operación P si un proceso es liberado ya que dicho proceso inmediatamente tendría que decrementar el semáforo otra vez para completar su operación V.

-Organización de la cola del semáforo.

Todavía no se ha dicho nada acerca de que proceso será el afortunado en ser liberado de la cola del semáforo después de una operación P, ni se ha establecido si un proceso debe agregarse al principio, al final, o en algún lugar intermedio de la cola en una operación V no exitosa.

Para la mayoría de los semáforos, una sencilla cola de tipo FIFO (primero en entrar primero en salir) es adecuada, ya que asegura que todos los procesos bloqueados serán liberados eventualmente. En algunos casos puede ser preferible ordenar la cola con otros criterios, tales como la cantidad de recursos requeridos, el tiempo desde que el proceso se activó por última vez, y la importancia relativa del usuario, por ejemplo. Esto asegura que procesos con alta prioridad no esperarán largos periodos de tiempo en una cola de semáforo.

Como semáforos diferentes podrían requerir diferentes organizaciones de colas, en ese caso deberá incluirse un componente extra en la implementación del semáforo para indicar que organización de cola se aplica. Este componente puede ser

simplemente una descripción codificada del tipo de organización, o, en casos más complejos, podría ser un apuntador a una subrutina que realice las operaciones correspondientes con la cola.

-Indivisibilidad.

Como ya se dijo, las operaciones P y V deben ser indivisibles en el sentido de que únicamente se permitirá a un proceso a la vez ejecutarlas en un instante dado [14]. Si llegara a suceder que dos operaciones P o V tengan su turno de ejecución al mismo tiempo, alguna de ellas esperará hasta que la otra termine de ejecutarse.

Para realizar esto, ambas operaciones deben ser implementadas como procedimientos que comiencen con algún tipo de operación de "Lock" (cerrar la cerradura) y que terminen con una operación de "Unlock" (abrir la cerradura).

En un sistema de procesador único, la operación Lock puede implementarse muy fácilmente deshabilitando el mecanismo de interrupciones. De esta manera se asegura que un proceso no puede perder el control del procesador mientras ejecuta P o V ya que no hay forma de que pueda ser interrumpido. La operación Unlock se realiza simplemente rehabilitando las interrupciones.

En una máquina con varios procesadores el procedimiento anterior no es adecuado ya que cabe la posibilidad de que dos procesos ejecuten simultáneamente P o V corriendo en diferentes procesadores, por lo que es necesario otro mecanismo más seguro.

La manera más común de resolver el problema en estos casos

es una operación "Test and Set". Esta consiste en una instrucción que prueba y modifica el contenido de una localidad de memoria en una sola operación. Durante la ejecución de la instrucción se inhiben los intentos de otros procesadores de accesar la localidad.

Precisamente con esta idea, el juego de instrucciones del microprocesador MC68010 incluye la instrucción TAS sobre un operando de un byte, cuyo funcionamiento es el siguiente: se prueba el operando, se ponen las banderas N y Z (negativo y cero) del código de condiciones según corresponda y se escribe un 1 en el bit más significativo del operando. Todo esto se realiza de manera indivisible usando un ciclo de lectura-modifica-escritura.

Físicamente, lo que se hace en este ciclo es mantener activa (nivel bajo) la señal de control de direcciones AS* contínuamente durante los ciclos de lectura y escritura que componen a la instrucción, lo cual impide por hardware que cualquier otro dispositivo haga uso del bus.

Por otro lado, las especificaciones del bus VME también definen un ciclo de lectura-modifica-escritura que funciona de la misma forma, proporcionando de manera sencilla y segura un ciclo indivisible, ya que el control del bus de transferencia de datos (DTB) sólo puede ser transferido mientras AS* es inactiva (alta).

De esta manera tenemos para nuestro caso particular todos los elementos para implementar semáforos aplicando como operación Lock la instrucción TAS sobre una localidad de la memoria compartida por los procesadores, la cual funciona como una bandera que indica si se permite o no realizar los procedimientos P y V sobre un semáforo particular (para cada semáforo existe una bandera). La idea es efectuar la instrucción TAS sobre la bandera y si es negativa (bit más significativo en 1) entonces el

semáforo está en uso, por lo que el procesador deberá repetir la prueba hasta que el semáforo esté disponible. Esta técnica se conoce como espera activa (busy waiting). Si la bandera es positiva, el proceso continuará con su ejecución. Al dejar de usar el semáforo cada procesador tiene la responsabilidad de borrar la bandera para permitir la entrada de otros procesos, lo que corresponde a terminar con la operación Unlock, como se mencionó antes.

Una alternativa a la instrucción Test and Set para la implementación de la operación Lock, utilizada en los microprocesadores de la familia 8088 y 8086 [13] es una instrucción para intercambiar los contenidos de dos operandos (ya sean registros internos o memoria). La implementación consiste en intercambiar los valores de la bandera y un registro que previamente se había puesto en cero (por ejemplo). Después, el valor del registro se examina para determinar si el acceso al semáforo es permitido en ese momento o no, y mientras, cualquier otro proceso que intente entrar se encontrará con el cero dejado por el intercambio, indicándole que no puede hacer uso del semáforo hasta que la bandera sea modificada por otro proceso que ejecute una operación de Unlock.

La indivisibilidad se logra en este caso con el prefijo de instrucción LOCK, que activa una línea externa del procesador durante la ejecución de la instrucción que sigue al prefijo. Esta señal indica a los demás dispositivos del sistema que no pueden tomar el control del bus hasta que dicha instrucción se complete.

Debe decirse que las operaciones Lock y Unlock no pueden ser usadas como substituto para P y V, tanto por las diferentes cosas que tiene que hacer cada una como por el tiempo que se llevan en hacerlo. La tabla siguiente ilustra los niveles conceptualmente

diferentes en los que los dos tipos de operaciones existen[11]:

	P y V	Lock y Unlock
Propósito	Sincronización general de procesos	Exclusión mutua de procesos para P y V
Nivel de implementación	Software	Hardware
Mecanismo de espera	Colas	Espera activa/ deshabilitación de interrupciones
Tiempo de espera	Varios segundos	Varios microsegundos

4.3 TOLERANCIA A FALLAS

En el capítulo I se habló de las técnicas de la tolerancia de fallas , y se mencionaron las tres formas básicas de la redundancia de hardware. A continuación se explicaran algunos de los métodos más utilizados para tolerar una falla, haciendo uso de la técnica de redundancia de hardware.

-Método Pasivo:

Un método pasivo de redundancia de hardware es la triple redundancia modular (TMR), cuyo propósito es enmascarar la falla. Esto se logra triplicando el hardware y eligiendo el resultado correcto por medio de la mayoría de votos, lo cual se lleva a cabo gracias a un dispositivo llamado "voter". El uso de la TMR

con un voter ideal se ilustra en la figura 4.1.

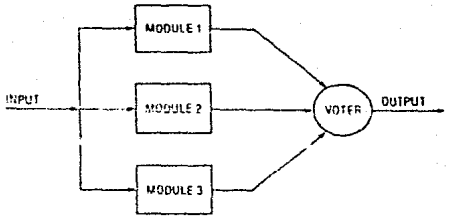


Figura 4.1

Debido a que los voters ideales son imposibles de construir, una configuración más confiable se muestra en la figura 4.2 en donde todos los canales de datos son triplicados. Sin embargo, se debe llegar a un solo resultado. En algunas aplicaciones este resultado es elegido por un voting de alta confiabilidad. Como se puede apreciar en la figura 4.2, cada voter examina todos los resultados, generando a su juicio una sola elección. Esta técnica intenta prevenir que los canales de datos sean alterados por una unidad de datos malfuncionante.

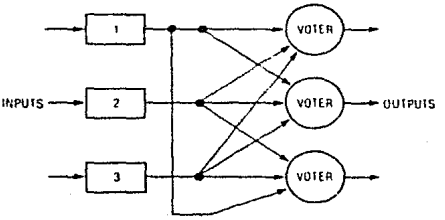


Figura 4.2

Otra manera de elegir el resultado, es por software. De esta forma, las tres computadoras realizan idénticas tareas, pasándose los resultados sobre un mismo canal de comunicaciones y votando por el resultado por medio de un programa. Un procesador produce datos y llama al sistema operativo para almacenar los dato en un buffer y mandarlos a los otros procesadores. Para adquirir los datos de entrada, el procesador llama a la subrutina voter, la cual obtiene los datos de los buffers de entrada, esperando si es necesario a que éstos se llenen. Una vez obtenidos los tres valores (A,B,C) se procede a votar por el resultado mediante el programa que se muestra a continuación:

```
      if (A equals B) then A
 else if (A equals C) then A
 else if (B equals C) then B
```

El sistema operativo también se puede encargar de determinar en caso de un desacuerdo, cuál es el módulo que está fallando.

-Método Activo:

El siguiente método es el de redundancia activa, que intenta incorporar la recuperación de la falla en el sistema, eliminando la capacidad de tolerar la falla. Un ejemplo es el simple esquema de duplicación que compara los resultados de dos sistemas y genera un mensaje de error si existieran desacuerdos. Suponiendo que éste fuera el caso, el sistema reporta el error, pero no lo corrige. El concepto de duplicación se ilustra en la figura 4.3.

Una segunda técnica de la réplica activa es el reemplazo de módulos en estados de espera. En esta configuración una unidad está operando, mientras una o más unidades permanecen en estado de espera. Varios esquemas de detección de errores se utilizan para determinar el momento en el que falla la unidad en línea y

reemplazarla por la unidad libre en estado de espera.

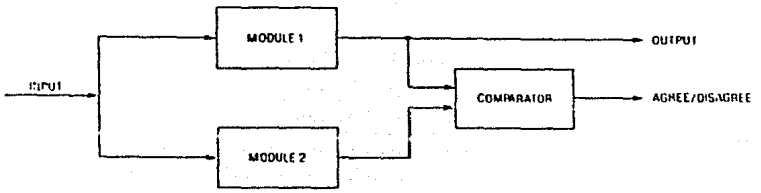


Figura 4.3

Esta técnica ofrece una completa capacidad operacional después de que una falla se ha presentado, pero una interrupción puede ser necesaria mientras se reemplaza la unidad libre. Si esta interrupción no puede ser tolerada en el proceso, se hace uso de una unidad en estado de actividad, la cual está sincronizada con la unidad en línea y preparada para ocupar su lugar en el momento en que ésta falle. Este concepto se ilustra en la figura 4.4.

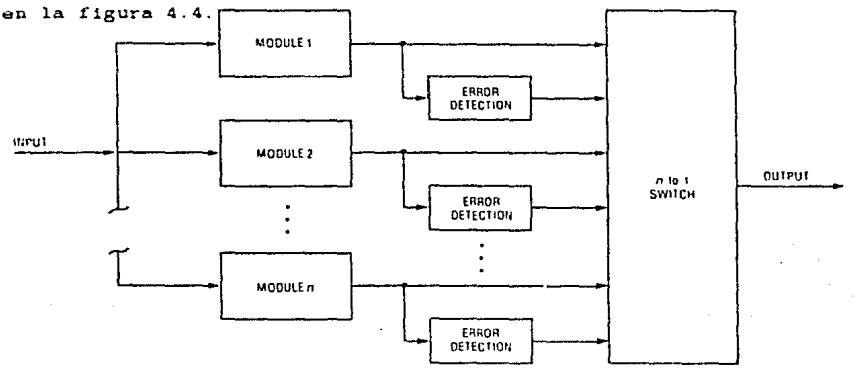


Figura 4.4

La tercera y última técnica de réplica activa combina el método de duplicación y el método de reemplazamiento de módulos en estado de espera. En este método dos unidades realizan las mismas operaciones, para posteriormente comparar sus resultados. En el caso de que ocurra alguna discrepancia entre las dos unidades, un módulo libre en estado de espera es activado. Normalmente, se aplican procedimientos de detección de errores para determinar cual de las dos unidades en línea está fallando, con la finalidad de sustituirla para tener siempre dos unidades operacionales funcionando correctamente. La figura 4.5 ilustra este concepto.

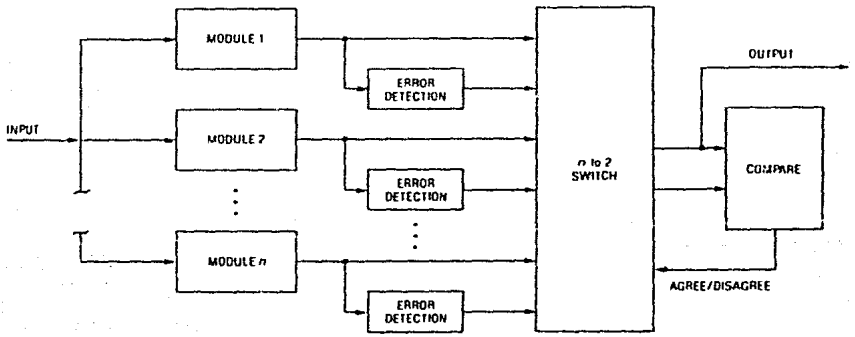


Figura 4.5

-Método Híbrido:

La combinación de las características más importantes de los métodos activos y pasivos se conoce como el concepto básico de técnicas de réplica híbrida. Esta técnica proporciona la detección y localización de una falla, además de tener la capacidad de poder reemplazar el módulo mal funcionante. Uno de

los más importantes métodos de réplica híbrida es la redundancia N-modular (NRM) con módulos libres. La NRM es una generalización de la triple redundancia modular, ya que utiliza N módulos para que el voter elija un resultado en vez de utilizar sólo tres. El concepto de la NRM se muestra en la figura 4.6, la cual muestra los N elementos modulares y los M módulos libres. El sistema permanece en la configuración básica de NRM hasta que el detector de desacuerdos determina que una unidad está fallando. Una vez determinada la unidad mal funcionante se desconecta y se reemplaza con una módulo libre. La confiabilidad del sistema se mantiene mientras se tengan módulos libres disponibles. La elección de el resultado se escoge de las unidades activas, enmascarando las fallas y asegurando continuidad de operaciones libres de errores. La figura 4.6 muestra esta idea.

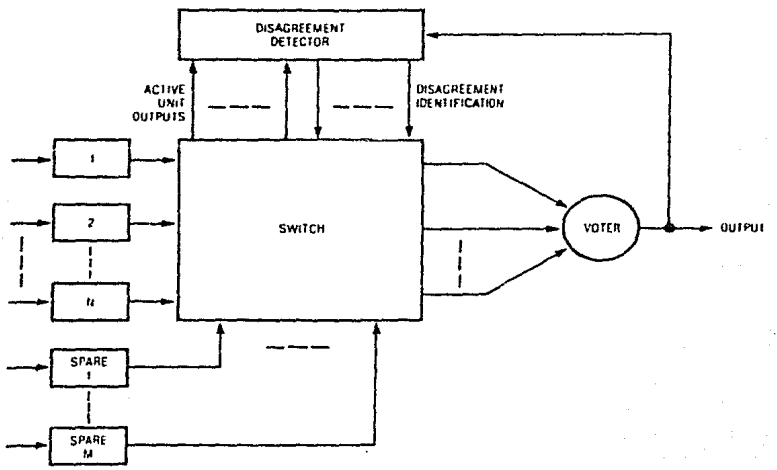


Figura 4.6

Un sistema con TRM se muestra en la figura 4.7 en donde el voter se encuentra conectado a los tres procesadores. Un circuito voter de un bit consiste de tres compuertas AND y una compuerta OR, conectadas como se ilustra en la figura 4.8.

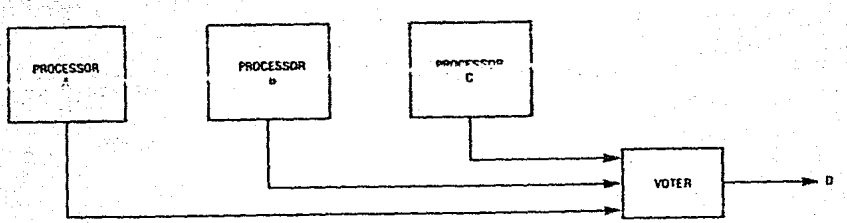
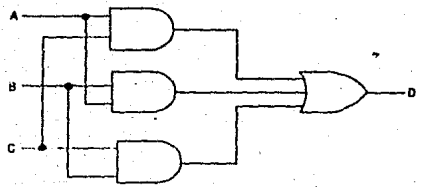


Figura 4.7

El mismo circuito se replica tantas veces como bits contenga el bus de datos. La tabla de la figura 4.8 muestra la forma en la que se elije el resultado por mayoría de votos, cuando dos de las tres entradas (A,B,C) tienen el valor de uno , una de las tres compuertas AND tendrá un uno a la salida, ocasionando que la compuerta OR entregue como resultado un uno. Si dos entradas son cero, ninguna compuerta AND entregará un uno, consecuentemente la compuerta OR entregará un cero. Por lo tanto la salida de el circuito será uno si dos o más entradas son uno, y será cero si dos o más entradas son cero. En otras palabras, la salida de el circuito corresponde a la mayoría de votos de la entrada.



A	B	C	D
0	0	0	0
1	0	0	0
0	1	0	0
0	0	1	0
1	1	0	1
1	0	1	1
0	1	1	1
1	1	1	1

Figura 4.8

Todos los métodos de tolerancia a fallas mencionados anteriormente se pueden implementar sobre la arquitectura VME, ya que ésta permite que le sean conectados cuantos módulos se deseen.

En el caso del método pasivo de la TMR se pueden conectar las tres tarjetas sobre el bus, y mediante software se vota por el resultado, enmascarando de esta manera la falla.

De igual forma se puede implementar el método activo, en el que se conectan dos tarjetas al bus, comparando los resultados por software y enviando un mensaje en el caso de que existiera un desacuerdo, sin corregir el error.

Por último, dada la capacidad del bus VME de soportar cuantas tarjetas se le deseen conectar, se puede implementar el método híbrido, teniendo N elementos modulares para elegir el resultado y M módulos libres. En este método tambien se vota por medio de la programación, y si algún módulo no coincide con el resultado de la mayoría, se localiza por medio de software y a través de el bus serial VMS se desconecta el módulo malfuncionante y se reemplaza por un módulo libre.

4.4 MEMORIA VIRTUAL

Hace algunos años, la introducción de microprocesadores de 16 bits señalaron una nueva generación de microprocesadores, los cuales eran capaces de realizar algunas de las funciones que estaban sólo disponibles en minicoputadoras y computadoras de gran capacidad. Sin embargo muchos de esos microprocesadores no tenían un manejo sencillo de sus nuevos atributos. Hoy en día se cuenta con microprocesadores capaces de direccionar gran cantidad

de memoria, así como el soporte de memoria virtual, proporcionando todos los mecanismos necesarios para su implementación, los que sólo se podían encontrar en computadoras de gran tamaño.

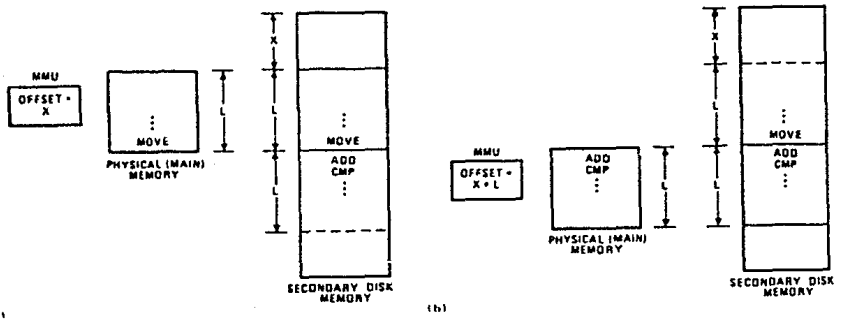
En un sistema con memoria virtual, un programa de usario puede ser escrito, pensando en que se tiene una gran cantidad de memoria disponible, cuando en realidad se cuenta sólo con una pequeña cantidad de memoria física presente en el sistema.

El objetivo de la memoria virtual es el de proveer a un sistema de los mecanismos necesarios para tener una limitada cantidad de memoria física de alta velocidad que pueda ser accesada directamente por el procesador, mientras se mantiene una imagen de gran cantidad de memoria virtual en un dispositivo de almacenamiento secundario como son las cintas, los discos duros y discos floppy. De esta manera se puede diseñar un sistema con gran capacidad de almacenamiento, teniendo una pequeña cantidad de memoria física de alta velocidad que resulta ser muy costosa, y dispositivos de almacenamiento masivo que son relativamente menos costos por byte que la memoria pero más lentos, con el objeto de que los programas residentes en memoria física corran rápidamente, frenando la ejecución únicamente en caso de que se requiera información del dispositivo secundario, logrando así un sistema de mucho menor costo en comparación con el costo de un sistema con la misma capacidad de memoria pero diseñado sólo con memoria de alta velocidad.

Cuando el procesador intenta accesar a una localidad en el mapa de memoria virtual que no reside en ese momento en memoria física, el acceso a esa localidad es suspendido temporalmente mientras el dato necesario es buscado en el dispositivo de almacenamiento secundario, y colocado en memoria física;

completando hasta entonces el acceso.

En un simple ejemplo (figura 4.9a), si la memoria primaria es de L palabras de longitud y contiene una copia de tamaño L de la memoria secundaria, y si el bloque de memoria en la memoria secundaria comienza en X, entonces el contenido de la memoria primaria es el mismo que el residente entre X y X+L de memoria secundaria. El CPU sólo conoce el contenido de la memoria primaria desde 0 hasta L. Cuando el procesador accesa a la dirección de memoria L+1 (ADD), la información no puede obtenerse directamente de la memoria. Para accesar a esta palabra, la cual es en realidad X+L+1, se debe copiar en la memoria primaria el bloque que principia en la localidad X+L+1, por lo que se carga en la memoria primaria el bloque contenido desde X+L hasta X+L+L, teniendo un bloque de longitud L, logrando así el procesador realizar el acceso no sólo a la dirección original L+1 (ADD) sino que también a la siguiente dirección L+2 (CMP) (figura 4.9b).



Figuras 4.9a y 4.9b

Existen dos métodos básicos para la implementación de memoria virtual en un procesador: continuación de instrucción y reinicio de instrucción.

En el método de continuación de instrucción, el estado del procesador es salvado cuando se detecta una falta, y una vez que se completó la rutina de manejo de falta, el procesador continúa con el procesamiento de la instrucción en el mismo punto en donde se suspendió.

En el método de reinicio de instrucción, la instrucción en la cual ocurrió la falta es comenzada desde el inicio, después de que el manejador de excepciones complete la actividad asociada con la corrección de la falta. Esto es hecho sin importar el avance de instrucción que el procesador había alcanzado cuando se reconoció la falta.

El microprocesador 68010 utilizado en nuestro diseño, soporta memoria virtual, y utiliza el método de continuación de instrucción.

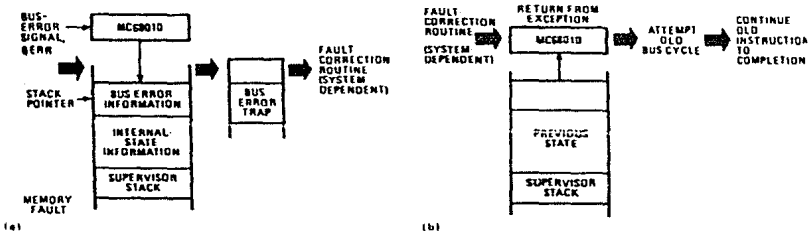
Para implementar este método, se utiliza la señal de error de bus (BERR), con el propósito de informar al procesador que existe un problema en el ciclo de bus en proceso.

Cuando el procesador recibe la señal de error de bus, aborta el ciclo y comienza una operación interna para suspender la instrucción. Entonces se posiciona en la pila (stack) supervisor, y coloca la información necesaria que ayudará a el sistema operativo a determinar la causa de la falta. Esta información incluye la dirección lógica, la instrucción que estaba siendo ejecutada, el registro de estado, y el valor del contador de programa que se tenía cuando ocurrió la falta. El sistema operativo puede entonces inspeccionar la falta para determinar la acción que se debe tomar y así corregir la falta; éste

procedimiento se ilustra en la figura 4.10a.

En el microprocesador 68010, una microinstrucción especifica toda operación interna asociada con cualquier instrucción, además cuenta con numerosos registros, latches y bits que contienen información que guía la operación del CPU. El microprocesador salva esta información ya que es necesaria para continuar una instrucción.

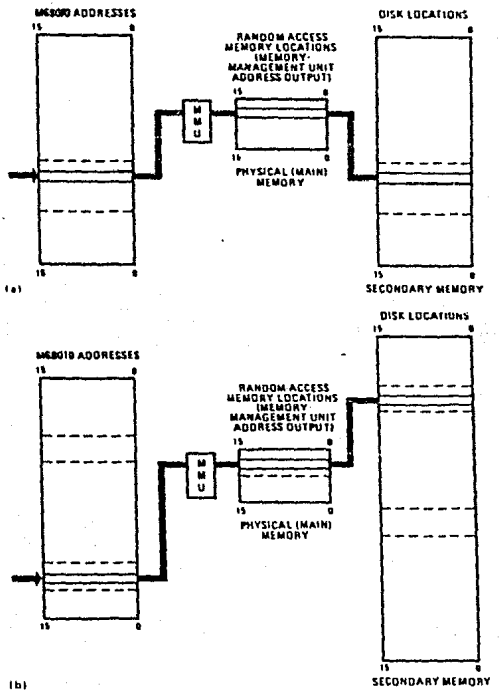
Cuando el CPU regresa de la subrutina de corrección, la instrucción RTE (regreso de excepción) carga nuevamente al microprocesador con el estado interno almacenado en la pila, continuando con la instrucción que se había suspendido (figura 4.10b).



Figuras 4.10a y 4.10b

Con el manejo de la memoria virtual también se pueden almacenar segmentos de la memoria virtual, en la memoria física, en vez de tener un solo bloque como en el caso del ejemplo anterior. De esta forma se puede tener un bloque que contenga instrucciones y otro que contenga datos, ambos almacenados en la memoria física, aunque la distancia entre las direcciones de estos bloques sea mayor que la capacidad de la memoria física, siempre y cuando el número total de localidades de memoria de los bloques sea menor o igual al número de localidades de memoria

física. En la figura 4.11a se observa la utilización de la memoria física como un solo bloque, mientras que en la figura 4.11b se ha dividido la memoria física en varios segmentos menores. El uso de la memoria segmentada tiene como finalidad el manejar la memoria física y secundaria en una forma más eficiente, con el objeto de realizar el menor número de accesos a la memoria secundaria.



Figuras 4.11 a y b

4.4 UNA APLICACION EN ROBOTICA.

A continuación se bosquejará la aplicación del Módulo Maestro VME cuyo diseño se describió en el capítulo 3 en un sistema de control de un robot manipulador, para ejemplificar las ideas que se desarrollaron anteriormente en este capítulo.

Los sistemas de control utilizados en robots varían de acuerdo a las funciones que debe desempeñar. El control de 6 grados de libertad (ver nota) para obtener una posición precisa generalmente requiere del uso de una minicomputadora, una potente microcomputadora de 16 bits o microprocesadores individuales para cada eje de movimiento, coordinados por una microcomputadora central. También se requiere de una cantidad considerable de memoria y alta velocidad para calcular y controlar los movimientos de los 6 ejes. Las pequeñas microcomputadoras por lo general no tienen estas capacidades, por lo que en muchos casos se utiliza una minicomputadora para estas aplicaciones, aunque una posibilidad alternativa son los sistemas de programación por aprendizaje en los que el operador define manualmente los movimientos de las articulaciones por lo que las necesidades de cálculo son mucho menores. Sin embargo, la limitación de estos sistemas es que el guiado manual es la única manera de programar al robot.

La disponibilidad de microcomputadoras de 16 bits y sistemas basados en uno o varios "buses" ha venido a reducir el costo de los sistemas de control de robots [4] y ha hecho cada vez más

---

NOTA: Una cadena cinemática formada por eslabones rígidos unidos dos a dos por articulaciones simples (rotacionales o translacionales) tendrá un grado de libertad G igual al número de articulaciones independientes que la constituyen.

frecuente la utilización de múltiples microprocesadores/ computadoras distribuidos de manera jerárquica para la adquisición de datos, análisis y control. Además de las ventajas de costo y velocidad, estas configuraciones facilitan la coordinación de los movimientos con la información de los sensores, sin que represente un problema grave el hecho de que algunos de estos sensores sean muy sofisticados y requieran por sí solos de un alto nivel de procesamiento, como por ejemplo los sensores de visión.

En la figura 4.12 se muestra la configuración general típica de un sistema distribuido jerárquicamento para un robot manipulador industrial.

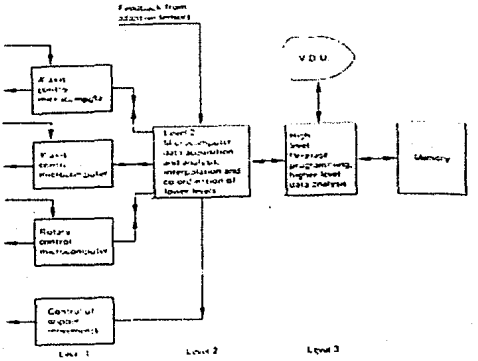


Figura 4.12 [4]

En esta configuración pueden distinguirse tres niveles de control y cada uno tiene una función específica acorde con su capacidad.

El primer nivel son los servos. Estos pueden ser un controlador analógico o estar basados en un microcontrolador de proposito especifico o constituir una microcomputadora dedicada. Este nivel comanda directamente los movimientos de los motores del robot, recibiendo la información necesaria de los sensores propioceptivos del robot, es decir, los sensores que proporcionan la información del estado interno de posición, velocidad y aceleración en cada articulación del robot [18].

En la siguiente página (fig. 4.13) se incluye una tabla de los sensores utilizados en un robot industrial.

Las acciones de control que se realizan en el primer nivel son coordinadas por la computadora usada en el segundo nivel; además, ésta también recibe información de realimentación de algunos de los sensores exteroceptivos, principalmente de los que requieren una respuesta más directa y rápida. Los sensores exteroceptivos son los que informan al sistema de control de: la ocurrencia de eventos externos, de la situación del entorno significativo, de los valores que adoptan las variables de posicionamiento relativo y de la magnitud de la interacción del robot con las piezas objeto de la tarea realizada. Toda esta información se analiza en este segundo nivel para determinar condiciones indeseables o de emergencia tales como colisión o fuerza excesiva y, si es necesario, tomar la acción pertinente o enviar un mensaje con esta información al tercer nivel.

El segundo nivel también realiza interpolaciones sencillas necesarias para mover el cuerpo del robot de manera suave y eficiente y tomar correctamente el objeto a manipular.

La computadora (mini o micro) en el tercer nivel es utilizada para el análisis de alto nivel de datos. Este nivel, en conjunto con los sensores exteroceptivos, constituyen la pieza clave en un robot inteligente ya que le permiten ser, por un

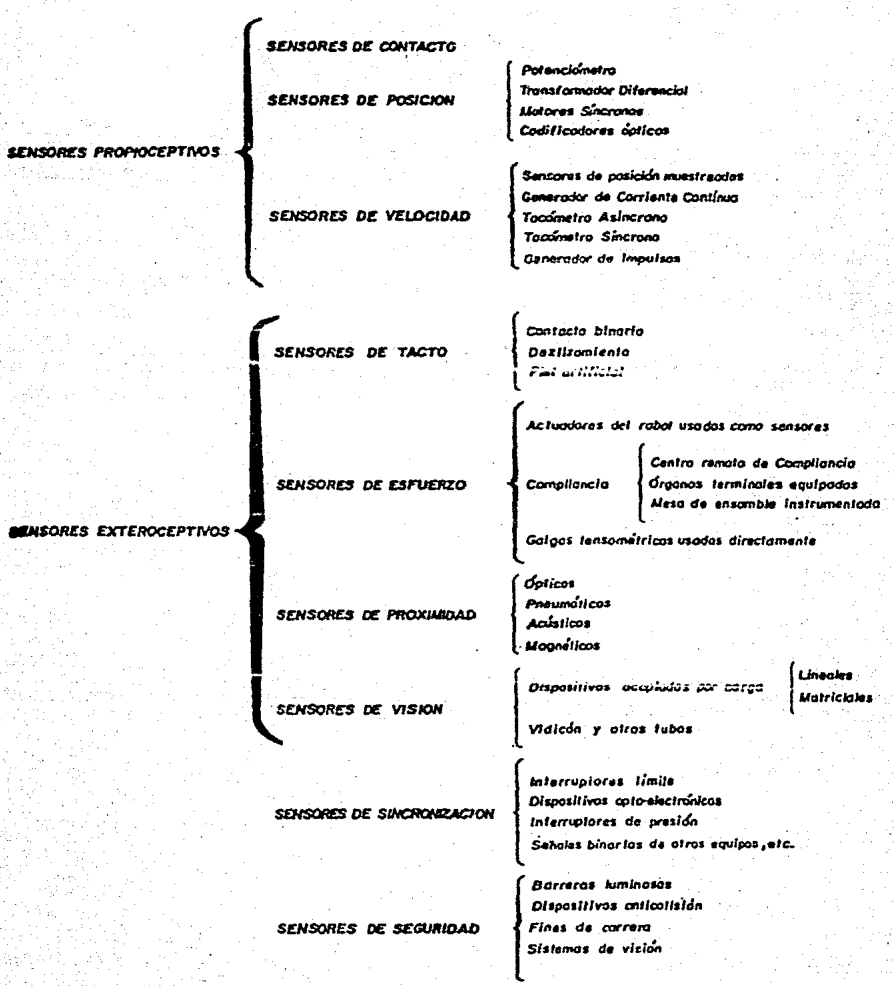


Figura 4.13 Sensores para robots

lado, capaz de realizar sus tareas aún ante cambios en sus condiciones de trabajo (cualidad conocida como adaptabilidad) sin necesidad de contactos de sincronización o de alimentadores sofisticados y, por otro lado, capaz de interactuar con el medio ambiente controlando dicha interacción. Es a este nivel que el operador tiene acceso a través de una consola (teclado y despliegue) para programar y transmitir órdenes al robot; gracias a las características de esta computadora ésta programación se realiza en un lenguaje de alto nivel que puede diseñarse de manera que el operador no necesite saber nada sobre la construcción interna del robot. A raíz de todo esto y de la sofisticación de la mayoría de los sensores exteroceptivos (visión, ultrasonido, piel artificial, por ejemplo), es lógico que en este nivel se requiera de alta velocidad y gran capacidad de cómputo.

4.5.1 Realización en VME

Aunque ya se han adoptado un canal de comunicaciones estandard exterior a los robots para conectarles otros equipos como máquinas de control numérico, sistemas de CAD/CAM, o inclusive otros robots (el "bus" IEEE-488/GPIB), no se ha establecido nada sobre el canal interno del robot; por lo que hasta ahora, cada fabricante ha venido definiendo su propia arquitectura interna de acuerdo a los componentes y productos que cada compañía maneja. De esta manera, las opciones y el crecimiento del sistema están limitados en la mayoría de los casos a la línea de productos que ofrece el fabricante del robot. Al utilizar el bus VME, que es un estandard industrial, en una arquitectura semejante a la de la figura 4.12, se tiene la ventaja de que existe una gran cantidad de compañías que ofrecen toda clase de tarjetas para este bus,abarcando una diversidad de controladores de periféricos, interfaces de conversión A/D y D/A y optoacopladores, así como sofisticados procesadores de video.

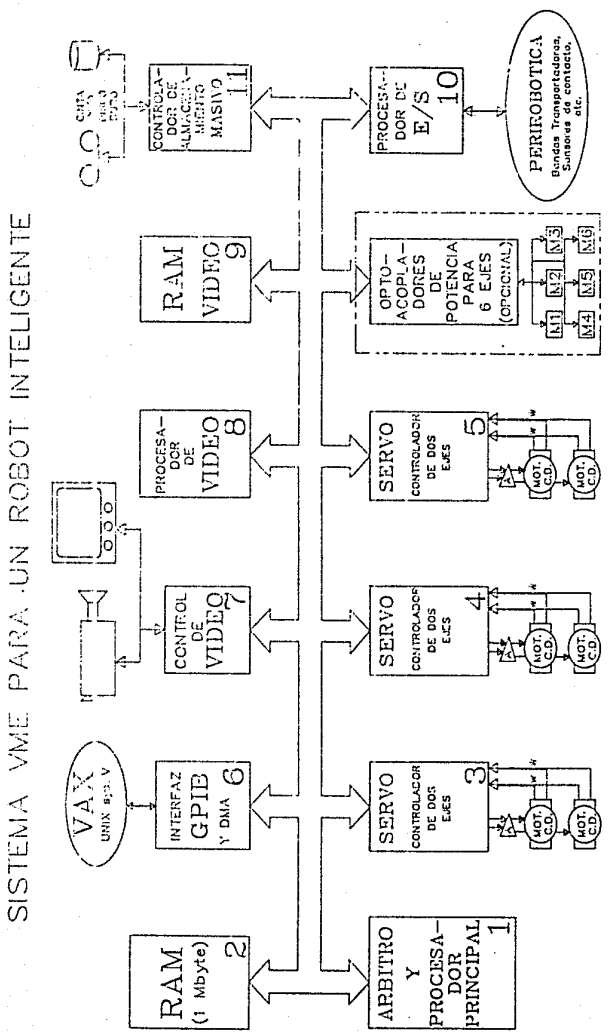


Figura 4.14 Sistema VME para un Robot

En la figura 4.14 de la pagina anterior se propone un sistema basado en VME,con el Módulo Maestro VME, para un robot manipulador industrial de 6 grados de libertad.

La primera tarjeta del sistema es el Módulo Maestro VME y constituye la computadora de nivel dos en el sistema de control. Como ya se detalló anteriormente en el capitulo III, esta tarjeta contiene el árbitro del bus en su opción ONE, puede atender todas las interrupciones y comunicarse con cualquier tarjeta en el bus; además representa el maestro de más alta prioridad por su posición en el bus. Sus características le permiten realizar otras funciónes adicionales a las definidas para el nivel de control dos. Se encarga de inicializar y programar el funcionamiento de los otros elementos en el sistema al encenderlo o reiniciarlo y continuamente atiende funciones necesarias para el funcionamiento conjunto del sistema; reprograma o modifica la operación de los otros elementos cuando es necesario, puede efectuar rutinas de prueba cada determinado tiempo como medida preventiva, recibe y maneja las situaciones críticas o de emergencia y tiene capacidad para tomar decisiones por encima del nivel tres de control si es necesario por alguna de esas situaciones, e inclusive puede asumir las funciones de algunos de los otros módulos (como los servos por ejemplo) en el evento de una falla.

Para cumplir con las funciones que corresponden al nivel dos de control, adquiere la información de la posición absoluta de las articulaciones a través del bus VME y puede recibir otros sensores propioceptivos y exteroceptivos (señales todo o nada) en sus dos puertos paralelos disponibles a través de un conector externo. Utiliza estos mismos puertos para controlar la pinza o la herramienta que constituya la "mano" del robot. Realiza un primer análisis de los datos generados por el sistema de visión antes de pasarlos al nivel tres para detectar la ocurrencia de cambios bruscos o inesperados dentro del área de operación. Finalmente, se encarga de coordinar y enviar la

información necesaria a las tarjetas servocontroladoras en función de las órdenes que recibe de la computadora del nivel tres y dependiendo del análisis que realice sobre los datos adquiridos de los sensores propioceptivos. En este punto también puede realizar cálculos sencillos para suavizar los movimientos o para adaptarse a una situación de la cual el nivel tres todavía no tenga conocimiento.

En la segunda ranura se encuentra la memoria RAM del sistema VME y es un módulo esclavo. Se propone una capacidad de 1 Mbyte aunque ésto depende de las necesidades de cada caso. Cualquier maestro del bus puede accesar esta tarjeta. Prácticamente todos los fabricantes de sistemas VME ofrecen varias opciones de tarjetas de memoria.

Las tarjetas en las posiciones 3, 4 y 5 conforman el nivel 1 del sistema de control. Son tres servo-controladores de dos ejes, modelo IV-1666 (Dual Axis Servo Controller) de la compañía IRONICS Inc. con las siguientes características: control de dos motores de C.D. por modulación de ancho de pulso, generador de frecuencia de repetición de pulso, dos generadores de ancho de pulso de 16 bits programables, dos interfaces para codificadores incrementales con contadores arriba-abajo de 24 bits, 4 convertidores A/D y 2 D/A de 8 bits, puerto de entrada y puerto de salida, ambos de 16 bits. Es un módulo esclavo y todo su funcionamiento es programable por software. Sus salidas y entradas están conectadas al conector P2 del bus. Proporciona los datos de posición, velocidad y aceleración (aunque esta no es muy utilizada por la dificultad para conseguir sensores de aceleración adecuados para las articulaciones de robots) en los contadores y convertidores A/D.

Para manejar los motores se requiere de una etapa externa de amplificación de corriente, aunque existe la opción de utilizar la tarjeta encerrada en línea discontinua en el dibujo; esta tarjeta es también de IRONICS y contiene los acopladores ópticos

y amplificadores de potencia necesarios para 6 ejes. La función de control de estas tarjetas consiste en mantener a los motores en un régimen constante ya sea de posición, velocidad o aceleración en su caso. Esta función es ordenada por el nivel dos y por su capacidad limitada, los servos requieren de cierta cooperación de éste nivel dos para realizarlas, por lo que en el esquema de control implementado el nivel dos cumple con algúnas funciones que se habían definido para el nivel uno. La mayor capacidad del Módulo Maestro VME utilizado en el nivel dos hace posible subsanar la capacidad limitada de los módulos utilizados en el nivel uno.

En la siguiente ranura del sistema está una tarjeta de interfaz GPIB con controlador de DMA, modelo IV-1621 (IEEE-488/GPIB) Interface DMA Controller) también de IRONICS cuyas características son: controlador DMA HD68450 (500KBytes/sec, tamaño de transferencia ilimitado, operación programable), completa capacidad como Talker, Listener y Controlador del GPIB, configuración programable, e/s del GPIB en el conector P2 o en el panel frontal. Es un módulo maestro con la segunda prioridad de acceso al bus. A esta tarjeta se conecta la computadora de nivel tres que bien podría ser otro sistema VME con las características adecuadas para cumplir con este propósito, o una minicomputadora tipo VAX como la que se propone en el dibujo.

Esta tarjeta constituye la vía de comunicación entre los niveles dos y tres de control y su operación puede ser programada por el nivel tres (pero sólo a través del Módulo Maestro VME) para controlar el sistema de video y el procesador de E/S y adquirir datos de estos y para accesar la RAM y el controlador de almacenamiento masivo.

Las tarjetas en las posiciones 7, 8 y 9 constituyen el sistema de adquisición, almacenamiento y despliegue de imagen para cámaras CCD y son el modelo DSSEIMAGC-5X de Data Sud Systèmes s.a. (Francia). La parte de adquisición recibe

directamente la señal de hasta 4 cámaras, puede digitalizar una imagen en 34 ms (convertidor A/D rápido de 8 bits) y tiene un registro programable para controlar la digitalización. La Tarjeta de almacenamiento contiene 256 KBytes de RAM dinámica de auto-refresco rápido con doble puerto, interfaz a VME con acceso a 128K palabras y transparencia para el procesamiento de la imagen. En la parte de despliegue maneja un monitor blanco y negro con 256 tonos de gris, tiene un convertidor D/A de 8 bits y de alta velocidad y un registro para seleccionar la salida del DAC: imagen verdadera, invertida, complemento a dos o complemento a dos invertida. Estas características facilitan por ejemplo el variar contrastes, detectar bordes y realizar amplificaciones. La adquisición puede ser disparada por una fotocelda y generar interrupciones. Además de la comunicación con el bus VME, las tres tarjetas se comunican entre sí por otro canal privado (DSSIMAGC-5X BUS) en el conector P2 que viene a ser el bus VMX, por lo que no interfieren con la operación del bus VME.

Los datos generados son supervisados continuamente por el Módulo Maestro VME ántes de que los lea la VAX con el fin de detectar rápidamente la aparición de objetos extraños inesperados en el campo de acción del robot o verificar que la trayectoria de la extremidad del robot se desarrolle dentro de ciertos límites calculados.

En la ranura 10 se colocó un procesador que controla los puertos de entrada y salida tanto digitales como analógicos necesarios para el equipo de perirobótica que rodea al robot y que conforman un puesto robotizado completo. Ejemplos de éstos equipos son: bandas transportadoras, conmutadores de haz de luz, sensores de contacto y herramientas diversas (taladro, sierra, pistola de pintura, etc.). En la última posición del sistema esta el procesador de almacenamiento masivo, que controla el acceso a dispositivos como disco duro, blando y/o cinta magnética. Estas dos últimas tarjetas pueden ser módulos maestros, e inclusive contar con un controlador de DMA que es muy útil sobre todo para

el controlador de almacenamiento masivo. Son utilizadas por la Vax y por el Módulo Maestro VME y su operación es controlada por este último. Casi todos los vendedores de equipos VME ofrecen diversidad de opciones para elegir estas dos tarjetas.

A continuación se explicarán las interacciones entre los diferentes elementos del sistema.

La memoria RAM (2) (ver nota) constituye el área de comunicación entre los procesadores. Esta memoria es compartida modularmente entre el procesador principal o Módulo Maestro VME (1), la VAX a través de la interfaz GPIB, el procesador de video (8), el procesador de E/S (10) y el controlador de almacenamiento masivo (11). El espacio de memoria estaría segmentado en diferentes regiones, definiendo "buzones" de intercomunicación entre los diferentes procesos que se realizan en el sistema. Una de esas regiones contendría todos los semáforos necesarios para la exclusión mutua en el acceso a los segmentos de memoria, a la memoria de video y a otros recursos compartidos como el disco duro, cinta o impresora, y para la sincronización entre los procesos que así lo necesiten. De acuerdo a los mecanismos ya detallados en la primera parte de este capítulo, cada semáforo tendría asociada una bandera para controlar el acceso al propio semáforo, y una cola de los procesos que queden bloqueados durante las operaciones con los semáforos. El acceso a las banderas se realiza con un ciclo indivisible generado por la instrucción TAS del 68010 en el Módulo Maestro VME, que ejecutaría la misma ya sea como parte del proceso que está corriendo o por órden de la VAX para permitirle a ésta el acceso a los semáforos. Los otros procesadores también pueden realizar ciclos indivisibles para utilizar semáforos.

NOTA: El número entre paréntesis es la posición de la tarjeta a la que se hace referencia.

Tanto el Módulo Maestro VME (1) como la VAX deben utilizar mecanismos de memoria virtual con la RAM (2) y la RAM propia. En el caso del Módulo Maestro VME, utiliza un espacio de la memoria física residente en la misma tarjeta para accesar páginas de la RAM (2), otro espacio para accesar páginas de datos o instrucciones en disco duro o cinta y la señal BERR* del 68010 y el canal disponible del DMAC para efectuar los cambios de página necesarios. La VAX puede efectuar acceso directo a la memoria de video e implementarla internamente como memoria virtual en un espacio de su memoria física. También el acceso a disco y cinta los realiza a través de un segmento de la RAM (2). Las rutinas de cambio de página las puede descargar en el Módulo Maestro VME (1) programando al controlador de DMA (6) para que genere una interrupción cada que se necesite el cambio de página; al atender esta interrupción, el procesador reprogramaría al DMAC (6) para que efectúe dicho cambio de página.

La comunicación entre la VAX y el Módulo Maestro VME (1) se realizaría también generando una interrupción después de efectuada la transferencia ente la VAX y el segmento correspondiente de la RAM (2). Para la comunicación en sentido inverso, después de escribir en la RAM (2) el Módulo Maestro programa el DMAC (6) para transmitir de RAM a VAX.

Dada la arquitectura del sistema, se observa que existe redundancia de hardware y de información en varias partes del mismo: la posición del extremo del robot puede obtenerse de las interfaces con los codificadores incrementales en los servos (3, 4 y 5) o con ayuda del sistema de video o de los sensores exteroceptivos y de perirobótica, por lo que se puede comparar esta información para detectar y localizar una falla de posicionamiento, misma que se podría enmascarar e incluso tolerar, aunque la operación del robot quedaría limitada. Por otro lado, tanto la VAX como el procesador (1) pueden enviar órdenes a los servos, por lo que si se detectara una falla en

éste procesador, el sistema se puede recuperar de ella al hacerse cargo la VAX de todo el funcionamiento. Dado que el Módulo Maestro VME (1) no puede controlar por sí solo toda la operación del robot, en caso de fallar la VAX o la comunicación con la misma sólo se podría detectar y localizar la falla y para evitar acciones indeseadas deteniendo en forma ordenada el sistema. Existen, sin embargo, partes del sistema que no son tolerantes a fallas, lo cual se podría subsanar conectando en el canal GPIB o en el VME otro sistema igual, separados por un dispositivo comparador que podría detectar una discrepancia entre los dos sistemas, ordenar un autodiagnóstico en ambos y deshabilitar aquél que no pasara las pruebas, por ejemplo.

Por último, como los movimientos del brazo de un robot no pueden ser muy rápidos y dado que con los elementos descritos esta arquitectura tendría una alta velocidad de procesamiento y siempre garantizaría un tiempo máximo de atención a eventos, con base en las definiciones establecidas en el capítulo I podemos concluir que la arquitectura propuesta aquí constituye un sistema de multiprocesamiento en forma de multiprocesador débilmente acoplado, trabajando en estricto tiempo real, con cierto grado de redundancia y por tanto de tolerancia a fallas, que es justamente el tipo de aplicaciones que se plantearon como objetivo de este trabajo.

CONCLUSIONES

CONCLUSIONES

A continuación se resumen las actividades realizadas durante la elaboración de este trabajo.

Primeramente se estableció el objetivo del trabajo: diseñar una microcomputadora de 16 bits para sistemas de multiprocesamiento con capacidad de comunicación entre procesos, manejo de memoria virtual, operación en tiempo real y cierto grado de tolerancia a fallas. Después de un análisis de los elementos de diseño disponibles, se eligieron el microprocesador MC68010 de Motorola y la arquitectura VME para dicho propósito. Después se realizó un estudio de los conceptos involucrados partiendo de sus bases teóricas para obtener un diseño que fuera capaz de soportar las características requeridas y coexistir en un medio ambiente de multiprocesamiento. Durante esta etapa se evaluaron las herramientas de diseño que se pensaban utilizar para la realización práctica del proyecto. Estas herramientas son: un emulador del 68010 trabajando a 8 MHz, manejado a través de una minicomputadora VAX con sistema operativo UNIX system V y con un macroensamblador y compilador de lenguaje C que corriendo sobre la VAX permiten ejecutar los programas a través del emulador. Se utilizó una computadora CODATA de la red del IIMAS para obtener las rutinas que generaba en lenguaje ensamblador del 68010 a partir de pequeños programas escritos en C para estudiar

la interaccion entre el sistema operativo y los programas en ensamblador, ya que esta máquina también se utilizaría para las pruebas de software.

Una vez realizado el diseño lógico de la tarjeta, se estudió con más detalle la realización práctica de las características que se deseaba cumpliera el sistema en una aplicación real. Se eligió un robot manipulador industrial ya que esta aplicación requiere de multiprocesamiento, mecanismos de comunicación entre procesos, memoria virtual, tolerancia a fallas y respuesta en tiempo real, aunque por estas características, el sistema también podría utilizarse para cualquier aplicación de control industrial.

Finalmente se construyó un prototipo basado en el microprocesador 68008, compatible casi totalmente con el 68010 pero en 48 patas y con un canal de datos de 8 bits. En este prototipo se realizaron pruebas de respuesta del sistema a los mecanismos de comunicación entre procesos (instrucción TAS) y de memoria virtual (señal de BERR*), encontrando que satisfacían las condiciones esperadas. La operación en estricto tiempo real depende principalmente del software, ya que el hardware siempre garantiza un tiempo límite máximo de respuesta a un evento.

Del trabajo desarrollado se obtienen las siguientes conclusiónes:

- El diseño del sistema cumple con las características propuestas de comunicación entre procesos, memoria virtual, tolerancia a fallas y tiempo real.

- La elección de la arquitectura del sistema (VME) es actualmente de gran aceptación en el mercado.

- La tecnología en el diseño es de gran actualidad con posibilidades de crecimiento a corto plazo (68010-68020).

- El diseño del sistema permite aplicación inmediata en medios ambientes de control industrial, en los que hay que considerar respuestas en tiempo real, procesamiento distribuido y tolerancia a fallas.

- Las aplicaciones en robótica se prestan en forma adecuada para utilizar este sistema, tal como se muestra en el capitulo de aplicaciones.

-Se contruyó un prototipo basado en 68008 que permitió verificar el cumplimiento de las caracteristicas que se definieron como necesarias para un medio ambiente de multiprocesamiento.

- EL presente trabajo sirve como base para un futuro desarrollo de un sistema completo aplicable a diversos problemas de multiprocesamiento y control industrial.

BIBLIOGRAFIA

[1] Jonathan A Humphry, "Fault tolerance and Mirros in the Real World",IEEE Micro, December 1984.

[2] Barry W. Jonshon, "Fault Tolerant Microprocesor Based Systems", IEEE Micro, December 1984.

[3] Walter S. Heath, "A System Executive for Real-Time Microcomputer Programs", IEEE Micro, June 1984.

[4] A.K. Kochnar & N.D. Burns, "Microprocessors and their Manufacturing Applications", Ed. Edward Arnold, G.B. 1983.

[5] C.L. Smith, "Computer Process Control", Ed. Intext, N.Y. 1963.

[6] Chuan-Lin Wu, "Multiprocessing Technology", Computer, Vol. 18, No. 6, June 1985.

[7] P.C. Patton, "Multiprocessors: Architecture and Applications", Computer, Vol. 18, No. 6, June 1985.

[8] R. Rosenberg, "Battle of the Buses: AND THE WINNER IS...", Electronics, Nov. 25, 1985.

[9] M.J. Flynn, "Very High Speed Computing Systems", Procs. IEEE, Vol. 54, Dic, 1966.

[10] Kai-Hwang & Fayé A. Briggs, "Computer Architecture and Parallel Processing", Mc Graw-Hill, USA, 1984.

[11] A.M. Lister, "Fundamentals of Operating Systems", Mac Millan, S.B., 1985.

[12] A. Guzmán y A. Segovia, "A Parallel Reconfigurable Lisp Machine", Comunicaciones Técnicas IIMAS Vol. 7, Serie Naranja, Num. 133, Mex. D.F. 1976.

[13] Yu-Cheng Liu & Glenn A. Gibson, "Microcomputer Systems: The 8086/8088 Family. Arch. Prog. and Design", Prentice-Hall, 2nd. Ed. N.J. 1986.

[14] Hanna Oktaba, "Programación Concurrente (Primera Parte)", Comunicaciones Técnicas IIMAS, Serie Azul No. 86 Mex. 1985

[15] The TTL Data Book for Design Engineers Texas Instruments Inc. Dallas, Texas, 1976

[16] MC 68010 16-/ 32 -BIT Virtual Memory Microprocessor, User's Manual, Motorola, Austin, Texas, 1984.

[17] MC 680440 Dual Channel Direct Memory Access Controller, User's Manual, Motorola, Austin, Texas, 1984.

[18] Juan M. Ibarra Z., Rafael Kelly M., Romeo Ortega, Apuntes del curso "Robótica industrial". Div. de Ed. Cont. F.I. UNAM, Mex. 1987

[19] MC 68901 Multifunction Peripheral, User's Manual, Motorola, Austin, Texas, 1984.

[20] MC 68230 Parallel Interface/Timer, User's Manual

[10] Kai-Hwang & Fayé A. Briggs, "Computer Architecture and Parallel Processing", Mc Graw-Hill, USA, 1984.

[11] A.M. Lister, "Fundamentals of Operating Systems", Mac Millan, G.B., 1985.

[12] A. Guzmán y R. Segovia, "A Parallel Reconfigurable Lisp Machine", Comunicaciones Técnicas IIMAS Vol. 7, Serie Naranja, Num. 133, Mex. D.F. 1976.

[13] Yu-Cheng Liu & Glenn A. Gibson, "Microcomputer Systems: The 8086/8088 Family. Arch. Prog. and Design", Prentice-Hall, 2nd. Ed. N.J. 1986.

[14] Hanna Oktaba, "Programación Concurrente (Primera Parte)", Comunicaciones Técnicas IIMAS, Serie Azul No. 86 Mex. 1985

[15] The TTL Data Book for Design Engineers Texas Instruments Inc. Dallas, Texas, 1976

[16] MC 68010 16-/ 32 -BIT Virtual Memory Microprocessor, User's Manual, Motorola, Austin, Texas, 1984.

[17] MC 680440 Dual Channel Direct Memory Access Controller, User's Manual, Motorola, Austin, Texas, 1984.

[18] Juan M. Ibarra Z., Rafael Kelly M., Romeo Ortega, Apuntes del curso "Robotica industrial". Div. de Ed. Cont. F.I. UNAM, Mex. 1987

[19] MC 68901 Multifunction Peripheral, User's Manual, Motorola, Austin, Texas, 1984.

[20] MC 68230 Parallel Interface/Timer, User's Manual

Motorola. Austin, Texas. 1983.

[21] Peñarrieta, L.H., "Multiprocesamiento basado en una memoria compartida por n procesadores" , Tesis de maestría en Ingeniería Electrónica, México D.F., 1978.

[22] IRONICS VME PRODUCTIVITY SERIES
Ironics Inc., N.Y., 1985

[23] Data Sud Systèmes s.a. VME
Data Sud Systèmes s.a., Montpellier, France, 1985.

[24] VMEbus Specification Manual
VMEbus Manufacturers Group, Motorola, Phoenix, Ariz., 1982.

[25] VMXbus Specification Manual
VMEbus Manufacturers Group, Motorola, Phoenix, Ariz., 1983.

[26] VMSbus Specification Manual
VMEbus Manufacturers Group, Motorola, Sunnyvale, Cal. 1983.