UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

IMPLANTACIÓN DE TÉCNICAS DE TOLERANCIA A FALLAS, ENSAMBLE Y VALIDACIÓN OPERATIVA DE LA COMPUTADORA DE VUELO DEL MICROSATÉLITE SATEX1

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO ELÉCTRICO ELECTRÓNICO

P R E S E N T A

HUGO ANDRÉS ORTÍZ MARTÍNEZ

ASESOR: M. I. ESAÚ VICENTE VIVAS

MÉXICO, D. F., ENERO 2003

A

A DIOS
A MIS PADRES
A MI HERMANO HUMBERTO (Q.E.P.D.)
A MIS HERMANAS NOHEMI Y CLAUDIA
Y A TODOS AQUELLOS QUE ME
AYUDARON A SER LO QUE HOY SOY

Agradecimientos

A la Universidad Nacional Autónoma de México y a la Facultad de Ingeniería, por haberme dejado ser parte de ellas.

A todos los profesores de la facultad de ingeniería, por su dedicación y esfuerzo para con todos sus alumnos, en especial:

Al Ing. Lauro Santiago Cruz, por su don de gente y por haberme dado la oportunidad de trabajar y aprender con el.

Al Ing. Rodolfo Peters Lammel, por sus consejos y enseñanzas.

Al Ing. Enrique Arenas Sánchez, por su amistad incondicional y sus enseñanzas durante los primeros años de la carrera.

A Roberto Ch."Morron", Carlos A. "Cuaresmeño", Alejandro C. "Jalapeño", Jessica O. "Chilindrina", Cynthia B., Pedro D. R. "PETE", Jaime M. "Tuersex", José Luis G. "Choche", Luis Antonio L. "Kazen", y a todos aquellos que por el momento no recuerdo pero que compartieron conmigo muchos momentos dentro y fuera de la FI.

A Esaú por ser buen amigo, por orientarme y apoyarme en el desarrollo de esta tesis.

A mis amigos y compañeros del proyecto SATEX: Ramón, Adán, Carlos, Jorge, Leo, Iris, Luis, David y Manuel, por ser buenos amigos, por el apoyo brindado durante la realización de esta tesis y por la forma de hacer equipo.

A Gaby T. BB, por que a lo largo de casi 5 años de Noviazgo, me has apoyado y no has dejado que esto quede en el olvido.

A las familias Junco Ortiz, Pérez Martínez, Reta Martínez, Ortiz Venegas, Martínez Cortés, Tavares Berber, y a todos aquellos que de forma desinteresada me apoyaron para la realización de este trabajo y para mi desarrollo profesional y humano.

"Dale a un hombre un pez y se alimentará un día, enséñalo a pescar y se alimentará toda la vida"
Jesus de Nazareth

D

ÍNDICE

PAGINACIÓN DISCONTINUA

Capítulo 1. Introducción

1.1 Antecedentes

Desde el inicio de la era espacial han existido los satélites pequeños como el SPUTNIK-1, de la hoy desaparecida Unión Soviética, que fue el primer satélite artificial en orbitar la tierra. Sin embargo, han sido los satélites grandes y costosos los que han proliferado en las aplicaciones espaciales de las últimas décadas. Durante este tiempo, el uso de los satélites pequeños se ha limitado a pequeños grupos de científicos o amateurs, sin embargo, en años recientes, el desarrollo de satélites pequeños ha tomado auge, debido en gran medida a los grandes avances en el campo de la microelectrónica, lo cual ha convertido a los satélites pequeños en una alternativa viable.

Entre las ventajas que presentan los satélites pequeños respecto a los satélites de mayor tamaño podemos mencionar que, los primeros proporcionan soluciones efectivas a costos menores, además requieren de un tiempo de desarrollo significativamente más corto, lo que implica que sea posible el uso de tecnología avanzada en estos sistemas dando paso a una amplia gama de aplicaciones.

Actualmente es común que los satélites pequeños se utilicen para complementar los servicios de los satélites de mayor tamaño ya existentes, siendo capaces de proporcionar servicios eficientes y costeables en diferentes aplicaciones como lo son:

- Comunicaciones especializadas
- Misiones Militares
- Percepción remota
- Demostración de nuevas tecnologías

Algunos ejemplos de las aplicaciones de los satélites pequeños las podemos ver con los sistemas de comunicaciones recientemente propuestos GLOBALSTAR y TELEDESIC, que cuentan con constelaciones de satélites pequeños capaces de proporcionar servicios de comunicación móvil de voz y datos, de fácil acceso y alta capacidad, con cobertura mundial. A pesar del fracaso del sistema IRIDIUM, los sistemas citados permiten prever que los satélites pequeños tendrán un papel determinante en el ámbito satelital.

Debido a lo anterior, el interés mundial en los satélites pequeños ha ido en aumento rápidamente, lo cual se hace patente en el hecho de que grandes compañías, gobiernos, universidades y otras organizaciones han comenzado sus propios programas de satélites pequeños, tal es el caso de la universidad de Surrey en Inglaterra.

México por su parte, desde hace algunos años a tratado de incursionar en la investigación y desarrollo de tecnología espacial en lo que respecta a satélites de órbita baja, prueba de esto han sido los 2 intentos realizados por la UNAM a través del Programa Universitario de Investigación y Desarrollo Espacial (PUIDE). El primero con el UNAMSAT-A, el cual fracasó debido a una falla en el cohete lanzador, y el segundo con el

UNAMSAT-B, que logro ser puesto en órbita y que funcionó correctamente solo un par de meses. En la actualidad el PUIDE está extinto por lo que no se espera ningún avance por su parte.

Por otro lado, el ahora desaparecido Instituto Mexicano de Comunicaciones (IMC), impulsó un ambicioso proyecto que tenía como finalidad, el diseñar, construir y validar un microsatélite con tecnología espacial cien por ciento mexicana, siendo esta la primera experiencia a este nivel. Dicho proyecto recibió el nombre de **SATEX.**

1.1.1 Microsatélite experimental Mexicano, proyecto SATEX

En un principio el proyecto **SATEX** fue patrocinado por el IMC después financiado por la Comisión Federal de Telecomunicaciones (COFETEL), y actualmente se encuentra buscando el patrocino final para la etapa de pruebas de validación finales y lanzamiento a corto plazo.

Debido a la envergadura del proyecto, se invito a participar a diversas instituciones educativas y/o de investigación de todo el país, dichas instituciones se mencionan a continuación:

- Centro de Investigación Científica y de Estudios Superiores de Ensenada, CICESE, en Baja California.
- Centro de Investigación en Matemáticas, CIMAT, en Guanajuato.
- Instituto Politécnico Nacional, IPN, por medio de la Escuela de Ingeniería Aeronáutica y la sección de graduados de la Escuela Superior de Ingeniería Mecánica y Eléctrica, ESIME.
- Instituto de Astrofísica Óptica y Electrónica, INAOE, ubicado en Puebla.
- Universidad Nacional Autónoma de México, a través del Instituto de Ingeniería y del Instituto de Geografía.
- Centro de Investigación en Tecnología Digital, CITEDI, de Tijuana, Baja California.

Dentro del proyecto **SATEX,** cada una de las instituciones participantes es responsable de desarrollar, alguno o algunos de los subsistemas del microsatélite, esta asignación se observa en la figura 1.1.

Dentro de los objetivos iniciales del proyecto SATEX, podemos mencionar los siguientes:

- Formación de recursos humanos en proyectos de tipo espacial en todas las instituciones participantes.
- Participación de investigadores jóvenes y estudiantes.
- Promoción de convenios de colaboración académica entre instituciones nacionales.

- Desarrollo de tecnología espacial 100% mexicana aplicable a futuros proyectos.

INSTITUCIÓN	SUBSISTEMA A CARGO
CITEDI	• Cámara digital (Cámara CCD) • Sistema de potencia (SP) • Celdas solares • Baterías
IPN	• Coordinación general ESIME: • Bobinas de torque magnético (BTM) ESIME Posgrado: • Experimento en banda Ka (Ka). Aeronáutica: • Integración y pruebas
CICESE	• Equipos de radio para comunicaciones en VHF (VHF1 y VHF2). • Experimento de comunicaciones mediante enlace óptico (CUO). • Decodificador de tonos o Procesador de Sobrevivencia(DT ó PS). • Hardware de estación terrena.
INAOE	• Recepción en banda Ka (estación terrena de Ka)
CIMAT	• Modelado de la dinámica orbital del vehículo, de campo magnético terrestre y de estabilización en tres ejes. • Algoritmos de estabilización del satélite.
UNAM	Instituto de Ingeniería UNAM: • Computadora de Vuelo (CV) • Hardware de acondicionamiento y multicanalización de sensores • Sensores de corriente, temperatura, voltaje y magnetometría • Protocolos y software de telemetría y comando • Protocolos de control distribuido • Software de vuelo • Software de estación terrena • Sistema experto para control satelital • Cámara digital (CCD) • Experimento de computación semivirtual Instituto de Geografía • Mesa Suspendida en Aire, para validación de algoritmos de estabilización. • Sensores Finos de Sol (SFS)

Figura 1.1 Asignación de los subsistemas a las instituciones participantes.

Durante el ciclo de vida del proyecto SATEX, éste ha pasado por diversas etapas, en un principio todos los integrantes avanzaban de forma continua, sin embargo a lo largo de la evolución del proyecto se han encontrado diversos obstáculos que han impedido un desarrollo sostenido por parte de las instituciones involucradas. El principal problema ha sido el financiamiento, a pesar de ello, el Instituto de Ingeniería y algunas de las otras instituciones participantes han seguido trabajando de manera continua por lo que los subsistemas a su cargo se encuentran en la fase final de su desarrollo.

1.2 Características Globales del microsatélite SATEX-1

Según la clasificación mostrada en la tabla 1.1, la cual es ampliamente aceptada, el Satélite SATEX se encuentra clasificado como un Microsatélite ya que es un cubo de 47 centímetros de lado con una masa estimada de 55 kilogramos.

Tabla 1.1: Clasificación de los satélites según su masa

CLASIFICACIÓN	MASA [Kg.]
Satélite grande	>1000
Satélite mediano	500-1000
Mini satélite	100-500
Micro satélite	10-100
Nano satélite	1-10
Pico satélite	0.1-1
Femto satélite	<0.1

Comúnmente un microsatélite se encuentra constituido por diversos subsistemas con tareas específicas que interactúan estrechamente, confiriéndole capacidad de operación y de sobrevivencia para lograr cada una de sus misiones [VICENTE, 1996]. A continuación se describen de forma general algunos de los subsistemas más importantes que constituyen un satélite:

- Subsistema de telemetría y comando: Permite al satélite mantener comunicación con su segmento en tierra (Estación Terrena) y/o en su caso con otros satélites. Mediante este subsistema el satélite es capaz de recibir comandos desde la Estación Terrena (ET), y de bajar la telemetría adquirida, es decir, las mediciones de los sensores integrados en el satélite.
- Subsistema de potencia: Es el encargado de la administración primaria de la energía en el satélite, es decir, el encargado de la conversión de la energía solar recolectada por los paneles, en energía eléctrica que es almacenada en las baterías químicas de abordo. También es el responsable de la distribución de la corriente y los voltajes eléctricos para cada uno de los subsistemas del satélite.
- Subsistema de Navegación: Este subsistema se encuentra integrado por los sensores necesarios para la determinación de la orientación y posición del satélite, además del equipo de control digital que permite ejecutar los algoritmos

de control para la navegación del satélite, mediante el accionamiento de los sistemas de propulsión y estabilización.

- Subsistema de propulsión y estabilización: los satélites grandes generalmente tienen requisitos de estabilización grandes, por lo que es común que utilicen sistemas de propulsión activos mediante cohetes. También pueden utilizar ruedas inerciales. Los microsatélites no utilizan sistemas de propulsión, estos satélites comúnmente cuentan con sistemas de estabilización pasivos como magnetotorques, bobinas de campo magnético y gradientes gravitacionales.
- Subsistema de control térmico: Puede ser pasivo o activo, en los primeros se utilizan disipadores para reflejar el calor en áreas expuestas a los rayos solares. En las áreas no expuestas, se utilizan materiales de color oscuro para retener el calor. Los sistemas activos utilizan calefactores y enfriadores eléctricos ya sea para aumentar o disminuir las temperaturas locales.
- Subsistema de comunicaciones: Se encarga de recibir y/o enviar las señales de comunicación las cuales pueden ser de audio, vídeo o datos.

Tomando como base lo anterior podemos señalar las características principales del microsatélite SATEX.

El sistema de potencia de SATEX se encuentra constituido por cuatro paneles solares colocados en cuatro lados del cubo para generar la potencia que se suministrará a los subsistemas electrónicos mediante el sistema de baterías, este sistema es controlado por un microprocesador de tipo militar, que además es el encargado de la conversión de voltajes (mediante el uso de convertidores DC-DC), distribución de corriente y administración de energía. El microsatélite cuenta con cuatro cargas útiles principales que son: un sistema de comunicaciones ópticas, un Detector de Tonos, una cámara digital y una arquitectura de cómputo semivirtual tolerante a fallas, sin embargo ya que se trata de una primera experiencia de diseño, integración y validación totalmente mexicana, cada subsistema construido para el microsatélite representa una carga útil adicional.

Para controlar el apuntamiento del satélite se cuenta con dos sistemas de estabilización, uno pasivo y uno dinámico, el pasivo conformado por un gradiente gravitacional, que tiene una masa de 2.5 kilogramos y una longitud de 6 metros cuando se encuentra totalmente desplegado. El dinámico esta constituido por 6 bobinas de torque magnético (BTM's) que generan pares de corrección en forma triaxial. Los datos requeridos para conocer la orientación del satélite, son obtenidos por medio de los siguientes sensores:

- 4 sensores burdos de sol, formados por los paneles solares.
- 4 sensores finos de sol, que permitirán una mayor exactitud en la determinación de la orientación del satélite.
- 2 magnetómetros triaxiales, uno principal y uno redundante.

En lo que respecta al subsistema de telemetría y comando se encuentra compuesto por un número importante de sensores de corriente, voltaje y temperatura, que nos

permitirán conocer el comportamiento de los diversos subsistemas del satélite, estos sensores serán colocados en lugares estratégicos como las computadoras, sistemas de comunicaciones, electrónica de acondicionamiento de señales, cargas útiles, baterías, paneles solares, etc. Los sensores requieren de un modulo electrónico que permita el acondicionamiento eléctrico de las señales, fuertemente ligado al sistema de comando principal para realizar de forma automática la adquisición y transmisión de los datos recabados. En vista de que la computadora debe además ejecutar algoritmos de control, regular las comunicaciones internas y con el segmento terrestre y ejecutar las misiones programadas desde el segmento terrestre, debe contar con una arquitectura capaz de soportar las interfaces requeridas por todos los subsistemas del satélite y que se constituya como una plataforma flexible para resolver los aspectos relacionados con su programación. Este sistema es la computadora de vuelo que es desarrollada en el Instituto de Ingeniería UNAM (IIUNAM).

Cabe señalar que otras de las características importantes de SATEX, es la tolerancia a fallas de la CV, por medio de software que controla los recursos redundantes de su arquitectura. Dentro del microsatélite SATEX se encuentra una red de área local con cinco nodos y 2 canales de red (uno principal y uno redundante). Los nodos están constituidos por los procesadores a bordo del vehículo, esto es, la computadora de vuelo (CV), la carga útil óptica (CUO), la cámara digital (CCD), el sistema de potencia (SP) y el decodificador de tonos o procesador de sobrevivencia (DT o SP). En esta red todos los nodos hablan y todos escuchan, y es controlada de forma centralizada por la computadora de vuelo, es decir, la CV funge como servidor en esta red de área local (RAL) [VICENTE, 1999].

1.3 Subsistemas a cargo del Instituto de Ingeniería UNAM

Actualmente, los sistemas que se encomendaron al Instituto de Ingeniería UNAM, se encuentran en su fase final de desarrollo, en lo referente al hardware de la computadora de vuelo, sensores y a la instrumentación, se encuentran prácticamente concluidos y en proceso de preparación para las pruebas finales previas a su integración. Solamente en caso de ser necesario se realizarán pequeñas correcciones, o en el mayor de los casos se realizará algún mejoramiento sin implicar cambios significativos.

En lo que respecta al software de vuelo y al software de estación terrena, ya están muy próximos a su conclusión, cabe señalar que gran parte de este software se encuentra funcionando en un prototipo comercial de la computadora de vuelo, que fue armado para la depuración del hardware y del software.

1.4 Requerimientos de Hardware y de Software para la operación y para los aspectos de tolerancia a fallas en la instrumentación del microsatélite

El desarrollo de sistemas a cargo del Instituto de Ingeniería UNAM ha requerido la elaboración de ciertas herramientas de hardware y de software para la depuración de los diseños de hardware y programación efectuados, así como para la validación de las técnicas de tolerancia a fallas propuestas.

Con el propósito de validar la electrónica de la computadora de vuelo se desarrolló un simulador de satélite (SIMSAT) por parte de otros Becarios-Tesistas del IIUNAM, que permite visualizar las acciones generadas por la computadora de vuelo, los estados de los actuadores del satélite (BTM's y BOOM), también permite simular a la mayoría de los sensores del satélite mediante potenciómetros con referencia a un voltaje o a una señal generada externamente. Además, SIMSAT contiene la electrónica necesaria para el control de los procesadores de la CV y electrónica necesaria para la simulación de ambos canales de la red de área local a bordo del satélite.

A pesar de que SIMSAT es una herramienta de gran ayuda para la validación de hardware y software, fue necesario desarrollar una herramienta de software para el monitoreo de las comunicaciones dentro de la red de área local del microsatélite, dicha herramienta recibió el nombre de SOFDEVO (Software de depuración y validación operativa) la cual fue desarrollada a su vez por parte de otros Becarios-Tesistas del IIUNAM [TORRES, 2002]. Esta herramienta además de monitorear las comunicaciones dentro de la RAL, emula la operación de todos los subsistemas dentro del satélite. Los subsistemas que SOFDEVO es capaz de emular son: CUO, CCD, SP, PS y CV, de tal forma que es posible validar las comunicaciones entre todos los subsistemas del satélite como si estos se encontraran ya todos construidos e interactuando entre sí, facilitando de esta forma el desarrollo y validación, no sólo de los sistemas encomendados al Instituto de Ingeniería UNAM, sino también de los subsistemas encomendados a otras instituciones, ya que no es necesario contar con la presencia de todos los subsistemas para la validación de las interfaces, lo cual garantizará una compatibilidad en cuanto a comunicaciones al momento de la integración de los mismos.

SOFDEVO, además de ser una herramienta para la validación de comunicaciones, es una herramienta para la validación de los algoritmos de control del satélite y para la arquitectura semivirtual tolerante a fallas del microsatélite [VICENTE, 2000].

1.5 Importancia de la implantación de técnicas de tolerancia a fallas dentro de los sistemas espaciales

Dentro del campo de la ingeniería existen ciertos sistemas cuyo funcionamiento adecuado es de suma importancia, ya sea por que de ellos dependen alguna o varias vidas

humanas como es el caso de los sistemas médicos, sistemas de aviación y sistemas militares; o bien sea por el costo que supone una falla en los mismos, tal es el caso de los sistemas de telecomunicaciones, sistemas de seguridad bancaria, sistemas de manufactura y sistemas espaciales en general.

Debido las características del medio ambiente en el cual opera, un sistema espacial se encuentra expuesto a diversos factores que ponen en riesgo su capacidad para llevar a cabo la función para la cual fue desarrollado. Este tipo de fallas conllevan por lo general la pérdida de grandes sumas de dinero ya sea simplemente por el costo del equipo y de la operación del mismo o por el efecto que dicho equipo pueda causar a los sistemas de los cuales forme parte.

Por lo anterior es de suma importancia contar con técnicas de tolerancia a fallas, que permitan al sistema operar de una manera aceptable por el mayor tiempo posible, así también, es de suma importancia que el diseño de los sistemas espaciales se realice considerando todas las necesidades ambientales y operativas que implica su operación.

A este respecto, algunas de las metas consideradas en el diseño del proyecto SATEX fueron: la funcionalidad a largo plazo, la confiabilidad de la operación y la posible aplicación a futuro de las tecnologías desarrolladas en el mismo.

Para lograr las metas mencionadas fue necesario plantear un diseño capaz de soportar las condiciones ambientales durante diversas etapas de la vida del vehículo espacial, además se optó por incluir diversas técnicas de tolerancia a fallas con el fin de extender la vida útil del sistema y mejorar la confiabilidad en la operación del mismo.

Como se ha mencionado, es importante que los sistemas espaciales utilicen técnicas de tolerancia a fallas, por ello, SATEX incluye un experimento que persigue realizar el mantenimiento automatizado de sus computadoras. No obstante que el experimento se realizará ocasionalmente, sus redundancias y sus propiedades de reconfiguración manual permitirán tolerar fallas importantes en la instrumentación del satélite. Al integrarse la arquitectura semivirtual cada una de las computadoras a bordo del satélite funciona como un procesador redundante independiente, que podrá ser utilizado para efectuar un voteo democrático de los diagnósticos de cada uno de los procesadores, voteo mediante el cual se tomarán las decisiones correspondientes en caso de que se detecte alguna falla en cualquiera de los procesadores a bordo del satélite. Por esto último es necesario contar con diversas técnicas de tolerancia a fallas para poder responder a las eventualidades que se presenten a lo largo de la vida útil del vehículo y así poder garantizar un periodo de funcionamiento en condiciones aceptables.

1.6 Objetivos del presente trabajo

En el caso de los sistemas a cargo del Instituto de Ingeniería la adaptación de técnicas de tolerancia a fallas es de suma importancia para el éxito de la misión, la cual

requiere alta confiabilidad y autonomía a pesar de las características del medio en el cual opera. Por estas razones la presente tesis se enfocará, como primer objetivo a implantar algunas de las técnicas de tolerancia a fallas, en particular las siguientes:

- Tolerancia a errores aislados en memoria RAM, debidos principalmente a la radiación, conocidos como "Single event upsets".
- Algoritmo de diagnóstico y detección de fallas en los procesadores y sus periféricos.
- Detección de fallas y reconfiguración por vigía de tiempo.
- Capacidad de envío y ejecución de software completamente nuevo para el satélite, aun después de que este haya sido orbitado, lo que implica la posible corrección y actualización de software para la computadora de vuelo aún después del lanzamiento.

El segundo objetivo de esta tesis, es elaborar un análisis de los diversos aspectos que deben ser tomados en cuenta durante el diseño del hardware de la computadora de vuelo, aspectos tales como la validación de la interconectividad entre componentes y entre tarjetas, la metodología de ensamble y pruebas de tarjetas electrónicas, así como la planeación de pruebas de hardware y de su software asociado. Para ello se utilizan herramientas de desarrollo asistido por computadora en las diversas fases de diseño de circuitos impresos, así como en el proceso de validación del hardware para la computadora de vuelo con el fin de cumplir con los requerimientos de funcionalidad en ambiente espacial.

Además, con este análisis, se persiguió detectar posibles fallas de diseño en la computadora de vuelo que ya se había desarrollado con anterioridad en el IIUNAM [MELO, 1996], así como errores de interacción entre los subsistemas del satélite y en su caso plantear y realizar los cambios pertinentes dentro del mismo. Todo esto con el propósito de terminar los equipos de vuelo finales para que el satélite cumpla sus funciones con éxito, dentro del tiempo requerido, además de servir como base de diseño para proyectos futuros que se apoyen en esta experiencia.

Capítulo 2. Algunos detalles de la instrumentación del SATEX vinculados con la tesis

2.1 Introducción

En el presente capítulo se señalan las características globales del hardware y del software desarrollado por el Instituto de Ingeniería UNAM para la instrumentación del microsatélite. El cual comprende la computadora de vuelo tanto en hardware como software, hardware de sensores, el software de estación terrena y el hardware y software de validación (SIMSAT y SEFDEVO respectivamente)[1]. Cabe señalar que todo esto conforma la plataforma sobre la cual se desarrolla el presente trabajo, y que dicha plataforma ha sido y continúa siendo desarrollada en el Instituto de Ingeniería UNAM.

Se efectúa una breve descripción de los aspectos de tolerancia a fallas de los sistemas espaciales y se describen las particularidades bajo las cuales ha sido diseñada la instrumentación del microsatélite, aspectos como redundancia de procesadores, redundancia de red de área local (RAL), etc. Se señalan los beneficios y utilidad en las diversas etapas de desarrollo y vida del proyecto.

También se presenta una breve descripción de las técnicas de tolerancia a fallas de las que se ocupa el presente trabajo, señalando los beneficios y necesidades a las que obedece su implantación.

Por último se efectúa una breve descripción del software de automatización de operaciones del microsatélite y del software de validación del mismo.

Con el conocimiento del hardware y del software del satélite se tendrán las bases necesarias para realizar el desarrollo e implantación de las técnicas de tolerancia a fallas de que se ocupa este trabajo.

2.2 Descripción global de la arquitectura de la computadora de vuelo CV-3PRO

La computadora de vuelo CV-3PRO (CV), es un sistema modular que en un principio estaba constituida por 7 tarjetas [MELO, 1996], [VICENTE, 1999] y que en su versión final esta constituida por 6 tarjetas, el cambio en número de tarjetas obedece a los cambios en la arquitectura que serán señalados más adelante.

Las 7 tarjetas iniciales estaban constituidas por 3 tarjetas de procesadores, uno principal y 2 redundantes, más tarjetas que contenían la electrónica de control, filtrado, recorte, conectores hacia el exterior y conmutación de procesadores. [VICENTE, 1996]. Actualmente la computadora de vuelo en su versión final, que incorpora las modificaciones

[1] Mas al respecto se puede consultar en [Vicente, 2002], [Vicente, 2000-1], [Vicente,2000-2], [Vicente 1999] y [Vicente 2001].

propuestas en la presente tesis cuenta con 3 tarjetas de procesadores; uno principal denominado CP, y dos redundantes CR0 y CR1, cabe señalar que los procesadores CP y CR0 pueden ser configurados para trabajar en modo de procesamiento paralelo. En la Figura 2.1 se presenta un diagrama de bloques de la arquitectura de la computadora de vuelo, a continuación se describen cada uno de los módulos ahí señalados:

Módulos de procesamiento (CP, CR0 y CR1): cada uno de los módulos de procesamiento se encuentra alojado en una tarjeta impresa denominada TCV, y cuenta con: un microcontrolador Siemens SAB 80C166, 7 temporizadores, controlador de interrupciones, 1 Kbyte de memoria RAM interna del microprocesador,64 Kbytes de memoria EEPROM externa al microprocesador, 1.186 Mbytes de memoria RAM externa protegida con una unidad de detección y corrección de errores en memoria ram (EDAC por sus siglas en inglés: Error Detection And Correctión unit) de 16 bits, 608 Kbytes de memoria RAM externa de síndrome para la corrección de errores en RAM, convertidor A/D de 10 bits de resolución con 10 canales de conversión, un sensor de temperatura y uno de corriente para la protección contra efecto "Latch-up". Cada una de estas tarjetas se conecta a la instrumentación exterior mediante el bus de instrumentación, que consta de 42 líneas formadas por líneas de control, líneas de adquisición de datos, y líneas de entrada y/o salida que son multiplexadas por el modulo de control y conmutación de procesadores, además cuenta con 13 líneas no multiplexadas que están formadas por líneas de Rx/Tx de la red principal interna, Rx externa, líneas de polarización de +5Volts y +12Volts y líneas de Tierra (GND) independientes para cada una de las tarjetas.

Modulo de Control y Conmutación de Procesadores (MCCP): este módulo efectúa la conmutación del bus de instrumentación hacia cada una de las tarjetas de procesadores logrando con ello la conexión de la instrumentación externa con el módulo de procesamiento operativo en un momento dado. Así también, el MCCP efectúa el encendido del o los módulos de procesamiento y de los módulos de electrónica de sensores pertinentes con el fin de lograr una mejor administración de la energía a bordo del satélite. Además adapta las señales de control para los actuadores del satélite (BTM's y BOOM). La electrónica asociada al MCCP se localiza en una tarjeta impresa denominada TCVCTRL con excepción del interruptor de estado sólido asociado al encendido del magnetómetro redundante que se encuentra localizado en la tarjeta denominada TCVLATCH.

Módulo de Acondicionamiento, Multiplexaje y Filtrado de señales de Sensores (MAMFS): Internamente la computadora de vuelo en su versión inicial contaba con 6 sensores de corriente y uno de temperatura, acondicionamiento para 14 señales de sensores externos y 28 entradas adicionales para sensores externos previamente acondicionados de 0 a 5 volts de CD, también contaba con canales libres de conversión A/D y líneas de polarización para sus periféricos. [VICENTE, 1996]. Actualmente la computadora de vuelo en su versión final cuenta con 6 sensores internos, 3 de corriente ("Latch-up") y 3 de temperatura, además de etapas de acondicionamiento filtrado y multiplexaje para señales de: sensores de corriente (9), sensores de temperatura (23), sensores finos de sol biaxiales (4), sensores burdos de sol (2) y sensores triaxiales de campo magnético (2), lo cual da un total de 48 señales de sensores, es importante señalar que estos no son todos los sensores a bordo del satélite, ya que el sistema de potencia está a cargo de 5 sensores de voltaje, 5 de

corriente y 3 de temperatura, los cuales son adquiridos por su computadora y cuyos valores son transmitidos a la computadora de vuelo mediante el uso de la red interna. Además, la computadora de vuelo cuenta con 1 sensor de temperatura y 1 de corriente por cada tarjeta de procesamiento, pero solamente la tarjeta conectada al bus de instrumentación puede ser monitoreada en su temperatura, los sensores de corriente de las tarjetas no se acondicionan ni multiplexan ya que solamente generan señales de presencia de efecto "Latch-up".

La electrónica asociada al MAMFS se localiza en una tarjeta impresa denominada TCVMUXFIL, este módulo proporciona el multiplexaje, acondicionamiento y filtrado para las 48 señales de sensores antes mencionadas.

Módulo de Protección de Efecto "Latch-up" (MPEL): la electrónica asociada con este módulo se encuentra en la tarjeta denominada TCVLATCH y en cada una de las tarjetas de procesamiento (sólo sensor de detección). El módulo genera las señales de control para el apagado de los módulos de procesamiento en el caso de que se detecte el efecto "Latch-up" en cualquiera de los procesadores de la CV, así también es el responsable de generar las señales de control para el encendido de los módulos de procesamiento, una vez que haya terminado el efecto "Latch-up".

Modulos de Electrónica de Red Interna (MERI): la electrónica de estos módulos se encuentra distribuida en las tarjetas TCVCTRL, TCVMUXFIL y TCVLATCH debido a razones de espacio, éstos dan soporte a las comunicaciones dentro del satélite mediante la implementación de una Red de Área Local (RAL) con 2 canales, uno principal y uno redundante totalmente independientes. Dichos canales de comunicación trabajan con una interfaz de especificación RS232 (+-12volts), la implementación de la RAL soporta comunicaciones de forma serial entre cada una de las computadoras a bordo del satélite inclusive de los módulos de procesamiento de la CV que se encuentren operando en forma paralela (solo canal principal).

Módulos de Electrónica para Comunicaciones Externas (MECE): de forma similar a la electrónica de la red interna, ésta electrónica se encuentra distribuida en las tarjetas TCVCTRL y TCVLATCH. Proporciona el enlace con los sistemas de comunicación con tierra (VHF1 y VHF2) mediante una interfaz del tipo RS232 con dos canales independientes, como se puede observar en la figura 2.1, las señales recibidas desde tierra no son multiplexadas, sino se envían a cada uno de los módulos de procesamiento, por lo cual, es posible que los módulos que se encuentren trabajando en forma paralela escuchen las señales provenientes del segmento terrestre; no siendo así con la transmisión a tierra ya que solo podrá ser efectuada por aquel módulo de procesamiento que se encuentre conectado al bus de instrumentación.

Con la excepción de los microcontroladores, los componentes electrónicos de la CV son CMOS de tipo militar para asegurar cierta tolerancia de operación en ambiente espacial. Los componentes militares aseguran su funcionamiento en el rango de temperatura de –55 a 125 °C y por otro lado la tecnología CMOS soporta los niveles de radiación esperados en la órbita del microsatélite.

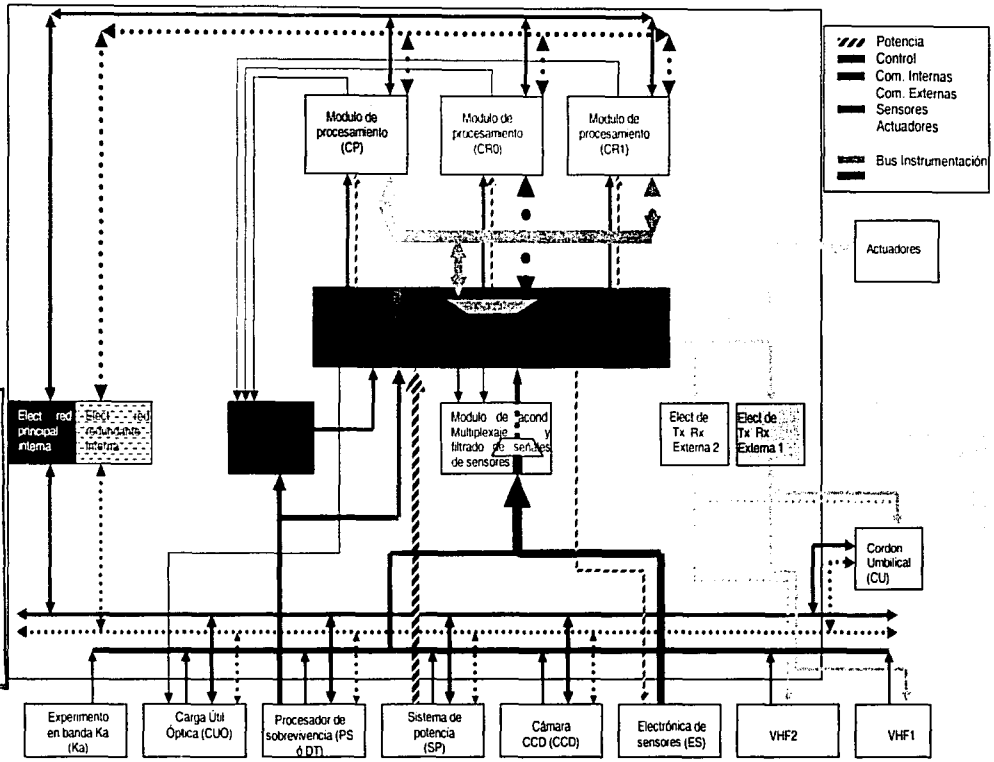


Figura 2.1: Arquitectura de la CV-3PRO.

2.3 *Algunos conceptos de tolerancia a fallas en sistemas espaciales*[2]

Existen diferentes formas en las cuales el hardware y/o el software de los sistemas espaciales pueden fallar. Algunas de estas fallas conocidas como "fallas duras" (Hard Failures), debido a que una vez que ocurren continúan teniendo efecto. Otro tipo de fallas son conocidas como "fallas suaves" (Soft Failures), lo que significa que la falla solo causa efectos una sola vez y después de eso el sistema retorna a su operación normal. En la medida de lo posible los sistemas espaciales deben ser capaces de tolerar ambos tipos de fallas. Las técnicas de diseño que se emplean en el desarrollo de estas capacidades son conocidas bajo el término genérico de Tolerancia a Fallas (TF).

El diseño de sistemas TF es muy complejo y extenso por lo cual en esta sección se presentan sólo aquellos conceptos requeridos para explicar los desarrollos elaborados para esta tesis. En cuanto a la bibliografía del área de TF, ésta es muy amplia, sin embargo, tanto [JOHNSON, 1989] como [PISACANE, 1994] presentan panoramas claros y objetivos del área.

2.3.1 Radiación espacial

La radiación encontrada en el ambiente espacial, puede dañar los dispositivos semiconductores de tal forma que su rendimiento decae con el tiempo. Los efectos producidos por la radiación son usualmente una función de la dosis total de radiación experimentada por cada uno de los chips de semiconductores. A medida que la dosis de radiación se incrementa, la Beta de los transistores disminuye, los voltajes de disparo cambian, ocurre una degradación en la movilidad del canal en los dispositivos de semiconductor óxido metálico (MOS), las corrientes de fuga se incrementan, etc. Esto se traduce en que los dispositivos de lógica digital se hagan más lentos, cambios en los voltajes de offset de los amplificadores operacionales, decaimiento de la capacidad de conducción de corriente, mayor disipación de potencia, etc.

Los rayos gama pueden pasar a través de un circuito integrado con muy poca pérdida de energía, dejando una estela de carga eléctrica a su paso. La mayoría de estas cargas se recombinan con cargas opuestas y desaparecen, sin embargo, algunas quedan atrapadas en la capas límites de los circuitos integrados, que se pueden acumular y causar cambios en las características de operación de los dispositivos. Este efecto es una función de la dosis total acumulada de radiación gama, que es medida en *rads*. Otras formas de radiación, principalmente de las partículas cargadas que impactan con el vehículo en órbita, causan efectos acumulativos similares.

[2] Para encontrar más información sobre este tema consulte [Pisacane, 1994] y [Johnson, 1989]

Un *rad* es la cantidad de cualquier forma de radiación ionizante que impartirá 100 *ergs* de energía a un gramo de materia, en electrónica el material más común de fabricación de circuitos es el Silicio, por tanto, la unidad de medida para las dosis de radiación es el *rad* silicio, abreviado "rad(Si)".

La radiación recibida por los componentes electrónicos en el espacio es esencialmente el resultado de neutrones y partículas cargadas, los neutrones son partículas altamente energéticas que se comportan como pequeños proyectiles que causan daños estructurales en los materiales de estado sólido de los circuitos integrados. Las partículas cargadas son electrones energéticos, protones, partículas alfa, o iones que pueden pasar a través de un dispositivo electrónico y generar una nube de carga electrónica, si esta carga aparece por ejemplo, en la compuerta (Gate) de un transistor MOS, puede producir un evento de alteración aislado, conocido como "SEU" por sus siglas en inglés "Single Event Upset", que puede tener serias consecuencias en la operación del sistema.

2.3.2 "Single-Event Upsets" (SEU's)

Debido a las dimensiones tan pequeñas de los dispositivos electrónicos integrados modernos, una partícula cargada de alta energía, puede depositar a su paso suficiente carga para cambiar, digamos, un dígito binario almacenado. Cuando esto sucede un bit guardado en RAM puede cambiar, esto no suena muy significativo, pero si tomamos en consideración que esto puede convertir una instrucción de algún programa almacenado en alguna otra instrucción, puede generar caos en todo el sistema.

Ya que es imposible predecir el tiempo o lugar en el que un SEU ocurrirá, el diseño del sistema debe asumir que este puede ocurrir en cualquier parte y momento. Los SEU's pueden ser detectados y corregidos usando códigos de detección y corrección de errores (EDC), vigías de tiempo ("Watchdog Timers"), retornos en falla (Fult Rollback) y otros métodos.

2.3.3 Efecto "Latch-up"

Con los dispositivos de lógica CMOS puede ocurrir un fenómeno llamado Efecto "Latch-up", si se presentan corrientes parásitas de suficiente magnitud en el dispositivo, causando que fluyan altas corrientes provenientes de la fuente de polarización a través del dispositivo. Estas altas corrientes usualmente provocan la destrucción del dispositivo en pocas decenas de milisegundos.

En la figura 2.2 se presenta un diagrama de corte transversal de un dispositivo CMOS típico, las partes sombreadas presentan el óxido metálico generado en el substrato de silicio, note los dos símbolos de transistores bipolares mostrados, estos dos transistores son una parte intrínseca del procesamiento CMOS. Normalmente los dos transistores

bipolares se encuentran en estado de corte, sin embargo, si suficiente corriente parásita se presenta fluyendo a la base del transistor *npn* o de la base del transistor *pnp*, el transistor correspondiente comenzará a conducir. La corriente de colector de cada uno de los transistores se suma con la corriente de base del otro, al comenzar ésta *regeneración* de corriente el circuito bipolar se pone en cortocircuito a través de la fuente de alimentación.

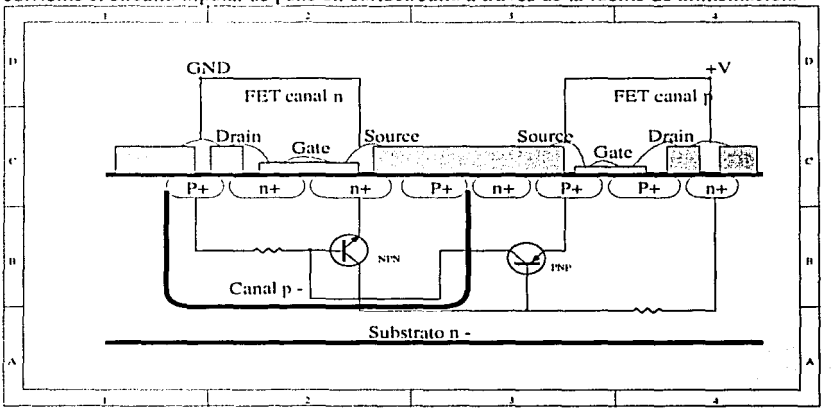


Figura 2.2: Estructura típica de un dispositivo CMOS.

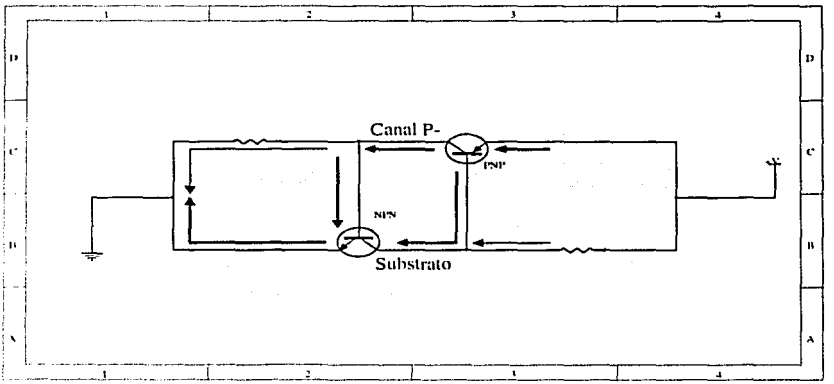


Figura 2.3: circuito de efecto Latch-up.

El circuito afectado por el efecto "Latch-up" (figura 2.3) se comporta de una manera similar a un SCR. Capaz de hacer fluir de la fuente varios amperes, lo que es catastrófico para el dispositivo CMOS. Si el flujo de corriente al dispositivo CMOS es interrumpido, o reducido por debajo de un cierto valor de extinción, la condición de "Latch-up" desaparece. Si la corriente de "Latch-up" no ha causado algún daño irreversible, el dispositivo entonces funcionará correctamente.

Si el límite de corriente es lo suficientemente bajo, el efecto "Latch-up" no destruirá el componente, entonces el efecto "Latch-up" puede ser eliminado mediante el apagado del dispositivo afectado y su posterior encendido.

2.3.4 Paridad

Una forma de detectar un SEU en una memoria es el seguimiento de la paridad de cada una de las palabras almacenadas, esto es conocido como chequeo de paridad, por ejemplo cuando una palabra se escribe en memoria, su paridad se calcula y se almacena junto con los bits de datos. Entonces cuando se lee la palabra, se calcula nuevamente la paridad y si esta es diferente a la anterior se asume que ha ocurrido un SEU.

También es posible implementar chequeos multidimensionales de paridad para localizar y corregir el bit erróneo, sin embargo, el chequeo de paridad se torna rápidamente complicado cuando se requiere para la detección de múltiples errores.

2.3.5 Detección y corrección de errores

Existen otras técnicas más sofisticadas que permiten detectar y corregir errores múltiples en los datos, como los chequeos de redundancia cíclica y los generadores de síndrome. Este método es atractivo debido a que, se corrigen los errores introducidos por la misma circuitería de chequeo de errores. Estos dispositivos son conocidos como "self-checking checkers".

Por el momento solo se hace la mención de estos dispositivos de detección y corrección de errores, en un capítulo posterior se hará un análisis más profundo, en particular del dispositivo que se empleará en la CV.

2.3.6 Técnicas de redundancia

Existen bastantes técnicas comprobadas para realizar el diagnostico y detección de fallas en sistemas de procesamiento, una de las más conocidas es la redundancia modular triple, la cual requiere bastante hardware, aspecto que la limita de forma determinante en

aplicaciones espaciales. Existen otros arreglos como el de maestro y esclavo, que requiere de hardware adicional para comparar resultados y para identificar fallas. Adicionalmente existen otros como el par y repuesto, similar al anterior pero con mayor capacidad de tolerancia a fallas.

Un aspecto importante de los sistemas TF es que pueden utilizar refacciones activas o pasivas (encendidas o apagadas, en caliente o en frío, etcétera). Para la CV se utilizan refacciones en frío para reducir el consumo de energía en el satélite.

2.3.7 Reconfiguración por vigía de tiempo "Watchdog Timer"

Un "Watchdog Timer" es un temporizador por hardware que permite identificar estancamientos en la ejecución de programas. En modo normal de operación el Watchdog se configura periódicamente, entonces cuando ocurre algún modo anormal de operación, la reconfiguración no se efectúa. Cuando el Watchdog alcanza su valor extremo debido a que no se reconfiguró, provoca un restablecimiento del sistema.

El Watchdog es un método simple de prevenir que los SEU's causen que el procesador entre en un ciclo de ejecución infinito. Sin embargo las fallas que continúen restableciendo el Watchdog no pueden ser detectadas, así mismo una falla en el propio watchdog puede volver al procesador entero inservible.

2.4 Particularidades de la instrumentación tolerante a fallas del microsatélite SATEX-1

2.4.1 Energización de la computadora de vuelo

Un aspecto de suma importancia dentro de los sistemas espaciales, y en particular en los microsatélites, es el referente a la distribución y ahorro de energía. La instrumentación TF de SATEX ha sido diseñada tomando previsiones a este respecto.

La energización de la computadora de vuelo se efectúa por módulos, de tal manera que al realizar una tarea dada solamente los módulos necesarios permanecen encendidos.

Al diseñarse los comandos que el segmento de control terrestre puede enviar al vehículo espacial, se identificaron los diferentes bloques de electrónica que pudieran ser encendidos de manera independiente para la realización de diversas tareas, tales como: lectura de sensores de corriente, voltaje o temperatura; lectura de magnetómetros y sensores de sol; etc. De tal forma se llegó a la configuración modular de distribución de energía de la CV, mostrada en la figura 2.4.

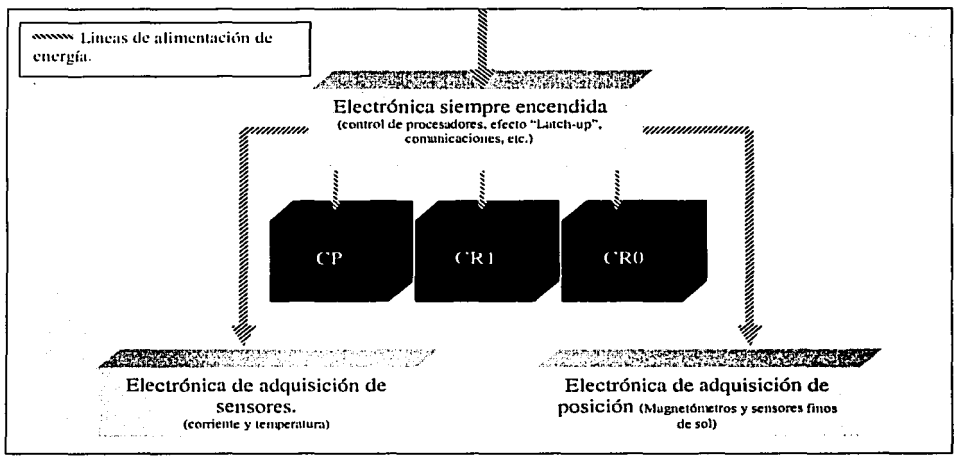


Figura 2.4 Distribución de energía de la CV

2.4.2 Circuito de protección de efecto "Latch-up"

Como ha sido mencionado, el efecto "Lacth-Up" puede contrarrestarse con las propias técnicas de diseño y manufactura del circuito, tal es el caso de los C.I. de especificaciones militares y espaciales; o por circuitos de protección externa.

En el caso de la CV, todos sus circuitos son de características militares, con excepción de los microprocesadores. Por ello se requirió de un circuito de protección externo para asegurar el correcto funcionamiento del procesador bajo condiciones de radiación espacial.

Dicho circuito consiste de un sensor de corriente, que detecta la elevación en el consumo nominal de corriente del microprocesador, y un temporizador que efectúa el apagado y el posterior encendido del procesador cuando se detecta el efecto "Latch-up".

En cada una de las tarjetas de procesadores se encuentra el sensor de Detección de efecto "Latch-up". La salida del sensor se lleva a través de las diferentes tarjetas de electrónica, hasta la tarjeta TCVLATCH, donde se encuentra el temporizador para la eliminación del efecto "Latch-up", dicho temporizador inhibe el encendido de las tres

tarjetas de procesador de la CV, con lo que el MCCP apaga los 3 microprocesadores sin importar en cual de ellos fue detectado el efecto "Latch-up".

2.4.3 Redundancia de procesadores

Debido a que el procesador recibe la carga de las funciones más importantes del satélite, existen tres tarjetas de procesador, una principal y dos redundantes. La tarjeta de procesamiento es el único modulo dentro de la computadora de vuelo que cuenta con doble redundancia, debido a su importancia y a que contiene el único circuito integrado que no es de calificación militar o espacial.

2.5 Descripción del software desarrollado para el microsatélite

Como se mencionó en el capitulo anterior, bajo el proyecto SATEX se han desarrollado diversos programas de software la operación del microsatélite, a continuación se presentará una breve descripción de cada uno de ellos.

2.5.1 Software de operación satelital para la computadora de vuelo

Este software junto con el software de estación terrena permiten realizar el control distribuido y la automatización del satélite. El segmento espacial del software de automatización del vehículo (software de vuelo), cubre funciones de control y automatización de tareas a bordo como: adquisición de telemetría, encendido y apagado de equipos, control de red interna, control de comunicaciones con Tierra, verificación de tiempos de ejecución, control de bobinas de torque, entre otras.

El software de vuelo de la computadora es una parte esencial del esquema de tolerancia a fallas propuesto para SATEX. Adicionalmente integra software dedicado para conformar experimentalmente la arquitectura de cómputo semivirtual a bordo de SATEX, cuyo propósito principal es realizar mantenimiento automatizado a la CV. La arquitectura realiza la ejecución del voteo entre los procesadores a bordo, la reconfiguración automática en caso de errores no corregibles en memoria RAM, la conmutación automática de canales de comunicación ya sea en la red interna o en la comunicación con Tierra en el caso de detectar alguna falla, etc.[VICENTE, 2001]

El software de vuelo del microsatélite fue desarrollado en lenguaje C ANSI y con algunas utilerías propias del microcontrolador SIEMENS SAB80C166. El compilador usado fue el "TASKING Cross C Compiler Ver. 3.5.6" [TASKING, 1993] de la compañía TASKING [www.tasking.com], dicho compilador proporciona las extensiones de lenguaje necesarias para el control de puertos, periféricos y registros del microcontrolador.

La programación desarrollada para la computadora de vuelo se puede dividir en tres módulos principales, la primera es la relacionada con las tareas de control y automatización de funciones a bordo, la segunda referente al control de comunicaciones de la red interna de SATEX y la última relacionada con el software de comunicaciones con la estación terrena [VICENTE, 1998].

2.5.2 Software de Estación Terrena (ET)

Este software representa el segmento de Tierra del software de control y automatización de funciones de SATEX, y constituye la interfaz del vehículo espacial con sus usuarios. El software de ET permite el enlace de comunicaciones con el satélite, permite presentar, almacenar y estudiar los datos recabados de telemetría, permite efectuar las tareas de mantenimiento y validación de operación de los equipos del satélite y permite la ejecución de tareas de reconfiguración ante posibles eventualidades.

Es importante mencionar que el software de ET, presenta una interfaz con grandes avances en comparación con las de otros microsatélites desarrollados por grupos como AMSAT, entre tales avances se encuentran 3 muy importantes que son:

- Presentación de telemetría en modo gráfico y numérico con activación de alarmas en caso de que excedan el rango de operación normal programado.
- Presentación del estado de operación de los diferentes equipos a bordo del vehículo mediante diagramas, gráficos y animaciones.
- Vinculación con un sistema experto para el análisis de telemetría y apoyo en el control de misiones.

Además de las características de vanguardia mencionadas, se mencionan las siguientes:

- Formación de una base de datos con el historial de eventos y datos de telemetría de las diferentes misiones.
- Capacidad de envío de nuevos programas o inclusive de nuevos sistemas operativos para la CV

Todos estos avances están enfocados a hacer el trabajo con SATEX, mas sencillo, seguro e independiente del usuario, algo que otros sistemas microsatelitales aún no han incorporado. Esto influye directamente en el costo de operación del satélite después de su lanzamiento, principalmente debido a que por su sencillez no será necesaria la contratación de personal especializado de tiempo completo para la operación del vehículo. Esto también abre la posibilidad de llevar a cabo convenios con otras instituciones nacionales o internacionales para el uso de los experimentos.

El software de estación terrena fue desarrollado con "MS Visual Basic 6.0 for Windows", por sus atributos de ayuda en el manejo de interfaces gráficas, manejo de los puertos de comunicaciones, manejo de bases de datos e integración con el WEB. En la figura 2.5 se presenta una de las pantallas del software de estación terrena.

2.5.3 Software de validación operativa (SOFDEVO)

Para validar el software de red y la arquitectura de computo semivirtual, fue necesario desarrollar una herramienta de validación, aún cuando los equipos satelitales no estuviesen completamente terminados. De esta forma se desarrolló un software para llevar a cabo dicha tarea, el software recibió el nombre de SOFDEVO (Software de Validación Operativa), las características principales de este software son:

- Soporte completo para los protocolos de cómputo tolerante a fallas a bordo de SATEX.
- Capacidad para emular a todos y cada uno de los procesadores de abordo en cuanto a su interacción con la red interna, inclusive a la computadora de vuelo.
- Capacidad para monitorear el tráfico de la red interna y efectuar la presentación cualitativa de los sucesos ocurridos en la misma.
- Capacidad de emular diferentes fallas en los procesadores de abordo.
- Interfaz gráfica de usuario sencilla e intuitiva, basada en "MS Windows".

El programa SOFDEVO fue desarrollado usando "MS VISUAL BASIC 6.0 FOR WINDOS", debido a la facilidad en el manejo del ambiente WINDOWS, facilidad en el manejo de los puertos de comunicaciones, programación estructurada, etc.

Además de servir para la validación del hardware y del software de la computadora de vuelo, SOFDEVO será útil para la integración final de los equipos en el satélite y para la evaluación de las operaciones del vehículo en sitio de lanzamiento. Debido a que puede emular a todas las computadoras de la red interna será distribuido a todos los equipos que tengan a su cargo el desarrollo de éstas, para que puedan efectuar la validación de forma lógica de su programación y de los protocolos de red.

En la figura 2.6 se muestra una de las pantallas de trabajo de SOFDEVO.

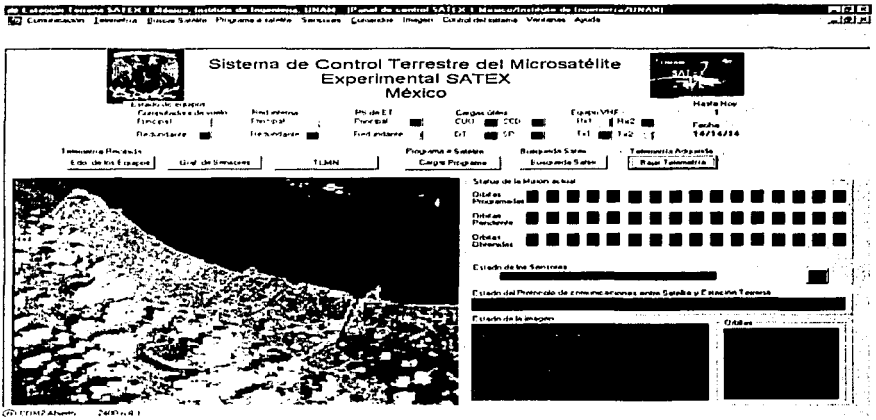


Figura 2.5: Pantalla principal del software de estación terrena.

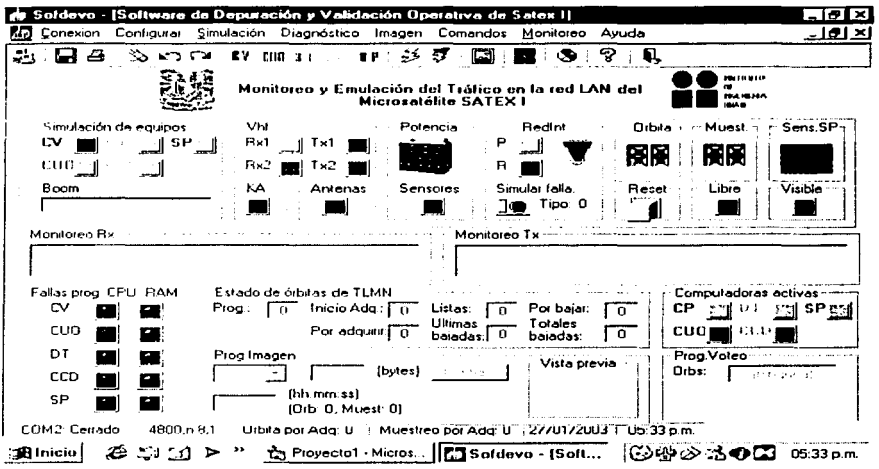


Figura 2.6: Pantalla principal de SOFDEVO.

Capitulo 3. Desarrollo de una técnica para el cargado y ejecución de nuevo software para el satélite

3.1 Introducción

En los sistemas espaciales es de suma importancia contar con capacidades que permitan la recuperación ante las fallas que se presenten durante la vida del vehículo, una de estas es la capacidad de cambiar en forma parcial o total el software de control del vehículo. Es muy común que esta característica se encuentre en equipos electrónicos aún cuando estos no sean de tipo espacial, ya que presentan una gran ventaja en cuanto a la flexibilidad en el desarrollo de software ya sea para mejoras o adaptaciones o para la corrección de errores de programación no detectados durante el proceso de desarrollo del mismo.

En lo que respecta al proyecto SATEX, esta característica permitirá la actualización total o parcial del software de la computadora de vuelo, proporcionará un tiempo de desarrollo más prolongado, mientras que el software de operación satelital siga siendo objeto de desarrollos en Tierra.

En este capítulo se hace un breve análisis de la técnica con que se contaba en la versión anterior de la computadora de vuelo, se presenta un análisis más profundo de la técnica alterna propuesta en esta tesis y que es incorporada en la versión final de la CV. También se destacan sus ventajas y desventajas con respecto a la técnica adoptada inicialmente, y se presentan los cambios en el hardware que conlleva su implantación.

3.2 Técnicas propuestas para el cargado y ejecución de nuevo software

Durante el desarrollo de la computadora de vuelo se han propuesto dos técnicas para el cargado de nuevo software al satélite, la primera se basa en el uso del mismo software de control de vuelo de la CV y la segunda se basa en el modo "BOOTSTRAP LOADER" del microcontrolador SAB80C166 [SIEMENS, 1997]. La primera técnica fue utilizada en la computadora de vuelo en versiones anteriores, y la segunda técnica fue desarrollada en la presente tesis; a continuación se presenta el análisis de cada una de las técnicas antes mencionadas.

3.2.1 Técnica de cargado y ejecución por medio del software de la computadora de vuelo

Esta técnica se basa en el uso del software de control de la CV, funciona de manera tal que la computadora de vuelo recibe un comando enviado desde el segmento terrestre, que le indica tamaño del nuevo programa y el número de los paquetes de código que serán transmitidos. Es importante mencionar que esta técnica no permite efectuar un reemplazo total del programa de vuelo, porque la rutina esta contenida en el mismo software de vuelo que es necesario para efectuar el cargado y la relocalización de los vectores de interrupción del código recibido, además de que el espacio en memoria para dicho programa es fijo. De forma general la técnica opera de la siguiente manera:

- El segmento terrestre envía el comando de carga de nuevo programa, dicho comando contiene la información del tamaño y del número de paquetes del nuevo programa.
- El segmento terrestre comienza el envío de paquetes de código del nuevo programa, la computadora de vuelo recibe dichos paquetes y los almacena en un espacio de memoria predefinido e inalterable.
- Al término del envío de los paquetes, el segmento terrestre envía información de la localización de los nuevos vectores de interrupción y la CV efectúa la relocalización de éstos, y toma conocimiento de la ejecución del nuevo programa.
- El software de la CV efectúa la ejecución del nuevo programa mediante un salto al inicio del código, el software de control de la CV debe efectuar un mapeo en tiempo real de los nuevos vectores de interrupción, debido a que, los vectores de interrupción anteriores no fueron sobre escritos.

El último punto mencionado ofrece la mayor limitante para que el software de la computadora de vuelo no pueda ser actualizado en su totalidad una vez que el satélite haya sido lanzado.

3.2.2 Técnica de cargado y ejecución con base en el modo "BOOTSTRAP LOADER" (BTL) del microcontrolador SAB80C166

Esta técnica se basa en una característica muy común en la mayor parte de los microcontroladores, que les permite establecer comunicación con una estación externa, de manera independiente del programa contenido en la memoria del mismo. Se utiliza ampliamente en los sistemas de desarrollo y de evaluación para efectuar la carga y ejecución de nuevos programas y en los sistemas de usuario final para su mantenimiento, actualización y servicio.

Entre los microcontroladores que emplean esta característica se encuentran los MC68HC11 –HC05 de Motorota [www.mototrola.com], los TMS320C5X de Texas

Instruments [www.ti.com], y los SAB8XC166X de Siemens [www.siemens.com], entre muchos otros.

En este caso, el microcontrolador utilizado fue el SAB80C166 de SIEMENS, que presenta esta característica mediante el modo "BOOTSTRAP LOADER" (BTL). Para colocar al microcontrolador en el modo BTL, es necesario el accionamiento de diversos mecanismos que serán descritos posteriormente en este capítulo.

De manera general se presenta a continuación la técnica de carga y ejecución de programas basada en el modo BTL

1. El microcontrolador es puesto en modo BTL.
2. La estación externa envía una secuencia binaria por el canal de comunicación, y espera una respuesta de reconocimiento por parte del microcontrolador, este proceso es conocido comúnmente como "Handshake".
3. La estación externa envía un bloque de código de tamaño predefinido, que después ejecuta el microprocesador, este bloque recibe el nombre de "Código de precarga o precargador ".
4. Al ejecutar el código "precargador" el microcontrolador recibe un segundo bloque de código con la finalidad de ejecutar las rutinas correspondientes de configuración de periféricos y efectuar la carga del programa monitor de memoria, este segundo bloque de código recibe el nombre de "Cargador de Monitor".
5. Al ejecutarse el "Cargador de Monitor", efectúa la configuración de la memoria externa y de los periféricos necesarios para la carga y ejecución del siguiente bloque de programa denominado "Monitor de Memoria", y lo carga.
6. En el microcontrolador se ejecuta el programa "Monitor de Memoria", que permite una comunicación más segura para la transmisión del programa final o aplicación y permite efectuar algunos comandos para la administración de la memoria del microcontrolador.
7. Al terminar la carga del programa, el microcontrolador se configura para ejecutar el programa recién cargado en su memoria.
8. Se procede a la ejecución del programa mediante un reset por Hardware.

Como se puede observar en los párrafos anteriores, el envío del código de nuevo programa se efectúa en 3 bloques: "precargador", "cargador de monitor" y "monitor de memoria", se hace de esta forma debido a las limitaciones que presenta el microcontrolador SAB80C166 al trabajar en el modo BTL.

3.3 Descripción del modo "BOOTSTRAP LOADER" (BTL) del microcontrolador sab80c166

3.3.1 Introducción al modo "BOOTSTRAP LOADER"

El microcontrolador SAB80C166, contiene un cargador de arranque "BTL" (On Chip Bootstrap Loader). El código de BTL está alojado en una memoria ROM de arranque especial. Con el BTL es posible cargar un programa de 32 bytes en la memoria interna del SAB80C166 (1kbyte) vía el puerto serial 0, aunque no exista un programa disponible en memoria interna o externa. Este pequeño programa puede utilizarse para cargar software de usuario o aplicaciones más amplias a memoria RAM interna o externa [SIEMENS, 1993].

3.3.2 Operación general del "Bootstrap Loader"

Para la activación del BTL, es necesaria la intervención de un dispositivo de Hardware y de un mecanismo de software, que efectúe las siguientes acciones:

- Activación de la ROM de Arranque del microprocesador.
- Ejecución de la rutina de Arranque contenida en la memoria mencionada.
- Ejecución del código recibido en la rutina anterior.

El BTL se activa al final de un reset por HARDWARE si el pin ALE del microcontrolador se muestrea en nivel alto (+5Volts), y si el pin /NMI se activa directamente después de que termina la secuencia interna de reset. En este modo de BTL el SAB80C166 espera la recepción serial de un byte cero (un bit de inicio, un dato 00xH, un bit de parada, sin bits de paridad) proveniente de un anfitrión en el pin RxD0 (p3.11), a partir del cual el microprocesador calcula el factor necesario para el generador de baudaje, tomando en cuenta la frecuencia del reloj del CPU.

De acuerdo con el baudaje calculado, se inicializa el puerto serial 0 (ASC0) del microcontrolador (un bit de inicio, 8 bits de datos, un bit de parada, sin paridad), y se envía un byte de identificación, 0x55h. Después de enviar el byte de identificación, el BTL se va a un ciclo de recepción, esperando recibir exactamente 32 bytes de código desde el anfitrión. Si se reciben menos de 32 bytes el SAB80C166 espera por siempre, pues no se efectúa ninguna comprobación de tiempo. Los bytes recibidos se almacenan de forma secuencial en la memoria RAM interna a partir de la localidad 0xFA40h, y terminando en la localidad 0xFA5Fh. Después de la recepción de los 32 bytes, el BTL automáticamente efectúa un salto a la localidad 0FA40xH, y se ejecuta el programa cargado. Figura 3.1.

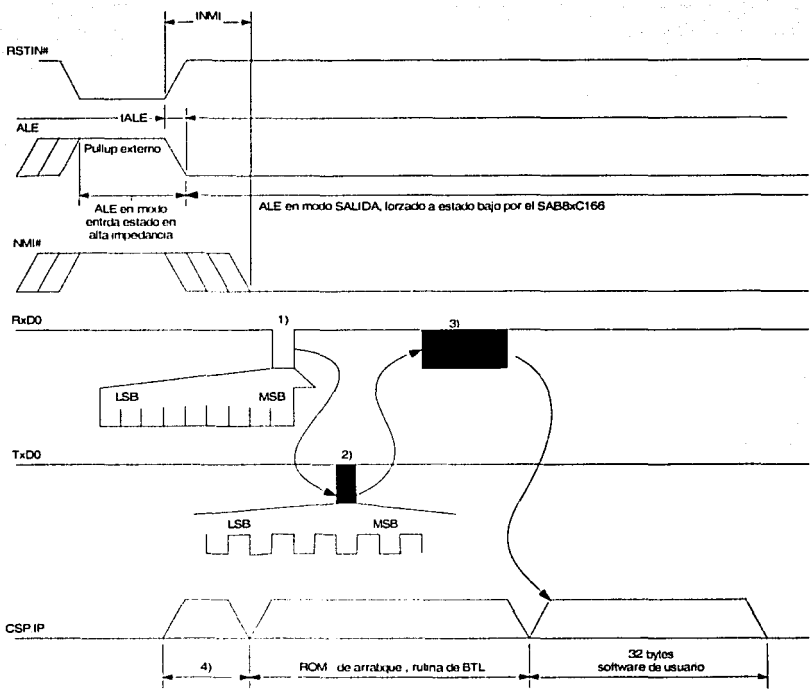
3.3.3 Características especiales del SAB80C166 en el modo BTL

Cuando el SAB80C166 inicia en modo BTL, cuenta con una configuración específica [SIEMENS, 1993], para la CV los aspectos más importantes que deben tomarse en cuenta corresponden a la configuración de los accesos a memoria y la configuración del Watchdog.

Con la excepción del "Watchdog", el sistema puede re programarse a la configuración deseada después de ejecutar la rutina de BTL. El "Watchdog" solamente puede habilitarse después de un reset ya sea por software o por Hardware.

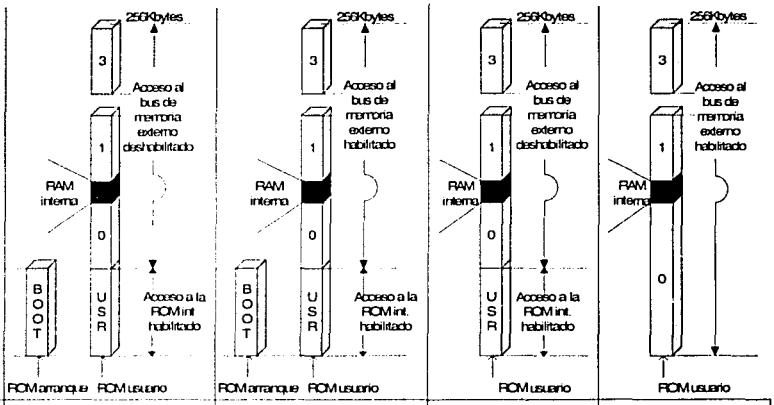
Si el SAB80C166 se encuentra en el modo BTL, todas las lecturas de código (code fetch) de las direcciones 00000xH a la 07FFFxH del dispositivo se efectúan desde la memoria ROM de arranque interna. Las lecturas de código de esta área no se permiten al usuario y generarían comportamientos inesperados. Todas las lecturas de datos, por parte del usuario se deben efectuar en el área de la memoria RAM interna del SAB80C166, ver figura 3.2. Solo después de un reset por software o hardware (sin la activación del modo BTL), todos los accesos de código y datos a la memoria de las direcciones 00000xH a 07FFFxH se efectuarán a partir de la memoria RAM de usuario y no de la ROM de arranque.

La figura 3.2 muestra diferentes alternativas de configuración para la memoria del sistema mediante los pines /BUSACT, EBC0 y EBC1 durante un reset, y las diferencias principales en cuanto al acceso a memoria entre un reset normal y un reset bajo el modo BTL. Para efectuar un acceso al bus de memoria externo en el espacio del segmento 00 es necesario ejecutar una secuencia de comandos para la desactivación de la ROM interna ya que la versión del microprocesador de la CV no posee memoria ROM de usuario interna [SIEMENS, 1993], en este caso se ejecuta dentro la rutina de carga de monitor, mencionada anteriormente [Apéndice A].



1) Byte ZERO, Bit de inicio,byte de dato 00h y un bit de parada desde el huesped
2) Byte de identificación desde el SAB8xC166 55h
3)32 Bytes desde el huesped
4) ROM de Arranque interna
tALE (0-0.25)Microseg con reloj de CPU a 20MHz
tNMI (0-1.2) miliseg con reloj de CPU a 20MHz
tNMI(max)<=25900 tCPU

Figura 3.1: Secuencia de arranque en el modo BTL.



Modo BTL activo	Si	Si	No	No
Pin EBC0	bajo	x)	bajo	x)
pin EBC1	bajo	x)	bajo	x)
pin BUSACT#	alto	bajo	alto	bajo
Code fetch ROM	acceso a ROM de arranque 2)	acceso a ROM de arranque 2)	acceso a ROM de usuario 1)	——
Data fetch ROM	acceso a ROM de usuario 1)	acceso a ROM de usuario 1)	acceso a ROM de usuario 1)	——
Configuracion nº	I)	II)	III)	IV)

X) Depende de la configuración del bus de memoria externo
1) Todos los accesos al espacio de direcciones interno (0'00000h-07FFFh) de una versión sin ROM interna del SAB8xC166 resultan en valores indefinidos.
2) No se permite, pueden ocurrir resultados inesperados.

Figura 3.2: configuración de accesos a memoria después de un reset.

3.3.4 Software de usuario cargado con el BTL

Normalmente un programa requiere más de 32bytes. Entonces, con el propósito de cargar rutinas más grandes, los 32bytes de programa cargados vía el BTL serán en la mayoría de los casos un precargador para otras rutinas que ahora tendrán direcciones de inicio y final definidas por el usuario. Este precargador utilizarse para cargar software de usuario a la RAM interna o a la memoria externa que deberá inicializarse para este fin.

3.4 Implantación de la técnica de cargado de programas en la computadora de vuelo mediante el modo BTL

En los diseños previos de la computadora de vuelo no se utilizaba este esquema para la carga de programas. Por esto fue necesario actualizar tanto el Hardware como el software, de tal manera que permitiera la operación de ambos esquemas de carga de programas. Las adaptaciones de Hardware se enfocaron a la decodificación de los mapas de memoria de la CV y al circuito de activación del modo BTL.

3.4.1 Descripción de los mapas de memoria de la CV

En la CV existen 2 mapas de memoria diferentes, uno que controla memoria ROM y RAM, y el otro que controla solamente memoria tipo RAM.

El mapa de memoria que controla memoria ROM, permite direccionar un programa grabado de forma previa al ensamble y lanzamiento del satélite, esto sirve como un método de protección en el caso de que el programa cargado en memoria RAM presente errores incorregibles por efecto de la radiación. A continuación se presenta un diagrama del mapa de memoria 1 (Con memoria tipo ROM y RAM). Cuando éste mapa se encuentra activado, la CV cuenta con los siguientes recursos de memoria:

- 63Kbytes de Memoria ROM externa, usada para almacenar el software de operaciones de SATEX, el que se utilizará siempre como un modo seguro de software.
- 1 KB de Memoria RAM interna, usada por el microprocesador para almacenar registros de trabajo y configuración de sistema, y para la implementación de la pila o "Stack". El espacio restante puede ser administrado libremente por el usuario.
- 192 KB de Memoria RAM externa, usada para almacenar datos y código de programa. El esquema de carga de programa mediante el uso del software de la CV utiliza espacio de esta área memoria para almacenar el nuevo programa.
- 960 KB de Memoria RAM expandida, accedida mediante el traslape del segmento 03 de la memoria externa, este acceso se controla mediante software en la computadora de vuelo, por esto solamente se utiliza para almacenar datos de telemetría o imagen, no de código, de lo contrario ocasionaría resultados inesperados.

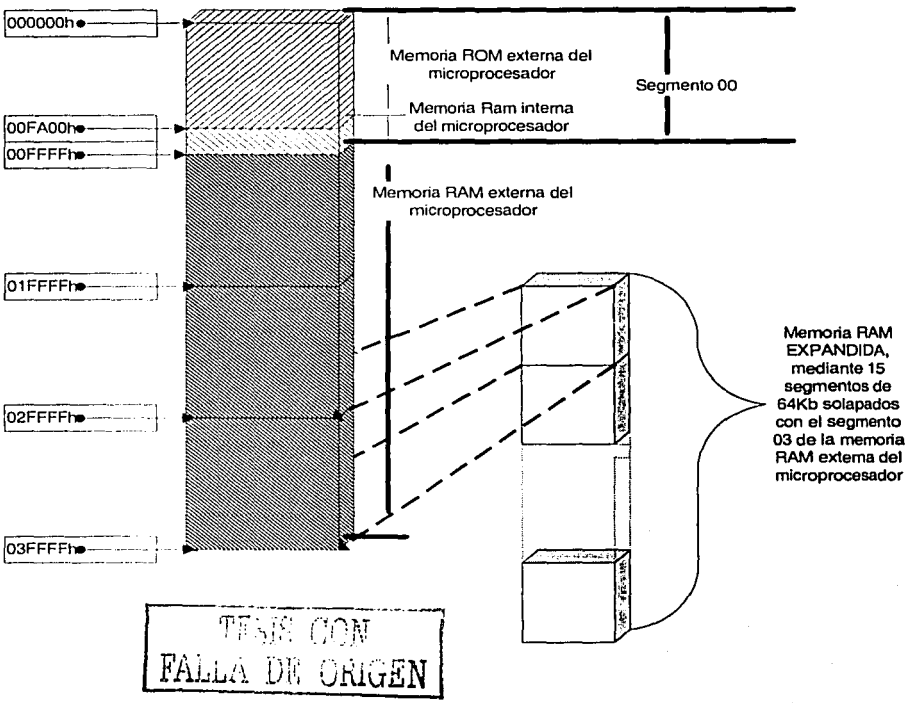


Figura 3.3: Mapa de memoria 1 (ROM + RAM)

El mapa de memoria 2 que cuenta solamente con memoria tipo RAM se utiliza para cargar nuevos programas mediante el modo BTL, figura 3.4. Cuando se utiliza este mapa de memoria la computadora de vuelo cuenta con los siguientes recursos de memoria:

- 255 Kbytes de memoria RAM externa de MCU que puede ser usada para el almacenamiento de código o datos de programa.
- 1 KB de Memoria RAM interna del microprocesador, usada por el microprocesador para el almacenamiento de registros de trabajo y configuración de sistema, y para la implementación de la pila o "Stack". El espacio restante puede ser administrado libremente por el usuario [SIEMENS, 1997].

- 960 KB de Memoria RAM expandida, accedida mediante el traslape del segmento 03 de la memoria externa. El acceso se controla mediante software en la computadora de vuelo, por esto solamente se utiliza para almacenar datos de telemetría o imagen, no de código, de lo contrario ocasionaría resultados inesperados.

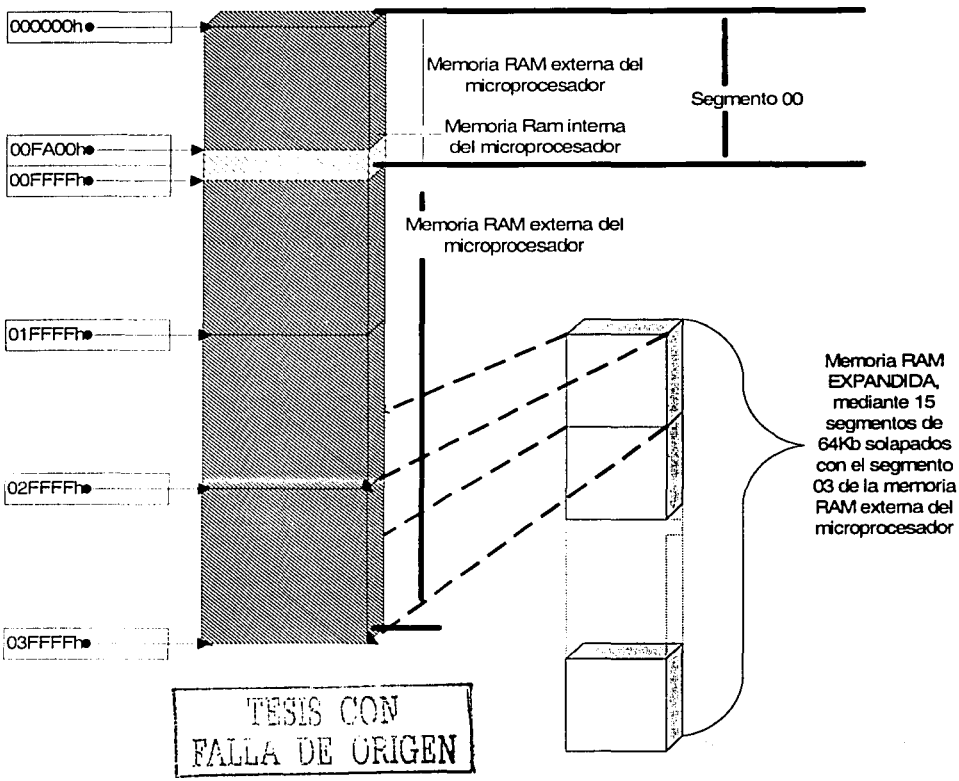


Figura 3.4: Mapa de memoria 2 (solo RAM)

Como se observa, los mapas de memoria 1 y 2 difieren solamente en que el segmento 00 de la memoria externa está compuesto por ROM en el primero y por RAM en el segundo, este último mapa de memoria es el que debe ser usado con el esquema de carga y ejecución de programas mediante el BTL, lo que permite usar más de 63Kbytes de memoria para el programa de control a diferencia del primer mapa de memoria en el cual el software de control está limitado a dicha cantidad de memoria.

3.4.2 Descripción del Hardware de activación del modo BTL en la computadora de vuelo

En el caso de la CV, el modo BTL será controlado por una computadora de carga útil, el sistema elegido para este propósito fue el decodificador de tonos o procesador de sobre vivencia (DT o PS). DT cuenta con líneas de control para la configuración de los procesadores de la CV, es decir, mediante ellas DT informa al MCCP [Apartado 2.2] cual o cuales de los procesadores y en que forma deben ser encendidos, así mismo estas líneas de control proporcionan medios a DT para activar el Hardware de BTL. Las líneas de control de DT a CV son:

- **SELCR0-1**: línea de selección que indica cual procesador redundante se usará CR0 ó CR1, mediante los niveles de voltaje 0 y 5v respectivamente.
- **ON_CR_DTL**: línea de selección que indica si se utiliza o no uno de los procesadores redundantes CR0 o CR1 (0-ON, 5v-OFF).
- **ONL/OFF CP**: indica si será encendido el procesador principal CP o no, 0v y 5v respectivamente.
- **DWNLSW TTL#**: En 0Volts activa el mecanismo de arranque en modo BTL, del procesador seleccionado.
- **ROM/RAM#**: selecciona uno de los mapas de memoria para el procesador conectado al bus de instrumentación.
- **RESET CV#**: aplica un reset por Hardware al procesador conectado al bus de instrumentación.

Además, la computadora de vuelo utiliza la línea **RCPROG**, para informar a DT el progreso de la operación de carga, debido a que esta línea se comparte para el control de las BTM's solamente debe ser observada por el DT durante el proceso de carga.

A manera de síntesis las figuras 3.5 a 3.7 y las tablas 3.1 y 3.2 muestran diversos diagramas lógicos y tablas de verdad de Hardware de encendido de procesadores, selección de bus, Hardware de arranque en modo BTL, y del decodificador de memoria.

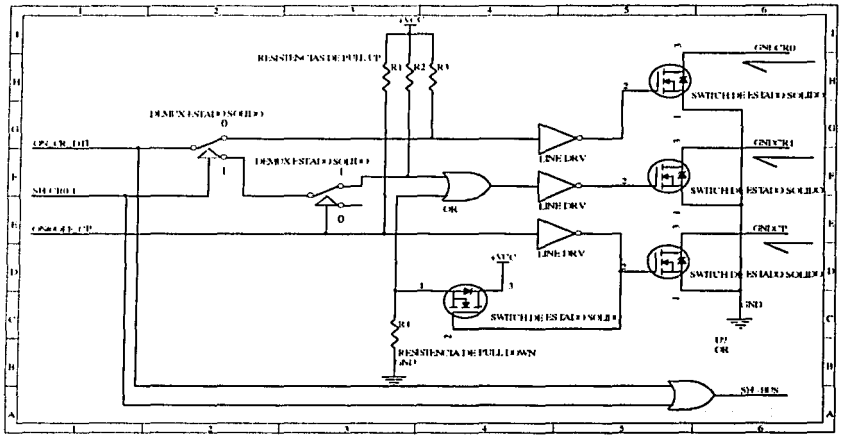


Figura 3.5: Diagrama lógico de electrónica de encendido de procesadores y control de bus.

Tabla 3.1: Tabla de verdad del Hardware de control de encendido de procesadores y asignación de bus de instrumentación.

Modo de operación	Líneas de control			Procesadores encendidos			Bus de instrumentación conectado a
	ON#/OFF_CP	ONCRDTL	SELCR0-1	CP	CR0	CR1	
P. P.	0	0	0	ON	ON	OFF	CR0
C.P.	0	0	1	ON	OFF	OFF	CP y CR1
C.P.	0	1	0	ON	OFF	OFF	CP y CR1
C.P.	0	1	1	ON	OFF	OFF	CP y CR1
CR0	1	0	0	OFF	ON	OFF	CR0
CR1	1	0	1	OFF	OFF	ON	CP y CR1
EF. LUP.	1	1	0	OFF	OFF	OFF	CP y CR1
EF. LUP.	1	1	1	OFF	OFF	OFF	CP y CR1

Descripción del los modos de operación	
P.P	Modo de procesamiento paralelo, CR0 actúa como procesador maestro, él posee el bus de instrumentación
C.P.	Modo de operación del procesador principal, con este modo inicia operaciones en vuelo la CV
CR0	Modo de operación del procesador redundante 0 (solo CR0)
CR1	Modo de operación del procesador redundante 1 (solo CR1)
EF. LUP.	Modo de operación con efecto "Latcth-up", el modulo de detección y eliminación de efecto "Latchup", tiene prioridad sobre el control externo de DT

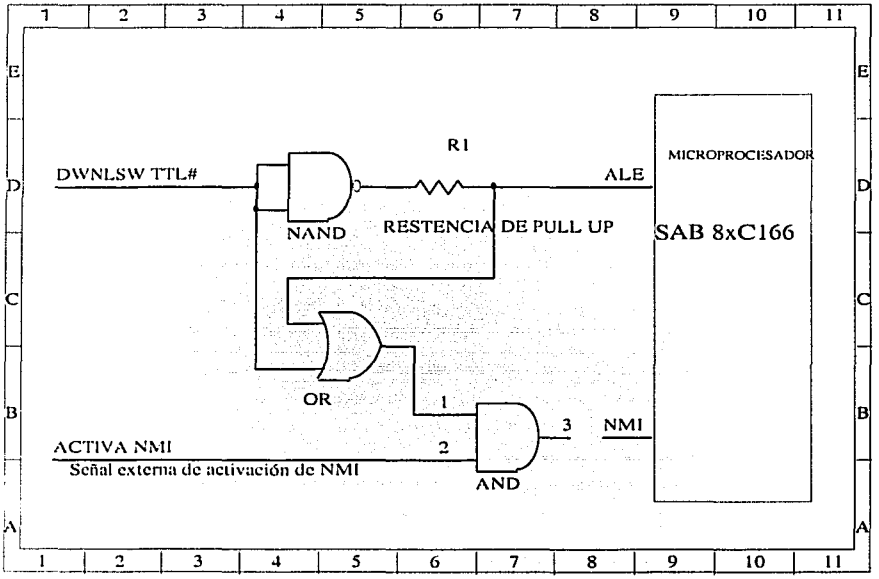


Figura 3.6: Diagrama esquemático del circuito de arranque en modo BTL

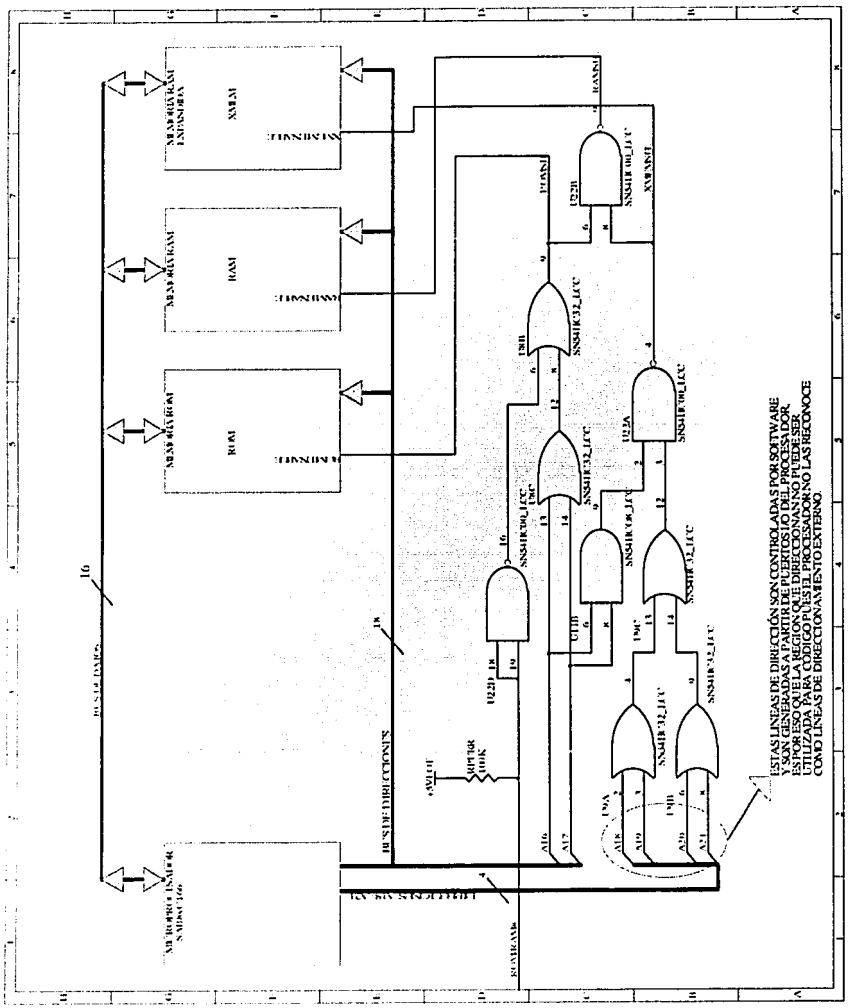


Figura 3.7: Diagrama esquemático del decodificador de memoria

Tabla 3.2: Tabla de verdad del decodificador de memoria.

Líneas de selección de segmento de memoria externa (controladas por el hardware del Microcontrolador)		Líneas de selección de segmento de memoria expandida (controladas por software)				Línea de selección de mapa de memoria	Líneas de habilitación de bloques de memoria (Líneas tipo activo bajo 0→Acceso 1→ No acceso)			Acceso al segmento de memoria	Modo de acceso
A17	A16	A21	A20	A19	A18	ROM/RAM#	ROMEN	RAMEN	XMEM		
0	0	X	X	X	X	0	1	0	1	00	1
0	0	X	X	X	X	1	0	1	1	00	2
0	1	X	X	X	X	X	1	0	1	01	3
1	0	X	X	X	X	X	1	0	1	02	
1	1	0	0	0	0	X	1	0	1	03	4
1	1	0	0	0	1	X	1	1	0	03	5
1	1	0	0	1	0	X	1	1	0	03	
1	1	0	0	1	1	X	1	1	0	03	
1	1	0	1	0	0	X	1	1	0	03	
1	1	0	1	0	1	X	1	1	0	03	
1	1	0	1	1	0	X	1	1	0	03	
1	1	0	1	1	1	X	1	1	0	03	
1	1	1	0	0	0	X	1	1	0	03	
1	1	1	0	0	1	X	1	1	0	03	
1	1	1	0	1	0	X	1	1	0	03	
1	1	1	0	1	1	X	1	1	0	03	
1	1	1	1	0	0	X	1	1	0	03	
1	1	1	1	0	1	X	1	1	0	03	
1	1	1	1	1	0	X	1	1	0	03	
1	1	1	1	1	1	X	1	1	0	03	

MODOS DE ACCESO A MEMORIA	
1	Acceso a memoria ROM: el acceso a ROM solo puede efectuarse en el segmento de memoria externa número 00, mediante la configuración del mapa de memoria 1 (ROM+RAM).
2	Acceso al segmento 00 de memoria RAM, este segmento es accesado en el lugar de la memoria ROM, cuando el mapa de memoria seleccionado es el mapa 2.
3	Acceso a los segmentos 01 y 02 de memoria RAM, este acceso es independiente del mapa de memoria seleccionado, no requiere del establecimiento previo de los valores de las líneas A18 a A21, ya que es efectuado independientemente por el hardware de control de bus externo del microprocesador.
4	Acceso al segmento 03 de memoria RAM, antes de llevar a cabo este acceso es necesario establecer los valores de las líneas A18 a A21 como cero con la finalidad de evitar el acceso a un segmento de memoria expandida.
5	Acceso a la memoria expandida, se lleva a cabo mediante el solapamiento de un segmento de 64 Kbytes con el segmento 03 de la memoria RAM externa (no expandida) del microprocesador, para lo cual es necesario establecer el número del segmento expandido en el que se llevará a acabo la operación, a través de las líneas A18 a A21 y efectuar la lectura o escritura de datos en el segmento 03 del microprocesador, de esta forma para el microcontrolador, un acceso a un segmento expandido será siempre un acceso al segmento 03 de su memoria externa.

Como fue explicado anteriormente, para cargar un programa es necesario que se efectúe un control externo sobre el Hardware de arranque y sobre la selección del mapa de memoria, este control lo realiza el DT. A continuación se describe el proceso de carga de nuevos programas en términos del Hardware del DT:

0) Mediante la estación terrena se envía el comando Nuevo programa a la CV, ésta lo retransmite DT y espera el reconocimiento del mismo en tierra. Cuando CV envía el comando a DT, le indica cual es el procesador al que se cargará el nuevo programa (CP, CR0 o CR1). De esta forma DT sabe cual procesador es el que estaba activado y en caso de que falle el intento de subir programa, deberá ordenar (al finalizar el intento) que quede como activo tal procesador en la CV (pasos 4 o 12).
1) DT apaga a CV mediante el las líneas: **ON_CR_DTL=1, ONL/OFF CP=1, SELCR0-1=1.**
2) DT activa el mapa de memoria 2, mediante la línea **ROM/RAM#=0.**
3) DT enciende a CV de acuerdo con las opciones recibidas en el comando de subir programa (paso 0). Las líneas de control de encendido se ajustan de acuerdo a la tabla de verdad de la figura 3.7, con los posibles modos que son los siguientes:
   - Procesador principal de CV
   - Procesador redundante 0.
   - Procesador redundante 1.
4) DT activa el Hardware de arranque en modo BTL mediante la línea **DWNLSW_TTL#=0.**
5) DT envía un pulso[1] por la línea **RESET CV#.**
6) DT espera 3 segundos (fase de inicialización del procesador).
7) DT Desactiva el hardware de arranque en modo BTL mediante el establecimiento de la línea **DWNLSW TTL#** a un nivel alto.
8) En el software de estación terrena se ejecuta la función de "Carga de Programa".
9) DT espera por un tiempo **T0max** = 30seg la respuesta de "continuar" que dará CV mediante una transición de un 1 lógico a un 0 lógico en la línea **RCPROG**, en caso de que no se obtenga una respuesta de la CV, DT debe proceder al paso 4 con un máximo de 10 reintentos, si no se recibe la respuesta de "continuar" después de 10 reintentos se procede al paso 12.
10) Si DT obtuvo la respuesta de continuar por parte de CV, DT debe esperar la respuesta de "éxito" de carga de programa durante un tiempo **T1max** = 10Min, por parte de CV mediante una transición de 0 a 1 en la línea **RCPROG**, si no se recibe la respuesta de "éxito" se procede al paso 12.
11) DT envía nuevamente un pulso por la línea **RESET CV#** y termina la carga de programa con éxito.
12) DT debe activar el mapa de memoria 1 y encender la CV con el procesador indicado en el paso 0 mediante el uso de las líneas pertinentes. (figuras 3.6 y 3.9).
13) Finalmente DT, envía un pulso por la línea **RESET CV#**, en este caso se suspende el intento de transferencia de nuevo programa al satélite.

A continuación se presenta un diagrama de flujo del proceso de subir programa a CV.

[1] Este pulso es de +5v a 0V y permanece en 0V por lo menos durante 500ms.

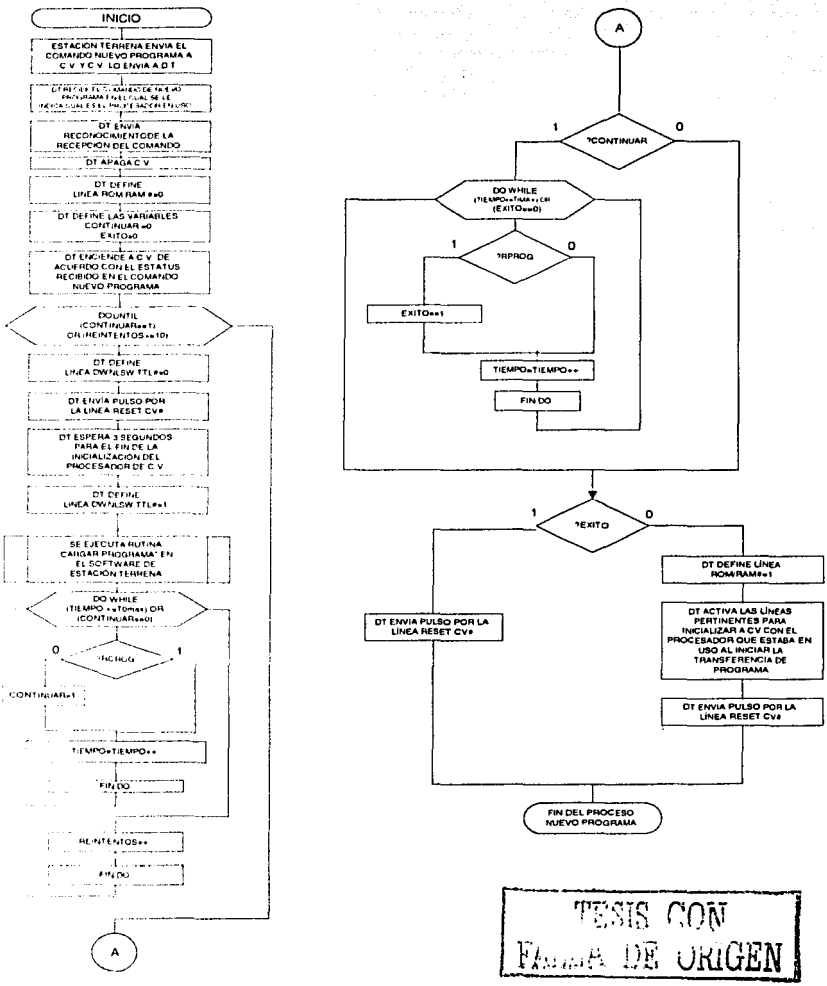


Figura 3.8: Diagrama de flujo del proceso de carga de programa desde un punto de vista de Hardware

3.4.3 Descripción del software para el proceso de carga de programas

Una parte importante del proceso de carga de programas a la CV es el software que controla los procesos de comunicaciones entre la estación terrena y la computadora de vuelo, dicho programa es el responsable de enviar el comando correspondiente a la CV para que esta lo envíe a su vez a DT, de establecer la comunicación inicial entre la estación terrena y el software de BTL a bordo de la CV (Handshake), de enviar los diferentes segmentos de código (32 bytes , precargador y monitor) y de efectuar el envío del código del programa de usuario al satélite. Comúnmente los procesadores que ofrecen una característica similar al BTL del SAB8xC166, al ser adquiridos en sus versiones de desarrollo incluyen un software de comunicaciones y protocolos para la carga de nuevos programas, sin embargo, para SATEX fue necesario desarrollar nuevamente este conjunto en su totalidad, de tal forma que permitiera:

- Mayor control sobre la transmisión.
- Integración plena con el software de estación terrena.
- Implantación de protocolos capaces de tolerar fallas en el proceso de carga de programa.
- Flexibilidad en la adaptación del software de cargado de nuevos programas con el Hardware de la CV

El conjunto de software desarrollado para este propósito consta de las siguientes partes:

- Módulo de software para la ET (se ejecuta en una PC).
- Módulo precargador ó primeros 32 bytes (se ejecuta en la CV).
- Módulo de código de inicialización y carga de monitor (se ejecuta en la CV).
- Módulo Monitor de memoria, para la carga del programa de usuario a CV (se ejecuta en la CV).

En las figuras 3.9 y 3.10 se observan a grandes rasgos el diagrama de flujo del software y su interacción dentro del proceso de carga de nuevos programas al satélite.

Del diagrama se observa que existen 3 procesos diferentes, ejecutados de forma paralela en cada una de las computadoras que intervienen en el proceso de carga de nuevos programas a CV, a en las páginas subsecuentes se describen de manera general cada uno de ellos.

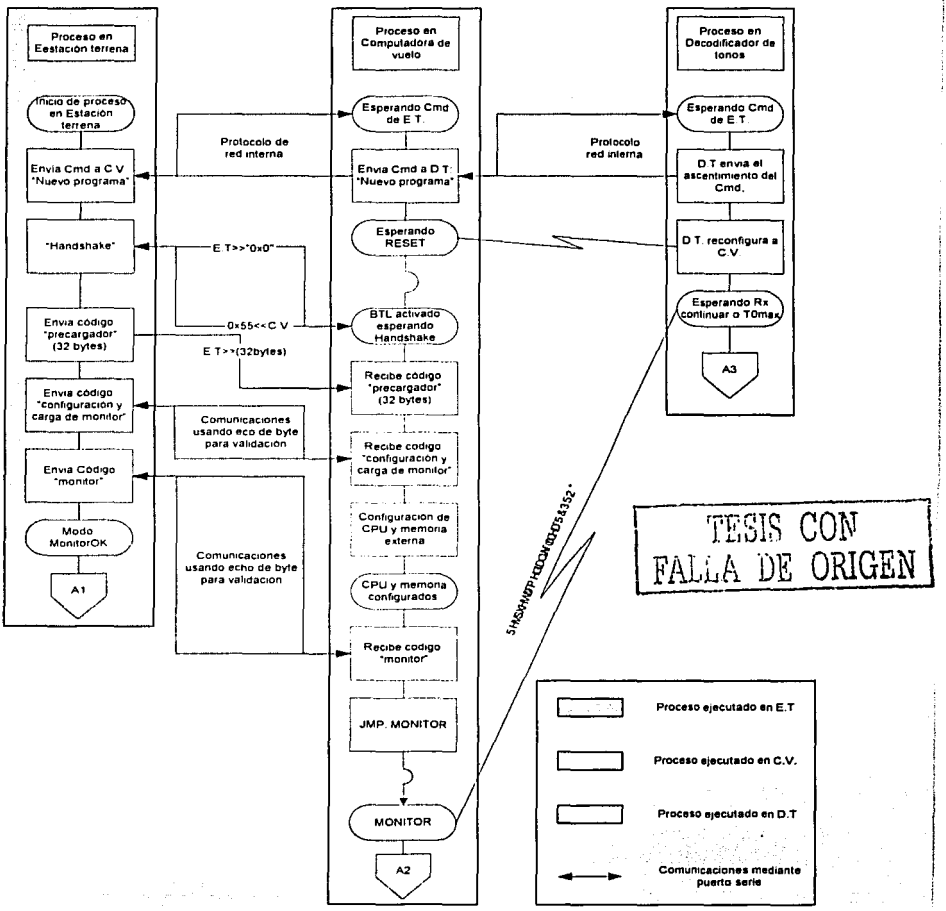


Figura 3.9: Diagrama de flujo del software para la carga de programas (1)

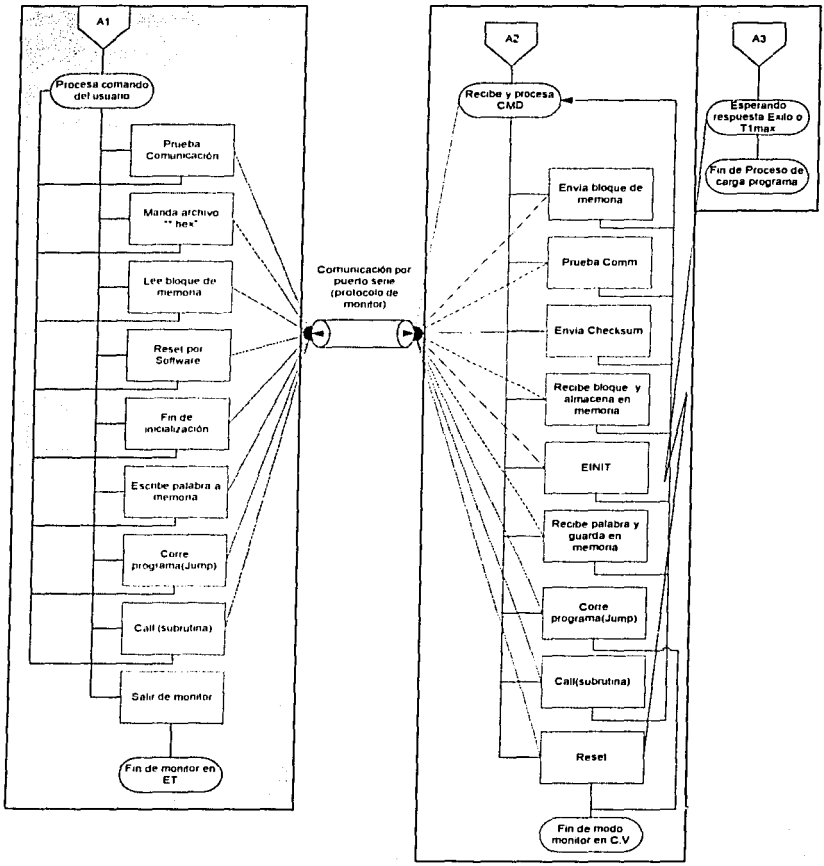


Figura 3.10: Diagrama de flujo del software para la carga de programas (Continuación.)

3.4.3.1 Proceso en estación terrena

El software fue desarrollado con el lenguaje Microsoft Visual Basic V.5, por lo cual está basado en el proceso de eventos ("mouse", puerto serie, etc.) que a su vez resuelve el sistema operativo, en la figura 3.13 se muestra la ventana principal del software de carga de nuevos programas ejecutado en Estación Terrena.

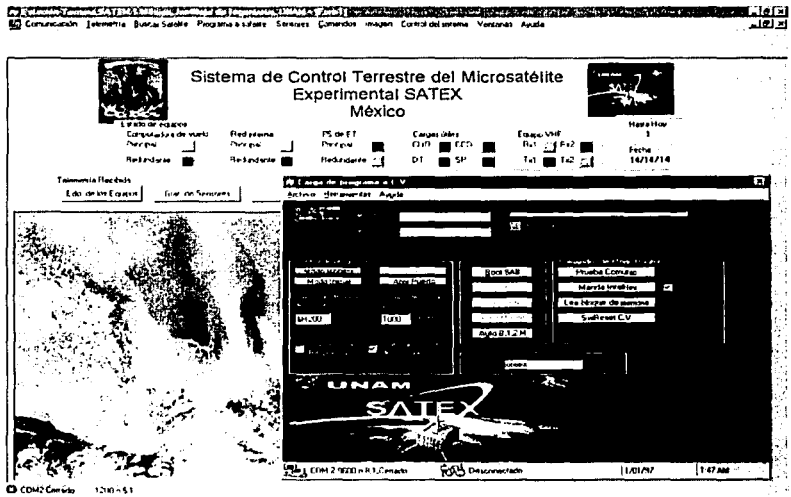


Figura 3.11: Software de Carga de nuevo programa en Estación Terrena.

Se pueden identificar dos modos básicos de operación, el modo inicial y el modo monitor. En el modo inicial, el programa no conoce información alguna en cuanto al estado del proceso de carga en la CV y se asume que la CV está lista para comenzar el proceso, dentro de este modo ocurre el envío del comando "Nuevo programa" mediante el protocolo de red interna [TORRES, 2002], luego dentro del mismo modo, se efectúa el envío de los bloques de código "Handshake", "precargador", "configuración" y "monitor". El envío de este código se efectúa con un protocolo de comunicaciones simple mediante el envío de un eco por cada caracter recibido por parte de la CV, además para propósitos de sincronización la CV envía Bytes preestablecidos para indicar la parte del proceso en la que se encuentra.

En caso de encontrar una falla en las comunicaciones durante cualquier momento de la ejecución del proceso dentro del modo inicial, se reestablece el sistema y se espera a que CV sea reestablecida por D.T (figura 3.9)

El modo monitor en ET puede establecerse de 2 maneras:

- De manera automática al terminar el envío del código monitor.
- Mediante la ejecución del comando de ET "Indaga Monitor", el cual puede usarse para intentar una sincronización entre los procesos de CV y ET, en el caso de haber ocurrido un error de pérdida de sincronía durante la ejecución del monitor en CV

El modo monitor en ET se comunica directamente con el software monitor en CV El monitor de CV funciona como un servidor de comandos, el modo monitor en ET hace las veces de cliente, que solicita la ejecución de comandos de manera asíncrona, mediante el siguiente protocolo:

1. ET envía 1 byte de comando que puede ser alguno de los listados en la tabla 3.3.
2. CV recibe el byte que le indica cual es el comando y envía la notificación de la recepción del comando mediante el byte "0xAAh".
3. Si CV identifica el comando éste se ejecuta y al finalizar se envía la notificación de una correcta ejecución mediante el byte "0xEAh". Si no se identifica el byte del comando no se notifica su correcta ejecución.
4. Durante la ejecución de los comandos se efectúa la transferencia de los parámetros usados por el comando, primero los de entrada (ET a CV) y luego los de salida como se muestra el la tabla 3.3.

Tabla 3.3: Comandos para el monitor de CV

Comando	Código	Parámetros	
		Entrada	Retorno
Escribe Bloque a Memoria	0x084h	3Bytes dirección ini. 2Bytes -- longitud (LSB primero) <longitud>Bytes de datos	Ninguno
Lee Bloque de Memoria	0x085h	3bytes dirección ini. 2Bytes -- longitud (LSB primero)	<longitud>Bytes de datos
Escribe Word a Memoria	0x082h	3Bytes dirección 2Bytes de dato	Ninguno
Llamar subrutina	0x09Fh	3Bytes Dirección de subrutina 8Words de parámetros de subrutina (R8-R15) (LSB primero)	8Words de parámetros de subrutina (R8-R15) (LSB primero)
Ejecutar programa	0x041h	3Bytes dirección de programa (LSB primero)	Ninguno
Ejecutar EINIT	0x031h	Ninguno	Ninguno
Ejecutar SWRESET	0x032h	Ninguno	Ninguno
Petición de Checksum	0x033h	Ninguno	1Word de cheksum
Prueba de comunicaciones	0x093h	Ninguno	Ninguno

Para el envío del código del programa a satélite, el proceso efectúa de forma iterativa la ejecución de los comandos "Escribe Bloque a Memoria" y "Petición de Checksum", en el caso de fallar alguno de los comandos ET intenta una resincronización con la CV. Si se obtiene una sincronización exitosa, continua el proceso con la ejecución del comando interrumpido, si vuelve a fallar el comando nuevamente, se reintenta tantas veces como el usuario halla seleccionado para reintento del mismo.

3.4.3.2 Proceso ejecutado en CV

El proceso ejecutado en la Computadora de Vuelo, está dividido en la ejecución de 5 módulos de software cargados en diferentes momentos; tales módulos se explican a continuación:

- Módulo de recepción del comando de inicio de transferencia de nuevo programa bajo modo BTL. Este módulo se encarga de la recepción del comando de ET y de su retransmisión mediante red interna hacia el decodificador de tonos, el fin de la ejecución de dicho módulo esta marcada por el reset a CV que es ordenado por DT lo cual genera la puesta en marcha del Módulo Bootstrapping del SAB80C166.

- Módulo de Bootstrapping del SAB80C166 grabado en la memoria ROM de arranque interna del SAB80C166. Es responsable de efectuar el HANDSHAKE con el software de ET y de recibir y almacenar 32 Bytes de código en memoria RAM, el fin de este módulo ocurre mediante un salto a la dirección inicial del código recién cargado que en este caso corresponde al código de precarga.

- Módulo de Precarga: efectúa la recepción del siguiente módulo; el código de "Configuración y Cargado de Monitor", ésta recepción se efectúa con un protocolo simple de comunicación mediante el envío de un eco de respuesta por cada caracter recibido. Ya que el código recibido por el módulo de precarga se coloca inmediatamente después de él, la ejecución del siguiente módulo de código se inicia sin efectuar explícitamente una instrucción de salto de ejecución, de manera transparente para la ET, por lo que el fin de la ejecución de este módulo está señalizado por el envío a Tierra de un caracter de control predefinido.

- Módulo de Configuración y de Cargado de Monitor: configura los dispositivos necesarios para la carga de memoria (Memoria Externa, Puntos de I/O, etc.); y a continuación carga el código monitor hacia la memoria externa recién configurada [Figura 3.3]. La recepción del código monitor, se realiza mediante el protocolo simple antes mencionado, el fin de la ejecución del presente módulo, esta marcado por un salto a la localidad 0, en la que fue almacenada una instrucción de salto a la dirección de inicio del monitor, la cual puede ser definida por el usuario de ET

- Módulo Monitor de CV[4]: El cual forma parte del servidor de comandos descrito anteriormente encargándose de la señalización de su inicio hacia DT mediante la línea "RCPROG" [Apartado 3.4.2] y de la señalización de su correcta ejecución mediante la misma línea. El final de la ejecución de este módulo esta señalizado por la ejecución del comando "SFWRESET"del monitor en ET esto causa que el código monitor de CV envíe una respuesta del comando ejecutado a ET y que inmediatamente después se envíe la respuesta de éxito a DT mediante la línea "RCPROG" causando que DT efectúe un reset por hardware a CV iniciando así la ejecución del nuevo software. (aquivoy)

Finalmente es importante mencionar que la técnica descrita se utiliza actualmente para cargar cotidianamente el software de la CV, lo cual ha servido como proceso de validación continua.

[4] El código de este monitor puede consultarse en el Apéndice B, el código está basado en el código de un monitor mínimo proporcionado por SIEMENS [www.siemens.com] en su aplicación suplementaria AP166402 para el 8XC166X.

Capítulo 4. Implantación de un dispositivo "EDAC" y ampliación de la memoria RAM

4.1 Introducción

Como se explicó en el capítulo 2, los sistemas espaciales están expuestos a condiciones de radiación que afectan la operación de sus componentes electrónicos, una forma en la que afecta ésta radiación a las memorias de tipo RAM son los "Single Event Upsets" [Apartado 2.3.2]. El caso de SATEX no es la excepción, por lo que se propuso la utilización de un dispositivo para la detección y corrección de errores en RAM, EDAC por sus siglas en ingles "Error Detection And Correction unit", que se basa en la utilización de una palabra de síndrome para la detección de errores.

Por otro lado uno de los experimentos del proyecto SATEX será la adquisición de imágenes, que serán tomadas por una cámara digital CCD de grado comercial adaptado para su uso en el espacio. Debido a que los equipos electrónicos comerciales son más susceptibles a los efectos de la radiación, aún cuando hallan sido adaptados, se decidió utilizar un esquema de operación para el equipo CCD en el que solamente estará encendido en el momento de tomar la o las fotografías para enviarlas inmediatamente a la CV para su almacenamiento, por lo que los requerimientos de memoria de la CV se ven afectados. Como solución se decidió efectuar una ampliación a la memoria RAM de la CV, que en un principio era de un máximo de 256kbytes, para que ahora sea capaz de alojar hasta 1.187 Mbytes de memoria.

Las modificaciones señaladas están muy ligadas entre sí porque ambas requieren del rediseño del esquema de decodificación de memoria y de la lógica de control de la misma, además de que el EDAC también deberá proteger a la memoria ampliada aunque no de la misma manera que a la memoria normal. A este respecto, el presente capítulo se enfoca a describir el trabajo desarrollado.

4.2 Códigos de protección

No sólo en los datos de los equipos de procesamiento se producen errores, un lugar común en el cual se producen estos es durante la transmisión de información de los sistemas de comunicaciones y precisamente en ellos fue donde se originaron los códigos de protección. La protección consiste en agregar cierta cantidad de bits al mensaje que se desea transmitir. Hay algoritmos para determinar cuales y cuantos bits se agregan y lo que se debe hacer en el receptor con tales bits.

Intuitivamente es más fácil decir si hay bits erróneos que decir cuales son; o sea que, es más sencillo y más barato un código para detectar que uno para corregir.

Entre los códigos de protección encontramos los siguientes:

- Procedimiento de detección por paridad par
- Procedimiento de corrección por verificación cruzada
- Procedimiento de síndrome para corrección de errores
- Procedimiento de detección con chequeo de redundancia cíclica

Algunas de estas técnicas han sido adaptadas para la detección de errores espurios en las memorias RAM, como es el caso de la detección por chequeo de paridad y de la corrección por medio da la generación de síndrome. Estas técnicas se usan para proteger las memorias dinámicas de alta densidad en aplicaciones comerciales y cualquier tipo de memorias RAM en aplicaciones espaciales, ya que éstas presentan una tasa de errores significativa que puede ser de suma importancia en el desempeño de los sistemas, [IDT, 1996]. A continuación se muestran diferentes tablas que nos muestran la probabilidad de ocurrencia de SEU's para diversas configuraciones de sistemas espaciales [TEMIC, 1997-1].

Tabla 4.1: Predicciones de SEU's, para diferentes orbitas, por BIT y por día

Tipo de orbita	Predicción SEU's por bit y por día debidos a iones pesados.	Predicción de SEU's por bit y por día debidos a Protones	Predicción total de SEU's por bit y por día
Orbita terrestre baja	6.7E-11	1.6E-7	1.6E-7
Orbita Polar	1.6E-7	4.9E-7	6.5E-7
Orbita Geoestacionaria o misiones de espacio profundo	6.2E-7		6.2E-7

Tabla 4.2: Tipos de computadoras

Computadora	Tipo de microprocesador	Chip	Tamaño de memoria en Kbytes
A	Microcontrolador de 8 bit	TEMIC 80C32E	64
B[5]	Microcontrolador de 16 bit	MA31750	256

Tabla 4.3: Tabla de SEU's para chips energizados e inactivos

Computadora	Tamaño de memoria en Kilobits	SEU en orbita baja	SEU en orbita polar	SEU en orbita geoestacionaria
A	512	1 en 12.2 días	1 en 3 días	1 en 3 días
B	2048	1 en 3 días	1 en 18 horas	1 en 19 horas

Para efectuar dicha detección y corrección de errores, diversos fabricantes de componentes electrónicos han desarrollado dispositivos electrónicos EDAC's, de tipo comercial y de tipo espacial. En el siguiente apartado se describe de forma general el funcionamiento de los diversos dispositivos.

[5] Esta es la computadora similar a la utilizada en satex.

4.3 Dispositivos electrónicos EDAC

La función básica de un dispositivo EDAC es la de analizar la integridad de los datos leídos del sistema de memoria, señalizar un error si se ha detectado y si es posible corregir ese error. La mayoría de los dispositivos EDAC implementan esta función usando los mismos principios generales, con algunas variaciones entre cada dispositivo.

La operación de un dispositivo EDAC puede dividirse en:

i) Generación de una palabra codificada basándose en la palabra de datos que está se escribe a memoria. Esta palabra codificada se llama "**Check-bit Word**" o palabra de chequeo. Esta operación es llamada **Generación.**
ii) Detección de errores en la palabra de datos leída de memoria, mediante la comparación de la correspondiente palabra de chequeo con una nueva palabra de chequeo (generada a partir de la lectura de la palabra de datos). Si es posible se efectúa la corrección del error. La comparación de las dos palabras de chequeo (una función lógica OR exclusiva XOR) produce una palabra de síndrome. Esta operación es llamada **Detección/Corrección.**

El esquema de codificación empleado por la mayoría de los EDAC es un código **Hamming** (desarrollado en los laboratorios AT&T Bell) con alguna variación. Por cada palabra de datos escrita a memoria, una palabra de chequeo, a la palabra de datos. La nueva palabra (palabra de datos + palabra de chequeo) se establece como un código valido. La distancia Hamming entre cada uno de los códigos validos varía de dispositivo a dispositivo.

La distancia Hamming entre dos palabras es el número de bits en que difieren una de la otra, por ejemplo:

10001110	11100101
00111000	11110111
D=5	D=2

Dos palabras serán más fáciles de distinguir cuanto mayor sea su distancia Hamming, ya que si la distancia es *d* será necesario que se produzcan *d* errores para que una palabra pase a ser la otra. De este análisis se desprende que la eficacia de un código será función de su distancia Hamming, que se define como la mínima distancia que puede encontrarse entre dos palabras que pertenezcan a ese código [MORENO, 2002]. En general:

- Un código de distancia mínima de Hamming **d** será capaz de detectar **d-1** errores.
- Un código de distancia mínima de Hamming **d** deberá ser capaz de corregir **(d-1)/2** errores.
- Un código que corrija t errores y detecte d (d>t) errores debe tener una distancia mínima igual a **dm=t+d+1.**

Las principales reglas relativas al control de paridad en los códigos Hamming son:

- Dos bits de paridad no pueden controlar la paridad de un mismo conjunto de bits de información.
- No se puede incluir en el conjunto de bits controlado por un bit de paridad, otros bits de paridad.
- Un error en un bit de información debe afectar a dos o más bits de paridad.

4.1.1 Arquitectura de los dispositivos EDAC

Existen dos arquitecturas básicas para la operación de los EDAC: Flujo a través o "Flow-Thru" y Flujo paralelo o "Bus-Watch".

4.1.1.1 Arquitectura de flujo paralelo o Bus-Watch

Un dispositivo con arquitectura Bus-Watch tiene un solo bus de datos. Una configuración básica de un sistema usa este tipo de arquitectura se muestra en la figura 4.1

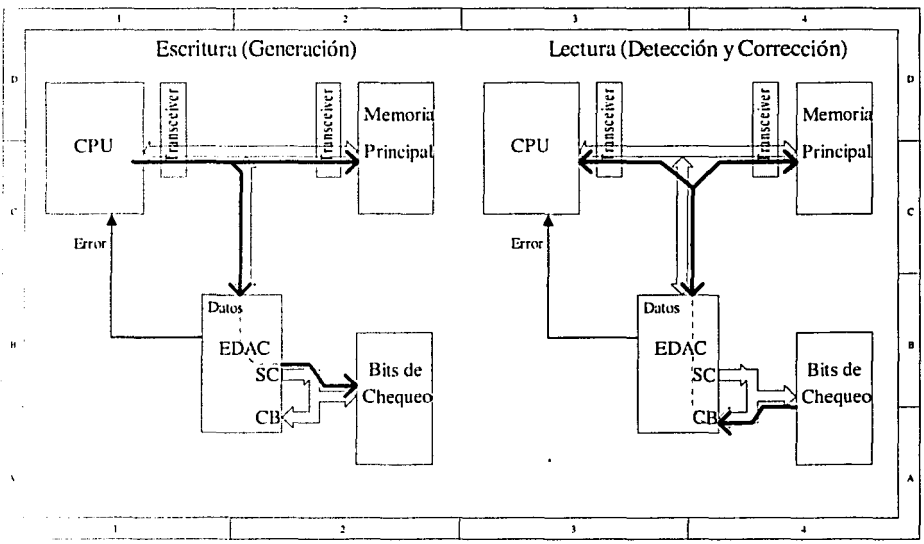


Figura 4.1: Configuraciones básicas utilizando un EDAC "BusWatch", con un sistema de E/S de memoria común.

Durante una operación de escritura. El CPU manda datos a la memoria principal. Al mismo tiempo, los datos van hacia la unidad EDAC, la cual genera la palabra de chequeo y la almacena en la memoria de verificación.

Por el otro lado, durante una operación de lectura, los datos de la memoria principal y la palabra de chequeo de la memoria de verificación pasan primero al dispositivo EDAC. Basándose en la información contenida por la palabra de chequeo, la unidad EDAC puede detectar y en su caso corregir los errores presentes en la palabra compuesta (palabra de datos + palabra de chequeo), en este esquema los datos de la memoria principal y los datos enviados al CPU pueden ser corregidos al mismo tiempo. El número de errores que un dispositivo puede detectar y/o corregir depende de las longitudes de las palabras de datos y de chequeo.

4.1.1.2 Arquitectura de flujo a través o Flow-Thru

En contraste a un dispositivo EDAC Bus-watch, un EDAC Flow-Thru provee por lo menos 2 buses de datos: un bus de datos de sistema SD(System Data) y un bus de datos de memoria MD(Memory Data). La arquitectura de bus doble mejora la velocidad de procesamiento de datos y simplifica la interfaz del bus de sistema del CPU y el bus de las memorias. En la figura 4.2 se muestra una configuración básica de un EDAC Flow-Thru.

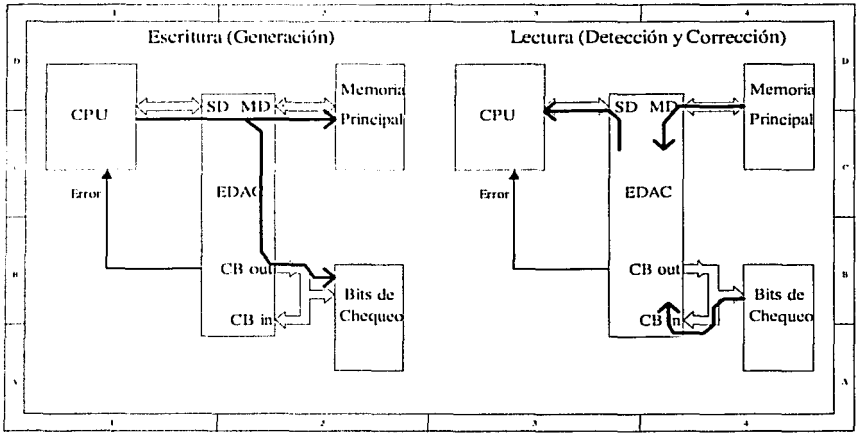


Figura 4.2: Configuraciones básicas utilizando un EDAC "FlowThru", con un sistema de E/S de memoria común.

En la configuración de Entrada/Salida común, durante una operación de escritura, los datos del CPU fluyen a través de la unidad EDAC y son escritos en

la memoria principal. Cuando los datos fluyen a través del EDAC, la palabra de chequeo se genera y escribe en la memoria de chequeo. Durante una operación de lectura, los datos de la memoria principal entran en el EDAC a través del bus MD mientras que la palabra de chequeo entra al EDAC a través del bus CB. Entonces la unidad EDAC detecta los errores y carga la palabra corregida en el CPU a través del bus SD.

En la configuración de Entrada/Salida separada, durante una operación de escritura, los datos del CPU se envían directamente a la memoria principal y al mismo tiempo al EDAC a través del bus SD. La unidad EDAC genera entonces la palabra de chequeo y la escribe a la memoria. Durante la lectura, los datos de la memoria principal entran en la unidad mediante el bus MD mientras que la palabra de chequeo entra por el bus CB. Entonces el EDAC detecta los errores y si es posible carga la palabra corregida en el CPU mediante el bus SD.

4.4 *Elección del dispositivo EDAC*

En el mercado existe una gran cantidad de EDACs disponibles y por tanto se requirió tiempo para buscar datos de ellos, estudiarlos y elegir alguno que requiriera la menor cantidad de cambios en la computadora de vuelo. Y aún así se debería afrontar el problema de conseguir el componente electrónico debido a que es una parte que puede ser utilizada en equipos bélicos, lo cual implica retrasos y restricciones de adquisición. [VICENTE, 2001].

En la Tabla 4.4 se muestran diferentes dispositivos EDAC encontrados en el mercado, entre los cuales se hizo el estudio y la elección del usado a bordo de la computadora de vuelo:

Tabla 4.4: Tabla de EDACs encontrados en el mercado.

Fabricante	No. Parte	Arquitectura	Longitud de palabra	Tipo	Observaciones
Integrated Device Technology (IDT)	IDT39C60	BUS WATCH	16-BIT	COMERCIAL	Obsoleto, se puede usar en cascada hasta 64 BIT
	IDT49C460		32-BIT		se puede usar en cascada hasta 64BIT
	IDT49C465	FLOW THRU			En producción
	IDT49C466		64-BIT		
	IDT49C3466				Obsoleto, 3.3volts
	IDT49C467				Obsoleto
Texas Instruments	SN54LS636	BUS WATCH	8-BIT	MILITAR /MIL-883	3 estados (salida)
	SN54LS637				Colector abierto
	SN74LS636			COMERCIAL	3 estados
	SN74LS637				Colector abierto
Mitel Semiconductors	MA31755	FLOW THRU	16 –BIT	COMERCIAL Y MILITAR/ MIL-883	OBSOLETO

INTERSIL	ACTS630MS	BUS WATCH	16-BIT	MILITAR/ ENDURECIDO CONTRA RADIACION/ MIL-PRF38535 CLASS V	Muestras comerciales disponibles, VIL= 0.8VCCMax. VIH= VCC/2Min.
	ACS630MS				Muestras comerciales disponibles, VIL=30%VCC VIH=70%VCC
TEMIC Semiconductors	29C516E	FLOW THRU	16-BIT	MILITAR Y ESPACIAL / ENDURECIDO CONTRA RADIACION/ MIL STD 883 CLASS B Or S	Capacidad para 2 CPU, Transacciones entre CPU's, y Memoria
	29C532E	BUS WATCH	32-BIT		Información preliminar.

Para la elección del dispositivo EDAC primero se efectuó un análisis general de las diferentes arquitecturas, con el que se observó cuál de ellas sería la más adecuada para la aplicación. De este análisis se obtuvo que para una arquitectura Bus-Watch sería necesario efectuar la sincronización de los ciclos de lectura y escritura del procesador y del EDAC, ya que el EDAC efectúa la lectura de datos de la memoria hacia el circuito y la escritura de los datos corregidos hacia procesador en dos ciclos diferentes, mientras que para el procesador ésta operación se efectúa en un solo ciclo, para lo anterior sería necesario desarrollar una máquina de estados finitos, además añadir circuitos de Transmisión/recepción (Tranceivers) para el bus de datos. En cuanto a la arquitectura Flow-Thru, se obtuvo que no sería necesario llevar a cabo una sincronización de ciclos de lectura y escritura ya que para el procesador y para el EDAC son los mismos, y por tanto el decodificador de memoria necesario emplearía solamente compuertas lógicas, además no sería necesario utilizar los circuitos de Transmisión/Recepción.

Por lo anterior se decidió utilizar una arquitectura del tipo Flow-Thru, lo que implicaría hacer menos cambios en la tarjeta de los procesadores, y por tanto efectuar menos cambios en el hardware de la computadora de vuelo.

De la Tabla 4.4 se observa que existen diferentes dispositivos EDAC de arquitectura Flow-Thru para elegir, pero la elección se ve limitada a los siguientes dispositivos: MA31755 y 29C516E de Mitel Semiconductors (anteriormente GEC PLESSEY Semi.) y TEMIC Semiconductors (Subsidiaria de ATMEL-WM) respectivamente, porque son los únicos componentes Flow-Thru que ofrecen una versión militar.

Inicialmente se optó por un diseño con el dispositivo MA31755 de Mitel, pero al efectuar la lista de compras de los componentes electrónicos y al hacer la revisión de existencias y de los periodos de entrega con los distribuidores, se encontró que ninguno de los proveedores en EU y Europa tenían en su inventario al dispositivo mencionado, por tanto se acudió a la página web del fabricante (www.mitel-semi.com) en la que se encontró un direccionamiento a la nueva pagina del fabricante (www.zarlink.com) en esta nueva pagina el dispositivo mencionado había sido descontinuado debido a que la compañía estaba siendo reestructurada. Por lo antes dicho el diseño se enfocó en el dispositivo 29C516 de TEMIC aún cuando este ofrece características extras, como el que puede ser empleado por múltiples usuarios, y que no serán utilizadas en el diseño de la computadora de vuelo.

4.5 Lógica de control para el dispositivo EDAC

El dispositivo EDAC 29C516E de TEMIC es un una unidad de detección y corrección de errores de muy baja potencia, con dos buses de datos de usuario. Durante un ciclo de escritura de procesador, añade una palabra de chequeo (6 ú 8 Bit de longitud) por cada localidad de memoria (16 Bit de longitud). Cuando realiza una operación de lectura, el 29C516E verifica la combinación entera de la palabra de chequeo y de datos. Puede detectar y corregir el 100% de los errores en un solo bit y detecta todos los errores en dos bit. Todos los errores son señalizados al sistema maestro (mediante 2 banderas de error) para permitir que el procesador efectúe la acción necesaria.

El 29C516E opera en dos modos posibles: Modo Corrección y Modo Detección. En el modo Corrección, los errores de 1 bit se corrigen, entonces la palabra corregida de datos se envía al puerto de salida y se activa la bandera "Correctable Error Flag" (bandera de Error Corregido). En el caso de errores de doble bit (o más), los datos corrompidos se envían al puerto de salida y se activa la bandera "Uncorrectable Error Flag". Hay que notar que algunos patrones de múltiples bits, pueden aparecer como posibles errores corregibles [TEMIC, 1997-2]. En nuestro caso la probabilidad de ocurrencia de este tipo de errores es despreciable [PISACANE, 1994].

El 29C516E actúa como un buffer de datos entre el microprocesador y la memoria. Este componente puede servir a dos usuarios diferentes del mismo espacio de memoria. El usuario 1(2) puede transferir datos de/hacia la memoria o de/hacia el usuario 2(1), esto último sin pasar por la memoria. Ambos usuarios tienen la posibilidad de escuchar las transacciones.

En la figura 4.3 se muestra el diagrama funcional del dispositivo 29C516E [TEMIC, 1997-2].

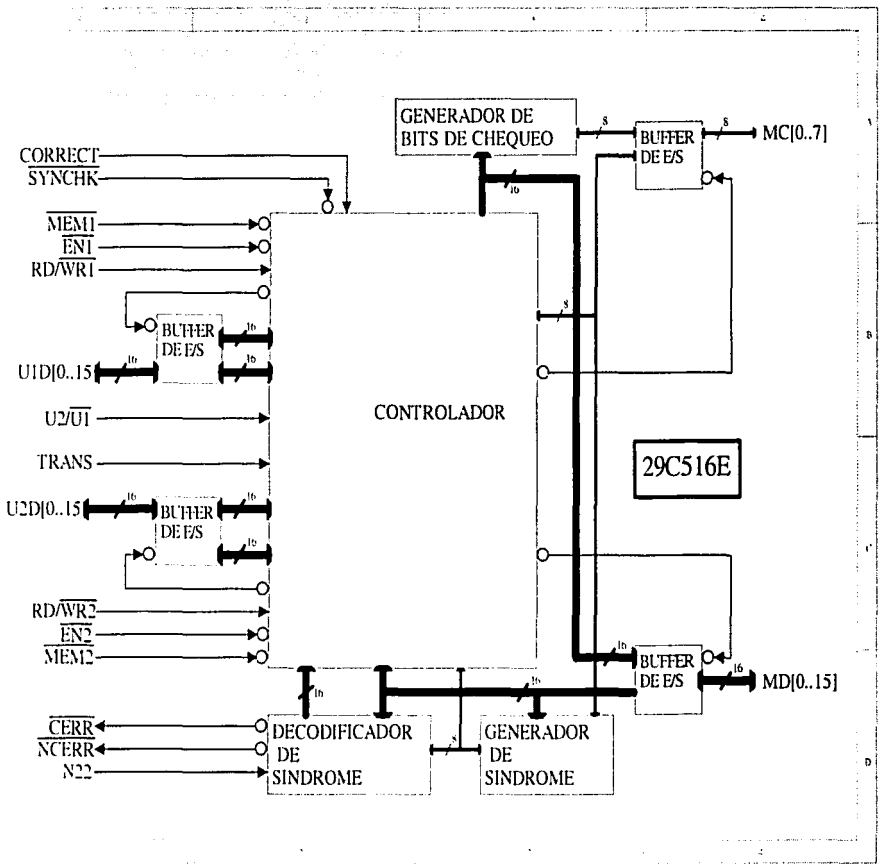


Figura 4.3: Diagrama funcional del 29C516E

El dispositivo 29C516E figura 4.4 proporciona las siguientes señales de control, datos y polarización [TEMIC, 1997-2]:

Nombre	Num. de alfiler	I/O	ACTIVO	DESCRIPCIÓN
BUSES				
U1D[0...15]	53,49..47,45..42,40..37,35..33,28	I/O*	ALTO	Datos del usr 1
U2D[0..15]	23..28,18..15,13..10,8..5	I/O*	ALTO	Datos del usr 2
MD[0..15]	59..62,64..67,69..72,74..77	I/O*	ALTO	Datos de memoria
MC[0..7]	83..86,88..91	I/O*	ALTO	Memoria de chequeo
BANDERAS DE ERROR				
CERR#	26	O	BAJO	Error corregible
NCERR#	25	O	BAJO	Error no corregible
SEÑALES DE CONTROL GENERAL				
CORRECT	98	I*	ALTO	Cuando esta activa, el EDADC esta en modo Corrección, si no, el EDAC esta en modo Detección
SYNCHK#	97	I*	BAJO	Selecciona si la palabra de síndrome y la palabra de chequeo son enviados por el bus de datos de usuario (byte alto y byte bajo respectivamente)
N22	27	I*	ALTO	Cuando esta activa, el EDAC usa 6 bit de chequeo, si no, usa 8 bit
TRANS	96	I*	A/ B	Selecciona la ruta de datos que se usara. Si es alto, el EDAC accede a memoria si es bajo accede al buffer de transferencia.
U2/U1#	95	I*	A/B	Selecciona quien es el maestro Usr1 o Usr2, el maestro es responsable de aplicar las señales: RD/WR#, MEMx# y ENx de forma correcta
SEÑALES DE CONTROL DEL USUARIO 1				
RD/WR1#	55	I*	A/B	Señal de lectura/escritura del usuario 1
EN1#	56	I*	BAJO	Habilitación de salida de usuario 1
MEM1#	57	I*	BAJO	Selector de memoria de usuario 1
SEÑALES DE CONTROL DEL USUARIO 2				
RD/WR2#	99	I*	A/B	Señal de lectura/escritura del usuario 2
EN1#	94	I*	BAJO	Habilitación de salida de usuario 2
MEM2#	3	I*	BAJO	Selector de memoria de usuario 2
BUFFERS DE POTENCIA				
VCC_B	9,19,32,41,54,63,73,87	I	---	Alimentación de buffers (5Volts Nominales)
GND_B	4,14,24,36,46,58,68,78,92	I	---	Referencia nominal de 0Volts de buffers
POLARIZACIÓN (NÚCLEO)				
VCC_C	100	I	---	Alimentación del núcleo (5Volts nominales)
GND_C	93	I	---	Referencia 0Volts del núcleo.
* BUFFERS PULL-UP				

Figura 4.4: Descripción de los alfileres del 29C516E

En el caso de la computadora de vuelo de SATEX, el dispositivo EDAC no utiliza su capacidad para servir a 2 usuarios, por lo que las líneas de control: **U2/U1#, TRANS,** se encuentran fijas a 5Volts (se eligió al usuario 2 por cuestiones de la localización de los buses de datos), además la línea **SYNCHK#**, también se encuentra fijada en 5Volts, esto se debe a que es improbable que ocurran errores múltiples en la palabra de datos. Además se optó por apagar el procesador cuando se detecte un error no corregible, ya que el aumento en el número de errores en la memoria RAM implica un aumento en la dosis de radiación recibida, lo que podría ocasionar el efecto Latch-up en el procesador.

La línea **N22,** se encuentra fija a 0Volts, para permitir la detección de todos los errores de doble bit en la palabra compuesta (Datos + Chequeo).

El esquema general de operación que debe cumplir el dispositivo EDAC, en la aplicación es el siguiente:

- Al comenzar operaciones, el EDAC se encontrará funcionando en el modo deshabilitado (no corrección), por lo cual durante este tiempo no podrá activar el circuito de eliminación de efecto Latch-up, esto es por que inicialmente el contenido de la memoria de chequeo no concordaría con los síndromes de la memoria de datos.
- Para activar el EDAC, será necesario realizar un ciclo de escritura en todas las localidades de memoria externa, ya sea expandida o normal. La activación del EDAC será efectuada por medio de la línea **ENEDAC** (alfiler 72 del microprocesador, denominado P2.10)
- Al activarse el EDAC deberá corregir todos los errores de un bit encontrados en la memoria RAM (Normal y Expandida), no en la ROM. Al corregir los errores, éstos serán notificados al procesador mediante el cambio de estado Alto->Bajo de la línea **CERRFLAG#** (alfiler 77 del procesador denominado P2.15).
- Si el EDAC es incapaz de corregir el error en la memoria RAM (errores múltiples), activará el circuito de eliminación de efecto Latch-up, si y solo si la localidad con error no pertenece a la memoria expandida, mediante una transición alto->bajo en la línea **NCERRFLAG#**, la cual es combinada con la salida del sensor de efecto Latch-up mediante una compuerta lógica AND. Esto es porque en la memoria expandida no se alojarán datos críticos para el buen funcionamiento de la computadora de vuelo, como será descrito posteriormente en este capítulo.

De lo anterior se obtiene la siguiente tabla de verdad:

Tabla 4.5: Tabla de Verdad para el control del EDAC.

Señales generadas por:	Decodificador de memoria	ROMSEL#	X	0	1	1	1	1	1
		XMEMISEL#	X	1	0	0	1	1	1
		RAMSEL#	X	1	1	1	0	0	0
	MCU	ENEDAC	0	1	1	1	1	1	
	EDAC	CERR#	X	X	0	1	X	0	1
		NCERR#	X	X	X	X	0	1	1
Señales de salida para:	EDAC	CORRECT	0	0	1	1	1	1	1
	MPU	CERRFLAG#	1	1	0	1	X	0	1
	Sensor de efecto Latch-up	NCERRFLAG#	1	1	1	1	0	1	1

Las líneas **ROMSEL#, XMEMSEL#** y **RAMSEL#** son generadas por el circuito de decodificación de memoria de la computadora e indican si el acceso es a ROM, memoria expandida o a RAM respectivamente, serán descritas con más detalle adelante en este capítulo.

De la tabla de verdad anterior se observa que la línea de salida denominada **CORRECT**, tiene el mismo valor que la línea **ENEDAC** si y solo si la línea **ROMSEL#** tiene un estado alto. La ecuación lógica que describe a esta línea queda como:

CORRECT = ENEDAC AND ROMSEL#

De la misma forma se obtiene que la línea **CERRFLAG#** puede generarse con:

CERRFLAG#= (NOT CORRECT) OR CERR#

Así mismo la línea **NCERRFLAG#** se obtiene como:

NCERRFLAG# = (NOTCORRECT) OR NCERR# OR (NOTXMEMSEL#)

Es importante hacer notar que el microprocesador al salir de un estado de reset, activa todas sus líneas de entrada/salida como entradas, es decir, con alta impedancia, por esto es necesario colocar resistencias de "Pull-up" y de "Pull-down" para fijar los valores necesarios. Es importante también hacer notar el problema que significa conseguir éste dispositivo, por esta razón se tomaron provisiones para el caso en que no se pueda conseguir, de tal forma que sí está ausente, no sea necesario efectuar cambios en las tarjetas del procesador.

4.5.1 Ciclos de Lectura y Escritura

El EDAC puede realizar tres tipos de transacciones de datos usuario a usuario, usuario a memoria y memoria a usuario, en esta aplicación, solamente se efectúan los tipos de transacción: memoria a usuario (Lectura de memoria) y usuario a memoria (Escritura de memoria).

4.5.1.1 Lectura de Memoria

El alfiler **TRANS** del EDAC se ubica en un nivel alto para seleccionar el acceso a la memoria. El alfiler **U2/U1#**, es fijado según el usuario que efectuará el acceso. Ambos alfileres mencionados se encuentran fijados a 5Volts. El usuario (en caso USR2) genera los comandos **RD/WRx#, MEMx#** y **ENx#**. En la tabla 4.6 se muestra la funcionalidad de los pines durante un ciclo de lectura de memoria [TEMIC, 1997-2].

Tabla 4.6: Funcionalidad del EDAC durante un ciclo de lectura

TRANS	U2/U1#	CORRECT	SYNCHK#	RD/WR1#	EN1#	MEM1#	RD/WR2#	EN2#	MEM2#	CERR#	NCERR#	FUNCIÓN
1	0	1	1	1	0	0	X	X	X	1	1	UD1[0..15] = MD[0..15]
										0	1	UD1[0..15] = MD[0..15] corregido
										X	0	UD1[0..15] = MD[0..15] corrupto
		0	1	1	0	0	X	X	X	X	X	UD1[0..15] = MD[0..15]
		X	0									UD1[0..15] = MC[0..7] | SINDROME
		X	X	1	1 / X	X / 1	X	X	X	X	X	UD1[0..15] = ALTA IMPEDANCIA
				1	X	X	1	0	0	X	X	UD2[0..15] = esperado (UD1[0..15]) (U2 escucha)
1	1	1	1	X	X	X	1	0	0	1	1	UD2[0..15] = MD[0..15]
										0	1	UD2[0..15] = MD[0..15] corregido
										X	0	UD2[0..15] = MD[0..15] corrupto
		0	1	X	X	X	1	0	0	X	X	UD2[0..15] = MD[0..15]
		X	0									UD2[0..15] = MC[0..7] | SINDROME
		X	X	X	X	X	1	1 / X	X / 1	X	X	UD2[0..15] = ALTA IMPEDANCIA
				1	0	0	1	X	X	X	X	UD1[0..15] = esperado UD2[0..15] (U1 escucha)

Para la aplicación, solamente se consideran los casos señalados en la tabla anterior. En la figura 4.5 se muestra el diagrama de tiempo del EDAC durante un ciclo de lectura:

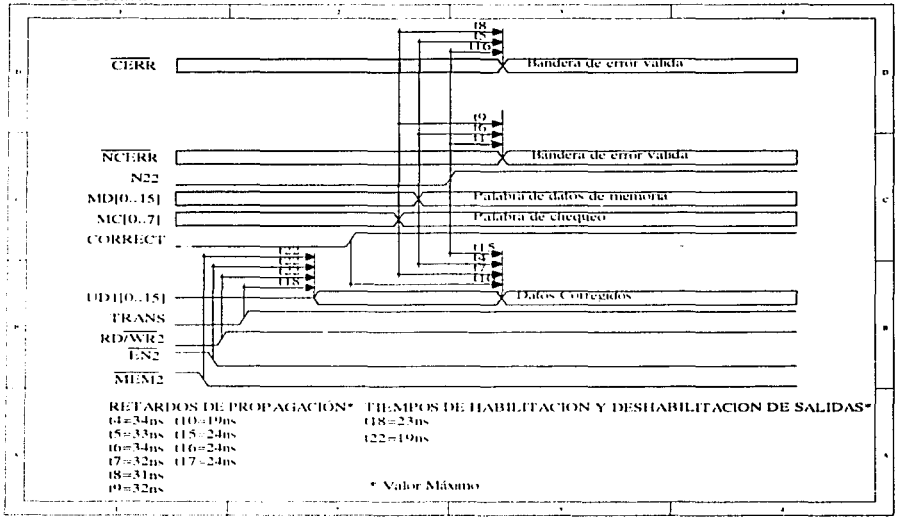


Figura 4.5: Diagrama de tiempo del ciclo de lectura del EDAC

4.5.2 Escritura a memoria

El alfiler **TRANS** se ubica en nivel alto para seleccionar el acceso a la memoria. Los controladores externos proporcionan las señales **U2/U1#, RD/WR2#, MEM2#** y **EN2#**. Todas las transacciones administradas por el usuario maestro pueden ser escuchadas por el otro usuario. En la tabla 4.7 se presenta un diagrama funcional del EDAC, durante un acceso de escritura a memoria [TEMIC, 1997-2].

Tabla 4.7: Tabla Funcional del EDAC durante un ciclo de escritura

TRANS	U2/U1#	RD/WR#	EN1#	MEM1#	RD/WR2#	EN2#	MEM2#	FUNCION
1	0	0	0	0	X	X	X	MD[0..15] = UD1[0..15] MC[0..7]=palabra de chequeo generada de UD1
		0	1	X	X	X	X	MD[0..15] = ALTA IMPEDANCIA MD[0..7] = ALTA IMPEDANCIA
			X	1				
		0	X	X	1	0	0	UD2[0..15]=UD1[0..15] (USR2 ESCUCHA)
1	1	X	X	X	0	0	0	**MD[0..15]=UD2[0..15]** **MC[0..7]=palabra de chequeo generada de UD2**
		X	X	X	0	1	X	**MD[0..15] = ALTA IMPEDANCIA**
						X	1	**MC[0..7] = ALTA IMPEDANCIA**
		1	0	0	0	X	X	UD1[0..15] = UD2[0..15] (USR1 ESCUCHA)

Para la aplicación, solo se toma en cuenta los casos señalados en la tabla anterior. En la figura 4.6 se muestra el diagrama de tiempo del EDAC durante un ciclo de escritura.

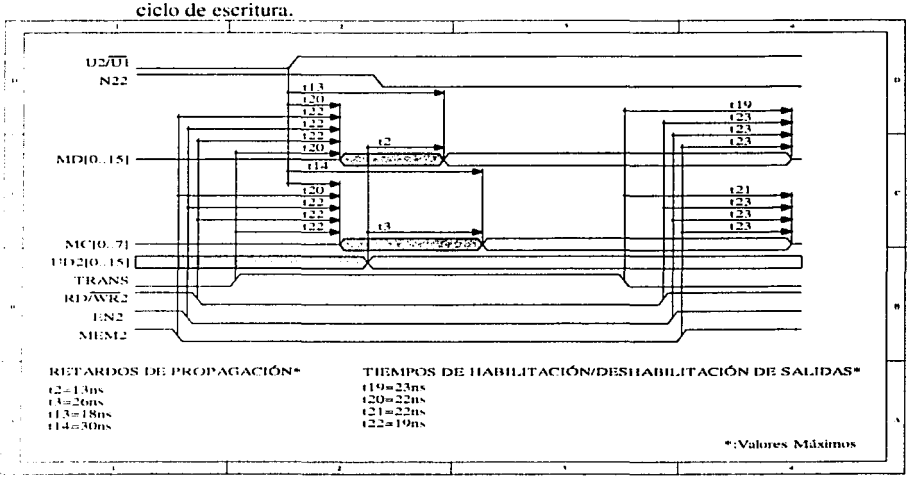


Figura 4.6: Diagrama de tiempo del ciclo de escritura del EDAC

Las líneas de control **MEM2#** y **EN2#** deben ser generadas siempre que exista un acceso a memoria, ya sea lectura o escritura, para esto el microprocesador proporciona 2 líneas de control: **RD#** y **WR#** que indican, cuando se encuentran en un nivel bajo, que está efectuando un acceso de lectura o de escritura respectivamente. Por tanto las líneas **MEM2#** y **EN2#** se pueden obtener mediante la siguiente ecuación lógica:

MEM2# = EN2# = RD# AND WR#

En cuanto a la señal **RD/WR2#** puede obtenerse mediante la línea **WR#** del microprocesador.

En la figura 4.7 muestra el diagrama esquemático del circuito de control del EDAC

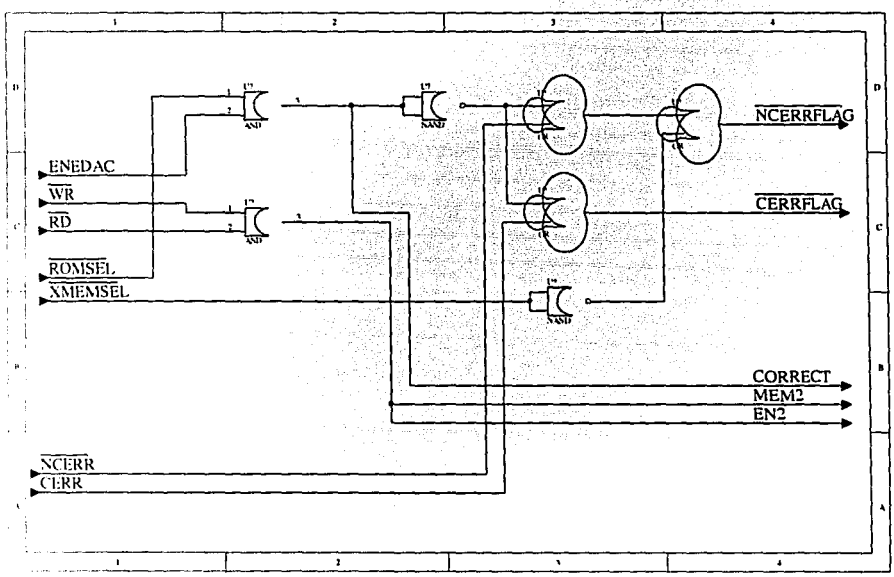


Figura 4.7: Esquemático del circuito de control del EDAC.

4.6 *Expansión de la memoria para el almacenamiento de imágenes*

El microcontrolador sab80c166 puede acceder hasta un máximo de 256 KB de memoria, mediante el uso de su controlador de bus externo (EBC) [SIEMENS, 1997], sin embargo, en la aplicación se utilizó un método de inmutación de bancos de memoria (bank switching) mediante el cual se accede a un máximo de 1,2 Mbytes de memoria. Con el uso de este método, el acceso a las localidades de memoria más allá de la 0x3FFFF (>256 KB) se efectúa de manera independiente del EBC.

Considerando que el espacio de memoria normal (256 KB) del procesador se divide en segmentos de 64 KB, el método propuesto para la ampliación de la memoria consiste en cambiar el lugar en el que se almacenan los datos correspondientes al último segmento de la memoria normal, mediante la generación de señales independientes al EBC, así por ejemplo, si existiera un chip de memoria RAM dedicado exclusivamente el almacenamiento del cuarto segmento de memoria, el método consistiría en cambiar físicamente el chip por otro sin borrar el contenido del primero (figura 4.8).

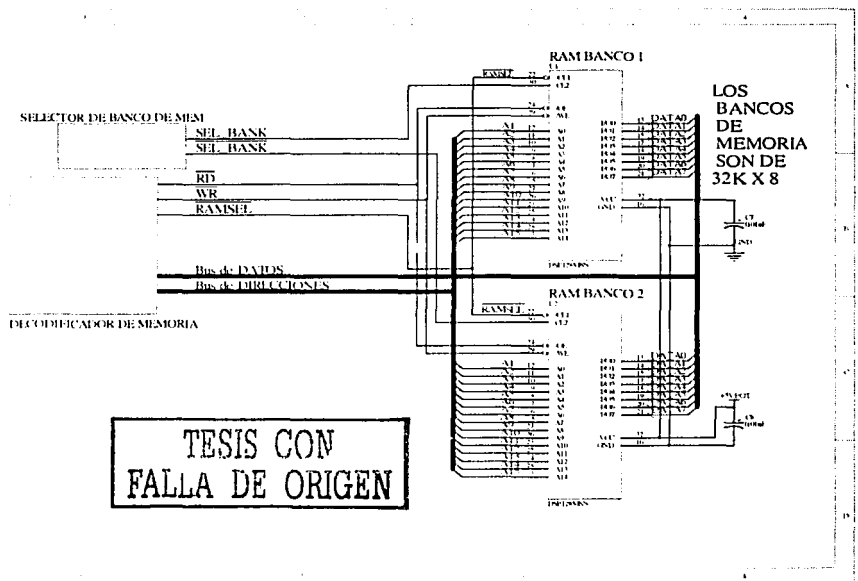


Figura 4.8: Ejemplo de conmutación de bancos de memoria para dos chips.

Este método se puede aplicar para "traslapar" tantos segmentos como sea necesario, donde el número de segmentos estará limitado por el número de señales de habilitación, trasladando esto para el caso en que los segmentos expandidos se almacenan en un mismo chip de memoria, se tendría que sumar un desplazamiento de 64 KB a la dirección base de los segmentos expandidos por cada uno de ellos, el valor del desplazamiento estará dado por la siguiente ecuación:

$$d = 64Kbytes \times NS_x$$

donde:

d= desplazamiento en Kbytes

NS_x= número segmento expandido, cuando NS_x= 0 se accede al segmento no expandido o segmento base.

En el caso de la memoria RAM normal de cada uno de los procesadores de la computadora de vuelo, se encuentra almacenada en 2 circuitos integrados de 128kx8 (bits), uno para los bytes de dirección par y otro para los bytes de dirección non. El 4° segmento constituye los últimos 32KBytes de cada uno de los circuitos, este espacio de direcciones de 64Kbytes se utiliza como segmento base de la memoria expandida.

El número de segmentos expandidos queda determinado por el número de líneas de entrada/salida del Microcontrolador que puedan ser usadas para el control de la memoria expandida. La cantidad de líneas libres fue de 4, con esto se obtiene que el número máximo de segmentos expandidos es de 16, ya que si tomamos éstas líneas como dígitos de un número binario, obtendremos un máximo de 16 posibles combinaciones ($16=2^4$) para indicar cada uno de los segmentos expandidos (numerados del 1-15) además del segmento base ó no expandido (numerado 0).

Con base en el número de segmentos expandidos se calcula el tamaño mínimo necesario de los circuitos de memoria para alojar los segmentos de la siguiente forma:

$T_{min}=NS_x*64KBytes=16*64KBytes=1024KBytes$

Por ello fueron elegidos 2 circuitos de 512Kx8, con lo que se obtuvo un tamaño total de 1024KBytes.

En cuanto a la decodificación de la memoria, se sigue el próximo esquema:

1. Los accesos al primer segmento de memoria (direcciones 0x00000h a 0x0FFFFh), se efectúan a la Memoria RAM o ROM contenida en los circuitos de memoria principal (2 circuitos de 128kx8 para RAM y 2 de 32Kx8 para la ROM), dependiendo del mapa de memoria seleccionado [Apartado 3.4.1].

2. Los accesos a los segmentos 01 y 02 (direcciones 0x010000h a 0x02FFFFh) de la memoria no expandida, se efectúan a la los circuitos principales de memoria RAM (128kx8).
3. Los accesos al segmento 03 (direcciones 0x030000h a 0x03FFFFh) de la memoria no expandida se efectúan a los circuitos principales de memoria RAM (128kx8), solamente si el indicador de segmento expandido (formado por las 4 líneas de control de memoria externa antes mencionadas) es igual a 0.
4. Los accesos a los segmentos expandidos, se efectúan en los circuitos de memoria expandida (2 circuitos de RAM de 512kx8) a través de las direcciones 0x030000h hasta 0x03FFFFh (direcciones del segmento base), solamente si el indicador de segmento expandido es diferente de 0.
5. los segmentos expandidos se almacenan en bloques contiguos de 64kBytes dentro de los circuitos de memoria expandida (512kx8), quedando el primer bloque de 64Kbytes inutilizado por el traslape del segmento base.

Para efectuar el esquema de decodificación de memoria son necesarias las siguientes líneas de control:

- **ROM/RAM#**: Nos indica el mapa de memoria utilizado [Apartado 3.4.1].
- **A16** y **A17**: Líneas de dirección generadas por el EBC, que indican el segmento de la memoria principal que se está accediendo.
- **A18, A19, A20** y **A21**: Líneas de control de memoria externa, generadas independientemente del EBC, forman el indicador de segmento expandido.

Además son necesarias las siguientes líneas de salida del decodificador para el manejo de los circuitos de memoria:

- **ROMSEL#**: Línea activo bajo que indica si el acceso es a memoria ROM (32kx8).
- **RAMSEL#**: Línea activo bajo que indica si el acceso es a los circuitos principales de RAM (128kx8).
- **XMEMSEL#**: Línea activo bajo que indica si el acceso es a los circuitos de memoria RAM expandida (512kx8).

Del esquema de decodificación anterior se obtiene la tabla 4.8

Tabla 4.8: Tabla de verdad del decodificador de memoria

Líneas de selección de segmento de memoria externa (controladas por el hardware del Microcontrolador)		Líneas de selección de segmento de memoria expandida (controladas por software)				Línea de selección de mapa de memoria	Líneas de habilitación de bloques de memoria (Líneas tipo activo bajo 0→Acceso 1→ No acceso)			Acceso al segmento de memoria	Acceso al segmento de memoria expandida
A17	A16	A21	A20	A19	A18	ROM/RAM#	ROMSEL#	RAMSEL#	XMEMSEL#		
0	0	X	X	X	X	0	1	0	1	00	No
0	0	X	X	X	X	1	0	1	1	00	No
0	1	X	X	X	X	X	1	0	1	01	No
1	0	X	X	X	X	X	1	0	1	02	No
1	1	0	0	0	0	X	1	0	1	03	0 - No
1	1	0	0	0	1	X	1	1	0	03	1
1	1	0	0	1	0	X	1	1	0	03	2
1	1	0	0	1	1	X	1	1	0	03	3
1	1	0	1	0	0	X	1	1	0	03	4
1	1	0	1	0	1	X	1	1	0	03	5
1	1	0	1	1	0	X	1	1	0	03	6
1	1	0	1	1	1	X	1	1	0	03	7
1	1	1	0	0	0	X	1	1	0	03	8
1	1	1	0	0	1	X	1	1	0	03	9
1	1	1	0	1	0	X	1	1	0	03	10
1	1	1	0	1	1	X	1	1	0	03	11
1	1	1	1	0	0	X	1	1	0	03	12
1	1	1	1	0	1	X	1	1	0	03	13
1	1	1	1	1	0	X	1	1	0	03	14
1	1	1	1	1	1	X	1	1	0	03	15

MODOS DE ACCESO A MEMORIA

Acceso a memoria ROM: el acceso a ROM solo puede efectuarse en el segmento de memoria externa número 00, mediante la configuración del mapa de memoria 1 (ROM+RAM).

Acceso al segmento 00 de memoria RAM, este segmento es accesado en el lugar de la memoria ROM, cuando el mapa de memoria seleccionado es el mapa 2.

Acceso a los segmentos 01 y 02 de memoria RAM, este acceso es independiente del mapa de memoria seleccionado, no requiere del establecimiento previo de los valores de las líneas A18 a A21, ya que es efectuado independientemente por el hardware de control de bus externo del microprocesador.

Acceso al segmento 03 de memoria RAM, antes de llevar a cabo este acceso es necesario establecer los valores de las líneas A18 a A21 como cero con la finalidad de evitar el acceso a un segmento de memoria expandida.

Acceso a la memoria expandida, se lleva a cabo mediante el solapamiento de un segmento de 64 Kbytes con el segmento 03 de la memoria RAM externa (no expandida) del microprocesador, para lo cual es necesario establecer el número del segmento expandido en el que se llevará a acabo la operación, a través de las líneas A18 a A21 y efectuar la lectura o escritura de datos en el segmento 03 del microprocesador, de esta forma para el microcontrolador, un acceso a un segmento expandido será siempre un acceso al segmento 03 de su memoria externa.

De la tabla anterior se observa que las líneas **A16** y **A17** son líneas que indican el segmento de la memoria direccionable por el procesador, y a partir de éste se definen los diferentes tipos de bloques de memoria: RAM o ROM externa y RAM expandida. Para la determinación de la línea **ROMSEL#** se obtiene la siguiente operación lógica:

ROMSEL# = A16 OR A17 OR (NOT (ROM/RAM#))

Para el acceso a la memoria expandida, la línea **XMEMSEL#** se puede obtener como:

XMEMSEL# = NOT ((A18 OR A19 OR A20 OR A21) AND (A16 AND A17))

La línea **RAMSEL#** se obtene a partir de las líneas **XMEMSEL#** y **ROMSEL#** mediante la siguiente operación lógica:

RAMSEL# = NOT(ROMSEL# AND XMEMSEL#)

Lo que se reduce a:

RAMSEL# = ROMSEL# NAND XMEMSEL#

De estas ecuaciones se obtiene el circuito lógico para la decodificación de memoria mostrado en la figura 4.9.

Para efectuar la suma del desplazamiento de 64Kbytes requerido para acceder a cada uno de los segmentos, se realiza el mapeo de las direcciones generadas por el microcontrolador a las direcciones utilizadas por los circuitos de memoria externa como sigue:

El número de líneas de dirección para acceder a una localidad dentro de cada bloque de 64kBytes se calcula como:

Tamaño = 2^N

Despejando N:
$N = \text{Log}_2$ (Tamaño)
N=16

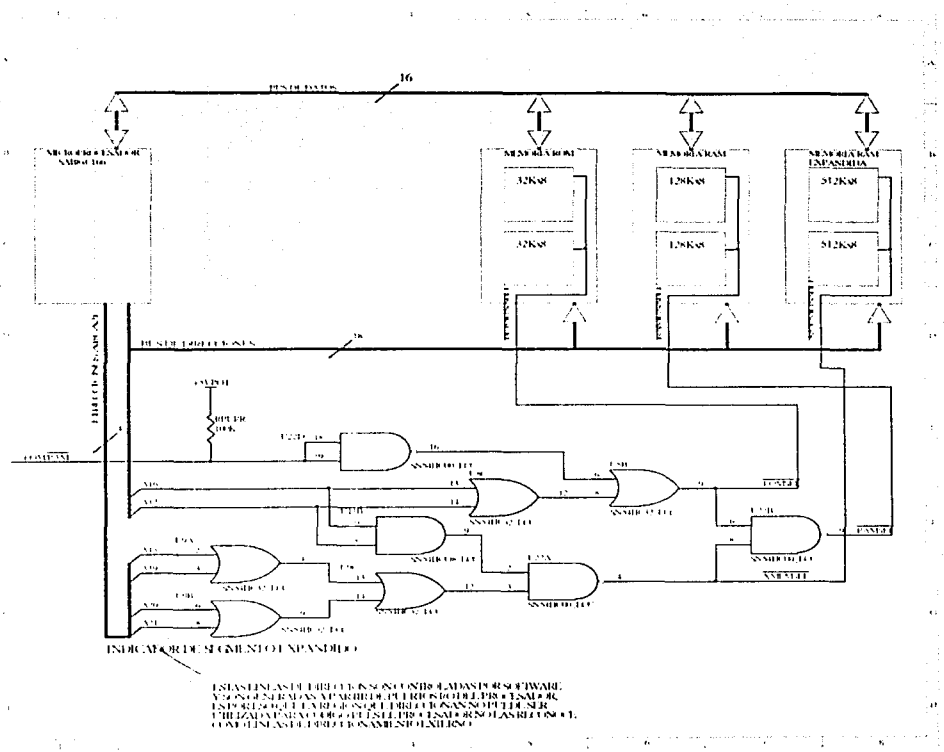


Figura 4.9: Diagrama lógico del decodificador de memoria

Todos los accesos a la memoria son de 16bit por lo que no es necesario direccionar cada Byte individualmente. Entonces se conectan las líneas de dirección A1 hasta A15 del microprocesador con las líneas de A0 hasta A14 de cada uno de los chips de memoria expandida, el desplazamiento de 64Kbytes estará dado por las líneas del indicador de segmento expandido A18 hasta A21 (generadas por el microprocesador independientemente del EBC), conectadas a las líneas A15 hasta A18 de los circuitos de memoria expandida como se muestra en la figura 4.10.

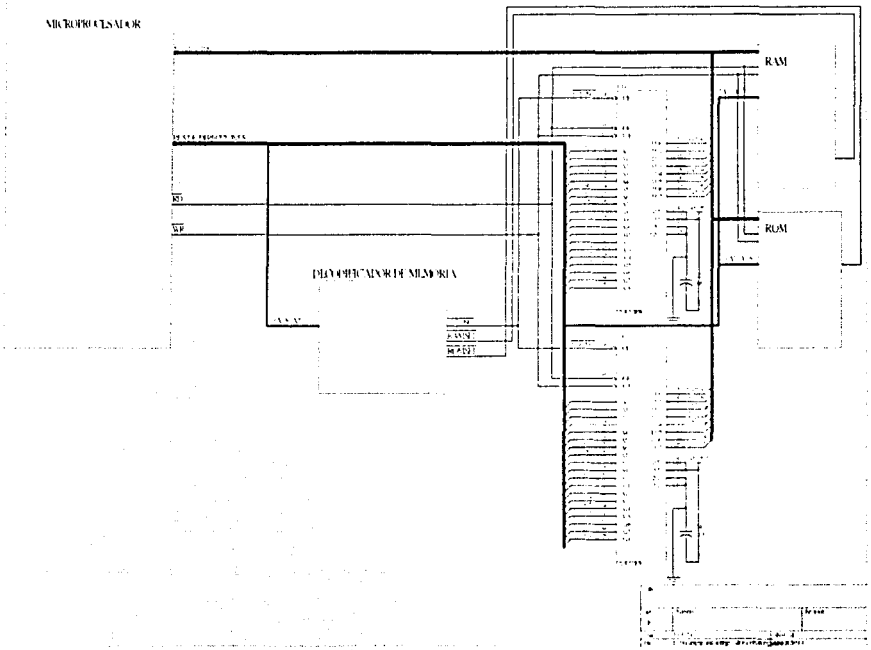


Figura 4.10: Diagrama del mapeo de memoria expandida.

4.7 *Integración del hardware de expansión de memoria y EDAC a la CV*

Debido a que el hardware de expansión de memoria y de EDAC fue diseñado teniendo en cuenta su interacción, no fue difícil obtener el diagrama de bloques final del subsistema MCU-EDAC-MEMORIA que se muestra en la figura 4.11.

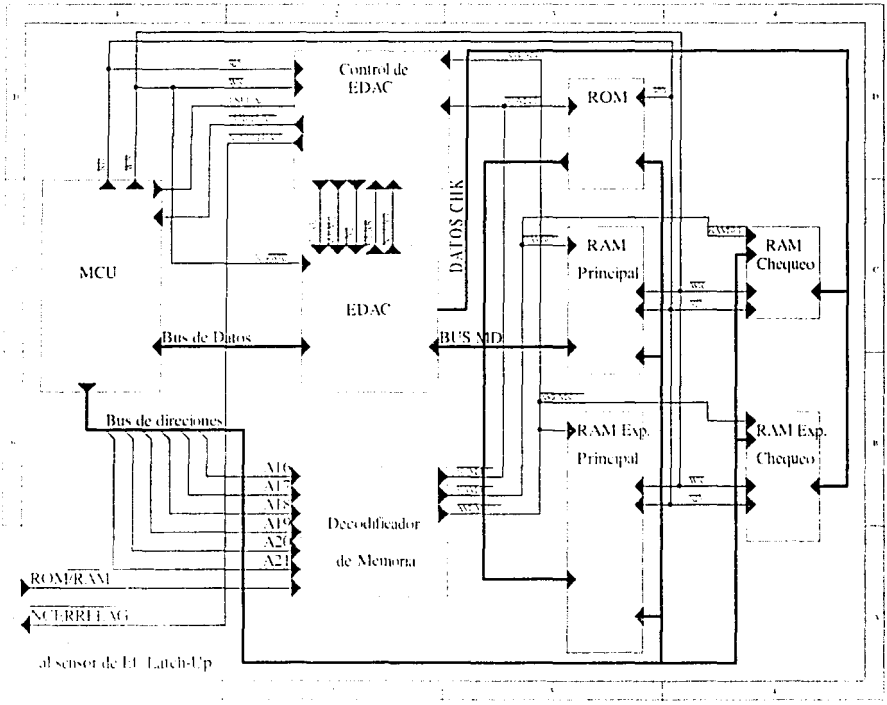


Figura 4.11: Diagrama de bloques del conjunto MCU-EDAC-MEMORIA.

Finalmente, a continuación se describen las consideraciones especiales que deben tomarse en cuenta para el uso del nuevo esquema MCU-EDAC-MEMORIA y que afectan directamente el desarrollo de software para la CV.

4.7.1 Acceso de 16bit a memoria externa

El microcontrolador SAB80C166, tiene un de bus de datos de 16bit, sin embargo, es capaz de efectuar accesos a memoria de forma parcial, es decir, con 8bit de longitud de palabra. En cuanto al EDAC 29C516E, este dispositivo no es capaz de efectuar accesos de escritura de forma parcial, para esto se requiere efectuar un proceso similar al siguiente:

- Cargar en el EDAC la palabra completa de 16bit de la localidad por escribir y cargar su palabra de chequeo correspondiente.
- Una vez que el síndrome del EDAC, se efectúa la combinación del Byte que se escribe con la palabra de datos (16bit) leída.
- Se efectúa la escritura en memoria de la nueva palabra de datos (16bit) y de la nueva palabra de chequeo generada.

Como se puede observar, el hardware de control no resulta trivial ni pequeño, lo que ocasionaría un aumento en la complejidad del procesador y en el número de componentes de la CV por otro lado resulta más simple y menos costoso (tiempo, dinero, confiabilidad, etc.) el limitar los accesos del microcontrolador para que se efectúen solamente de 16bit.

Con la finalidad de limitar los accesos a memoria a 16bit en el software de la CV solamente se deben tomar las 2 siguientes medidas:

1. Al desarrollar programas para la CV en leguaje C y compilados con el "C166 BSO/Tasking cross C Compiler, no se deberán utilizar variables o arreglos de variables almacenadas en memoria externa de tipo char (bytes).
2. Al desarrollar programas o algoritmos en lenguaje ensamblador, se deberá tener cuidado de no efectuar accesos de menos de 16bit a memoria externa, estos accesos se realizan cuando se usan instrucciones que toman operandos de 1Byte o de menor longitud (MOVB, ADDB, SUBB, etc.) [TASKING, 1993] [SIEMENS, 1997].

4.7.2 Almacenamiento de datos en memoria expandida

Anteriormente se mencionó que el procesador solamente puede acceder de forma directa a 256kbytes de memoria únicamente, sin embargo, por medio de la conmutación de bancos se incrementó la memoria. Debido al uso de esta técnica se debe tener cuidado de que el compilador no localice código o variables dentro del cuarto segmento de memoria, que funciona como espacio de acceso a la memoria expandida. De otra forma, si el compilador utiliza este espacio para el almacenamiento de código o variables, ocurrirían resultados inesperados cuando el EBC acceda a este espacio de direcciones mientras se encuentre activado el acceso a alguno de los segmentos expandidos ya que el EBC no distingue entre el acceso al cuarto segmento y el acceso a un segmento expandido.

Este espacio de memoria expandida se utilizará únicamente para el almacenamiento de datos de telemetría y de imagen mediante el uso de apuntadores, o en su defecto, mediante la localización de arreglos de variables previamente localizados

en el tercer segmento pero teniendo en cuenta que al acceder a diferentes segmentos expandidos, se estará accediendo a diferentes arreglos, aún cuando estos se encuentren aparentemente en el mismo espacio de memoria.

En la figura 4.12 se muestra un fragmento del código fuente del software de operaciones del satélite[6], mediante el cual se activa el segmento expandido en el cual se efectúa el acceso.

```
/*****************************************************
** función para inicialización de dir. Mem. Exp     **
** Inicializa los puertos de salida para las        **
** direcciones de la memoria Expandida A18-A20      **
*****************************************************/
void InitRamExp(void)
{
/*****************************************************
** Define Dp3.15, DP3.14, DP3.7 y DP3.6 como salidas **
** del microcontrolador                              **
*****************************************************/
   _putbit(1,DP3,15);
   _putbit(1,DP3,14);
   _putbit(1,DP3,7);
   _putbit(1,DP2,6);
   ActivaRamExp(0);
}

/*****************************************************
** funcion que activa un segmento(64Kbytes) de la ram **
** expandida, el numero permitido es de 0 a 15, si es **
** 0 se accesa al segemnto 03 de la memoria no exp    **
*****************************************************/
void ActivaRamExp(int NumSeg)
{
   int A18,A19,A20,A21; /*variables de cada direccion*/
   A18=(NumSeg&(0x01));          /* Segmento 0 */
   A19=(NumSeg&(0x02))>>1;     /* Segmento 1 */
   A20=(NumSeg&(0x04))>>2;     /* Segmento 2 */
   A21=(NumSeg&(0x08))>>3;     /* Segmento 3 */
/*****************************************************
** en las siguientes líneas se generan las direcciones **
** A18 A21                                           **
*****************************************************/
    _putbit(A18,P3,15);
    _putbit(A19,P3,14);
    _putbit(A20,P3,7);
    _putbit(A21,P2,6);
   uSegExpAct=NumSeg;
```

Figura 4.12: Acceso a segmentos expandidos (código C).

[6] En el APÉNDICE D de la presente tesis se muestra el código fuente de un programa de prueba completo para el acceso de memoria RAM expandida.

4.7.3 Tiempos de acceso a memoria

Del circuito esquemático mostrado en la figura 4.11, y considerando que los circuitos RAM tienen un tiempo de acceso 25ns se obtiene el diagrama de tiempos de acceso a memoria mostrado en la figura 4.13, de este diagrama se observa que el tiempo de acceso máximo a memoria es de aproximadamente 135ns. De acuerdo con las tablas de programación para los ciclos de acceso del microprocesador [SIEMENS, 1997], observa que para garantizar un acceso correcto a la memoria de la CV es necesario extender el tiempo de acceso del microprocesador en 2 ciclos de espera.

Para extender el tiempo de acceso es necesario programar el registro **MCTC** del SAB80C166 [SIEMENS, 1997] con el valor 0x1101b, y el registro MTTC con el valor 0x1b, la programación de estos registros debe efectuarse dentro del código de arranque de programas, ubicado en el archivo "estart.asm".

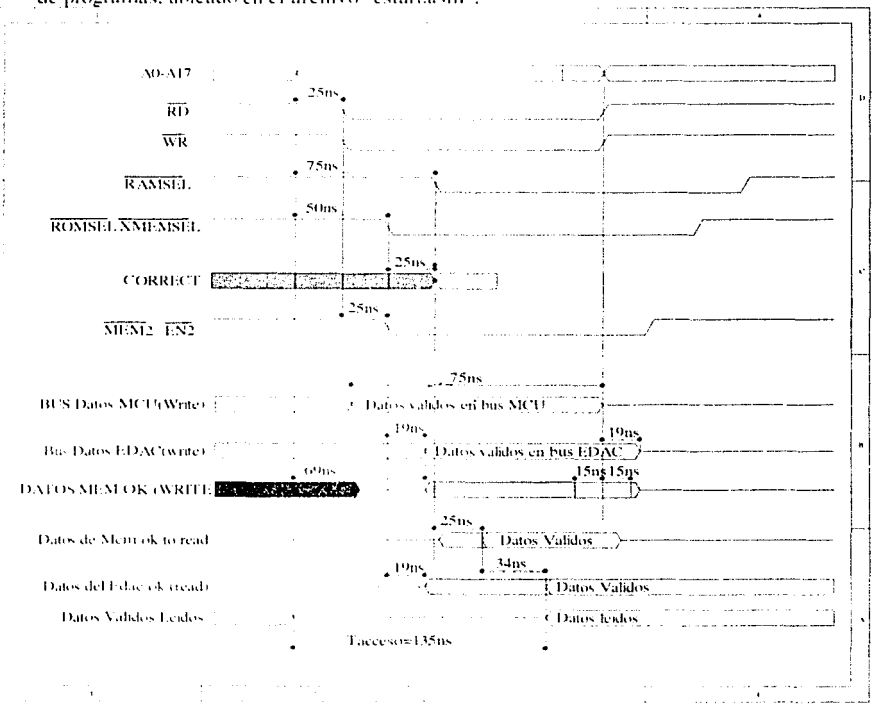


Figura 4.13: Diagrama de tiempo de acceso a memoria de la CV

Capítulo 5. Técnica de reconfiguración por vigía de tiempo y desarrollo del algoritmo de autodiagnóstico para CV

5.1 Introducción

Como fue señalado anteriormente [Apartado 2.3.9], una de las técnicas más empleadas en los sistemas tolerantes a fallas es la reconfiguración por vigía de tiempo o "Watchdog". En el proyecto SATEX se decidió su uso, debido a la simplicidad de su implantación y a su gran efectividad comprobada en los sistemas comerciales y espaciales.

Por otro lado, como parte de los experimentos de SATEX, se encuentra el uso de una arquitectura de cómputo semivirtual tolerante a fallas, que persigue realizar de forma automática el diagnóstico, detección y la reconfiguración ante fallas de la computadora principal de SATEX, es decir, la CV [TORRES, 2002].

A grandes rasgos, esta arquitectura de cómputo, consiste en efectuar un proceso de voteo democrático o Bizantino con los resultados de todos los diagnósticos de cada una de las computadoras a bordo del SATEX, para determinar cual de las computadoras presenta fallas.

Existen dos tipos de fallas que pueden presentarse en un procesador, estas son:

- Fallas silenciosas: Cuando el procesador se detiene y no responde.
- Fallas Bizantinas: Cuando el procesador continua operando, pero de manera incorrecta.

El proceso de voteo Bizantino[7] propuesto para SATEX, persigue detectar las fallas Bizantinas dentro de los procesadores, para esto es necesario el desarrollo de un algoritmo de autodiagnóstico que permita evaluar el buen funcionamiento de todos o de la mayor parte de los dispositivos que componen al microprocesador.

Con este capítulo se exponen las bases para utilizar del Watchdog del microcontrolador SAB80C166 dentro de la CV para lo que se requiere una interacción profunda con el entorno del compilador y del hardware empleado, sin embargo, no se pretende integrarlo dentro del software de operaciones de la computadora de vuelo, ya que esta última acción requiere del conocimiento profundo del software de control de la CV que es desarrollado como tema de otra tesis en el IIUNAM y que se encuentra fuera de los alcances de esta tesis. Por otro lado, con este capítulo también se pretende efectuar el análisis y desarrollo del algoritmo de autodiagnóstico del microprocesador. Con este desarrollo se determinan las bases para que este algoritmo pueda ser utilizado por las otras computadoras a bordo de SATEX.

[7] Información más detallada acerca del proceso de voteo bizantino a bordo de SATEX se puede encontrar en [Torres, 2002]

5.2 Requerimientos para el algoritmo de autodiagnóstico

El voteo Bizantino requiere disponer de un resultado con dos valores numéricos de un Byte [TORRES, 2002], el primero es el síndrome[8] de la computadora evaluada y el segundo es el resultado del diagnóstico de memoria RAM. Para las computadoras a bordo de SATEX, el síndrome es el valor numérico que define su estado operativo, este estará formado por bits de estado, que de estar activados indicarán una falla en alguno de los subsistemas del procesador[9].

Para el caso de la computadora de vuelo, el byte de síndrome deberá estar formado con los bits que se muestran en la tabla 5.1

Tabla 5.1: bits del síndrome de la CV

Bit	Falla detectada en
0	Unidad Aritmética Lógica "ALU"
1	RAM
2	Temporizadores T0 ó T1
3	Temporizadores T2 ó T3
4	Temporizador T4
5	Temporizador T5
6	Temporizador T6
7	Temporizador Watchdog

Para el Byte de diagnóstico de la memoria RAM se cuenta con dos opciones, una que comprende el uso del dispositivo EDAC y otra en el caso de que no se cuente con éste. En el caso de que no se cuente con el dispositivo EDAC, la palabra de diagnóstico estará formada por un código de falla, resultado de un barrido (escritura y lectura) en ocho zonas predeterminadas de la memoria, cada uno de los bits de esta palabra de diagnostico indicará el estado de falla de una zona de memoria mediante su activación. Para el caso en el que se encuentre presente el EDAC, la palabra de diagnóstico de RAM contendrá el número de errores en un solo bit detectados por el EDAC, si ocurriesen errores en dos o más bits, el EDAC automáticamente genera un reset en la CV

5.3 Arquitectura del microcontrolador SAB80C166

Para desarrollar el algoritmo de autodiagnóstico e implementar el uso del Watchdog para el procesador de la CV, se requiere el conocimiento profundo de la arquitectura del mismo. La CV cuenta con un microcontrolador RISC de 16bit SAB80C166[10] de Siemens, de tipo CMOS e industrial. A continuación se da una muy breve descripción de su arquitectura.

[8] Síndrome: Conjunto de signos y síntomas que constituyen un estado patológico y caracterizan el cuadro clínico de una enfermedad. Definición tomada de "Diccionario Enciclopédico Ilustrado Océano Uno, edición 1993".
[9] En SATEX cada uno de los procesadores que intervienen en el voteo genera su propio síndrome.
[10] Para conocer las características detalladas de este microcontrolador consulte [SIEMENS, 1997].

Entre sus características principales encontramos:

- CPU de 16bit de alto desempeño, con 4 etapas de "Pipeline", con un desempeño de hasta 10 MIPS (millones de instrucciones por segundo).
- Ciclo de instrucciones de hasta 100ns como mínimo.
- 256Kbytes de espacio de direcciones lineales para datos y código, con una arquitectura Von Neumann.
- Set de instrucciones reducido de alta eficiencia (RISC), orientado al control.
- Memoria RAM integrada en el circuito (1Kbyte).
- Interfaz de expansión para memoria Externa (External Bus Controler, EBC).
- Unidad de captura y comparación de canales Analógicos (CAPCOM UNIT).
- Sistema de Interrupciones con 16 niveles de prioridad además una unidad controladora de eventos periféricos ó PEC.
- 2 unidades de temporizadores de propósito general.
- 2 Canales de comunicaciones seriales (USART).
- 10 canales de conversión Analógica/Digital de 10bit
- Temporizador Watchdog con intervalos de tiempo programables.
- 76 líneas de Entrada/Salida con direccionamiento individual.
- Proceso CMOS Siemens de 1.2 micras.
- Rango de temperatura -40 a 110 °C.

En la figura 5.1 se muestra un diagrama de bloques funcional del SAB80C166, en las figuras 5.2 y 5.3 se presenta un diagrama de bloques de la estructura interna del SAB80C166 y un diagrama de bloques del núcleo del CPU.

Figura 5.1: Diagrama Funcional del SAB80C166.

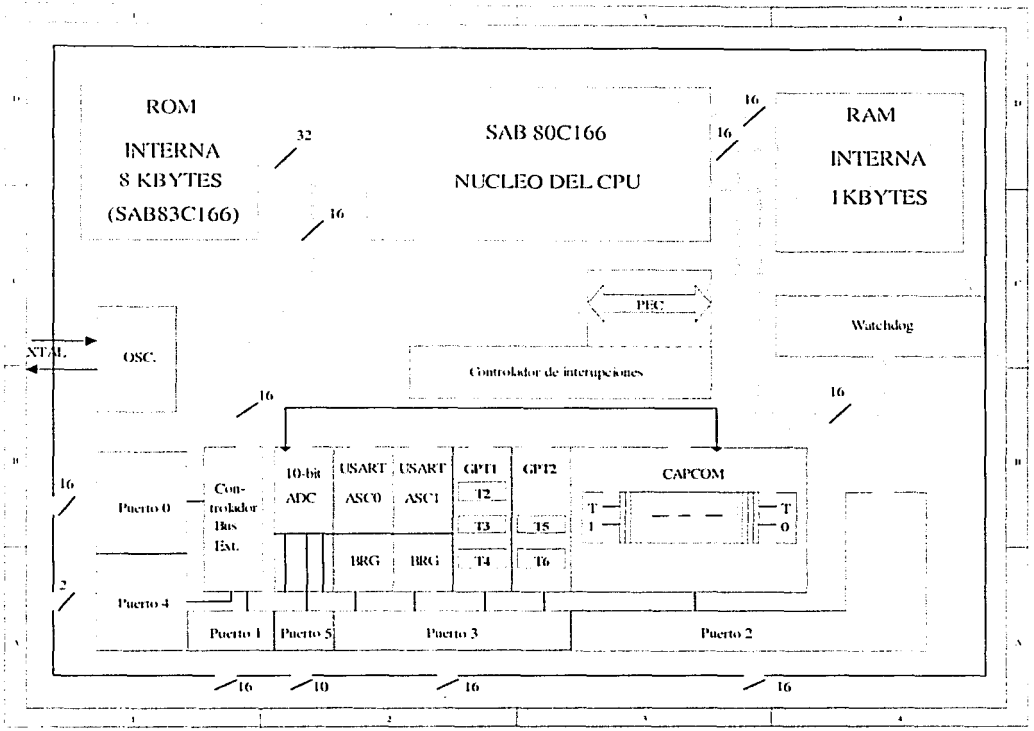


Figura 5.2: Diagrama interno de bloques del SAB80C166.

Figura 5.3: Diagrama de bloques del núcleo del CPU.

Cabe señalar que gran parte de las características que este microcontrolador ofrece son utilizadas por de la computadora de vuelo. Entre las características utilizadas por la computadora de vuelo se encuentran:

- ALU de 16bit: utilizada implícitamente durante la ejecución del programa de control.
- Sistema de interrupciones y PEC: ejecución programada de comandos, atención a los canales de comunicación, cuenta de errores en RAM (PEC), etc.
- 2 Unidades de Temporizadores (T0-T6): estabilización activa, generación de retardos de tiempo para sincronía de procesos, generación de señales de tiempo para la red redundante interna, etc.
- 6 canales de conversión A/D: adquisición de telemetría normal, monitoreo del estado de equipos (voltaje, corriente y temperatura), sensores de orientación (SFS's y Magnetómetros), etc.
- CAPCOM Unit: captura rápida de sensores para telemetría.
- 2 Canales de comunicación serial (ASC0 y ASC1): comunicación con Tierra y canal principal de red interna.
- Temporizador Watchdog: implantación de técnicas de tolerancia a fallas.
- EBC: Acceso a memoria Externa.
- Líneas de E/S: control de periféricos, memoria expandida, actuador, canal redundante de red interna, etc.

Como puede observarse, se utiliza la gran mayoría de las características del procesador, sin embargo, no todas ella pueden verificarse directamente por el algoritmo de autodiagnóstico como se verá adelante en este capítulo.

5.4 Características del Watchdog del microcontrolador SAB80C166

El Watchdog del SAB80C166, es un contador incremental de 16bit que se alimenta con una señal de reloj de frecuencia igual a la frecuencia del oscilador (f_{osc} [11]) dividida entre 4 o con $f_{osc}/256$. Los 8 bits más significativos del temporizador pueden programarse para obtener un mayor control sobre el tiempo del Watchdog. La figura 5.4 muestra un diagrama a bloques del Watchdog, así mismo las figuras 5.5 y 5.6 presentan los registros y alfileres asociados al Watchdog y una descripción de su registro de control **WDTCON**, respectivamente.

Después de cualquier tipo de reset al microprocesador (Software, Hardware ó Watchdog), el Watchdog se habilita y empieza a contar desde 0 con una frecuencia de oscilación $f_{osc}/4$. El Watchdog puede deshabilitarse mediante la instrucción **DISWDT** (Deshabilita temporizador Watchdog). **DISWDT** es una instrucción protegida de 32bit que se ejecuta durante el tiempo entre un reset y la ejecución de una instrucción **EINIT** (Fin de Inicialización) o de una instrucción **SRVWDT** (Servicio del Watchdog).

[11] Para la CV de SATEX, la frecuencia del oscilador es f_{osc}=40Mhz.

Cuando el Watchdog no se deshabilita, continuará contando aún en el modo "**IDLE**[12]", si este no recibe un servicio por parte de la instrucción **SRVWDT** dentro del tiempo que este alcance la cuenta de 0xFFFFh, el Watchdog se desbordará y causará un reset interno. Cuando el Watchdog causa un reset se activa la bandera **WDTR** y permanece así hasta que ocurre otro reset por hardware o hasta que se ejecuta la instrucción **SRVWDT**.

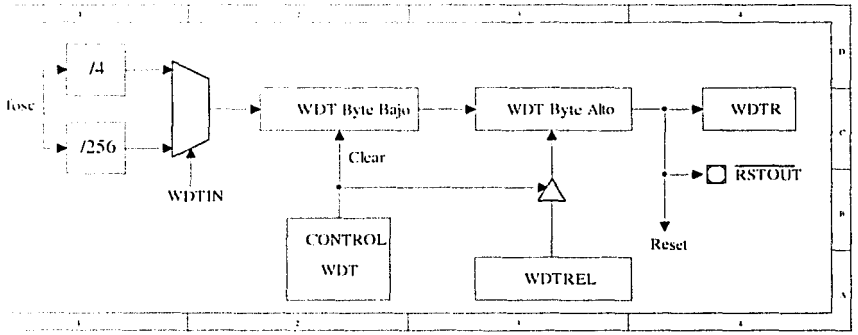


Figura 5.4: Diagrama de bloques del temporizador Watchdog

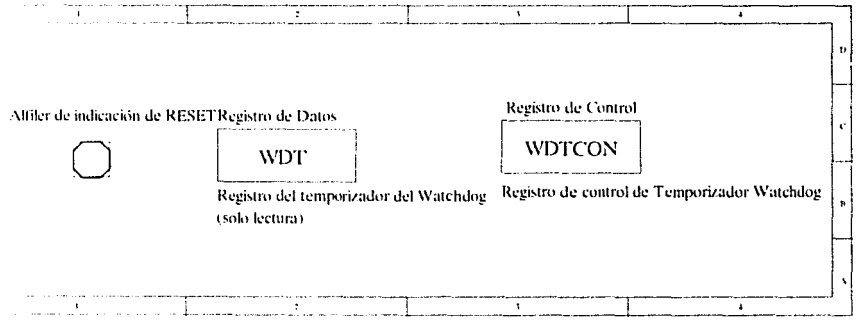


Figura 5.5: Registros y alfileres asociados al Watchdog

[12] Modo de ahorro de energía del microprocesador, para mas información consulte [SIEMENS, 1996].

WDTCON (0xFFAEh/D7 Valor de RESET: 0x0000h o 0x0002h

Simbolo	Posición	Función
WDTIN	WDTCON.0	Selector de frecuencia de entrada del Watchdog WDTIN=0: Fosc/4 WDTIN=1: Fosc/256
WDTR	WDTCON.1	Bandera de Indicación de Reset por Watchdog (Solo Lectu
WDTREL	WDTCON [15..8]	Valor de recarga del byte alto del temporizador Watchdog
—		Resevado

Figura 5.6: Registro de control del Watchdog WDTCON

En la tabla 5.2 se muestran los rangos posibles de tiempo para el Watchdog con una f_{osc}=40MHz, por razones de seguridad, se recomienda recargar el registro **WDTREL**, de cada servicio al Watchdog.

Tabla 5.2: Tabla de rangos de tiempo del Watchdog.

WDTREL	Multiplicador de f_{osc}	
	4 (WDTIN = 0)	256 (WDTIN = 1)
0xFFh	25.6 µs	1.6 ms
0x00H	6.55 ms	419 ms

5.5 Habilitación del Watchdog

5.2.1 Compilación de programas que utilizan Watchdog

El compilador utilizado para desarrollar el software de la computadora de vuelo es el compilador cruzado **80C166 Cross-Compiler BSO TASKING** [TASKING, 1993] de la compañía Tasking Boston Systems Office [www.tasting.com], el cual utiliza librerías específicas para el microcontrolador. El proceso de compilación a partir de un programa fuente en C, se muestra en la figura 5.7.

El compilador también es capaz de generar código ejecutable a partir de código fuente en lenguaje ensamblador.

Cuando se efectúa una compilación a partir de lenguaje C, es necesario enlazar el código objeto con la librería precompilada llamada "CSTART", en esta librería se encuentran definidas las opciones de arranque del microprocesador, opciones como: configuración inicial del registro **SYSCON**[13], código para la inicialización de variables, configuraciones de memoria externa, habilitación del Watchdog, etc.

Para cambiar cualquier opción de inicio para los programas que se compilan a partir de C, es necesario modificar el código fuente de la librería "estart", para esto el compilador incluye un directorio con el archivo fuente "CSTART.C" y un archivo de control de compilación para cada uno de los modelos de compilación (Small, Medium, huge, etc.).

Para habilitar el uso del Watchdog, se realizan los siguientes pasos:

a) Ubicar dentro de la librería "**CSTART.C**", a los comandos "**DISWDT()**"; y "**EINIT()**;", que respectivamente deshabilita y terminan la inicialización del Watchdog.
b) Sustituir el comando "**DISWDT()**"; por dos comandos, el primero, "**WDTCON=[Valor de configuración];**" para configurar el temporizador y el segundo, "**SRVWDT();**" para darle servicio.
c) Asegurarse que los comandos anteriores, se efectúen antes del comando "**EINIT()**";
d) Compilar el código fuente utilizando el archivo de control de compilación asociado.
e) Al compilar el código fuente del programa se debe, enlazar con el archivo "**CSTART.OBJ**" que se ha generado.

[13] **SYSCON:** Es el registro principal para la configuración del SAB80C166 (Memoria Externa, Tiempos de Acceso, Interrupciones, etc.), para mas información consulte [SIEMENS, 1997]

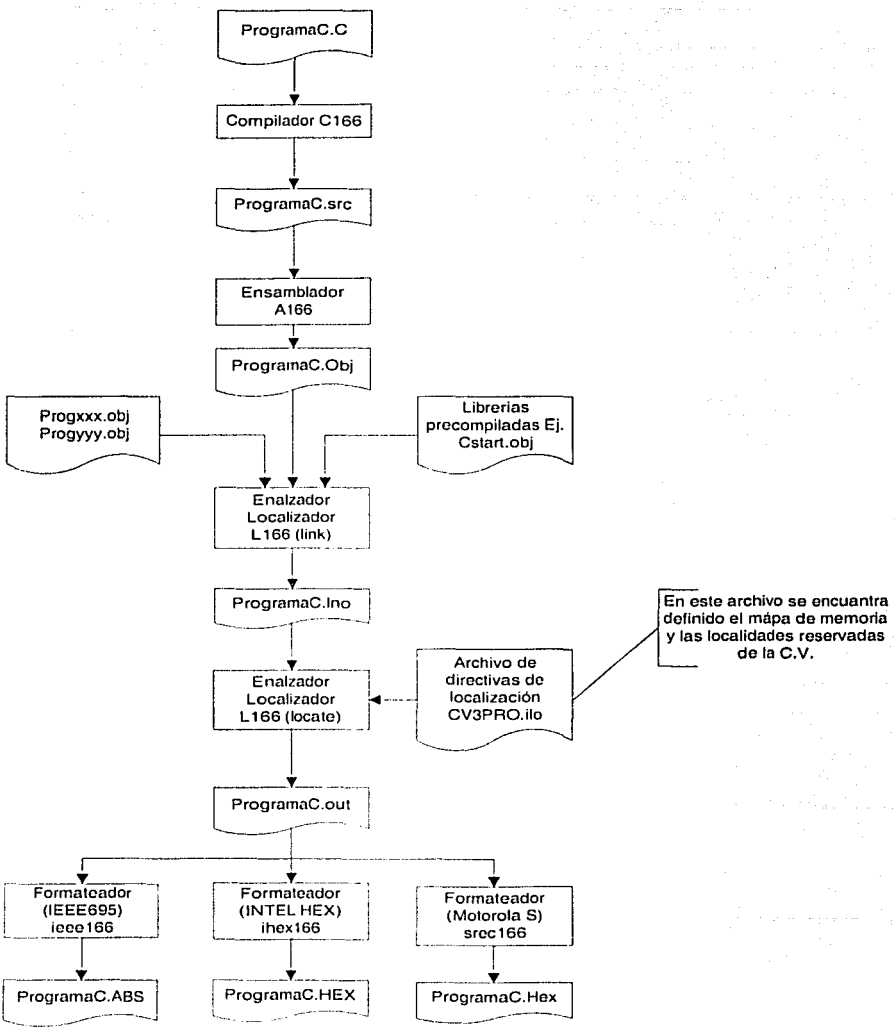


Figura 5.7: Proceso de compilación de software para el SAB80C166

5.6 Desarrollo de programas que usan el Watchdog

Para desarrollar programas que utilicen el Watchdog, es necesario, deshabilitar el dispositivo mediante el proceso descrito anteriormente, una vez que se tiene inicializado el Watchdog, se requiere aplicarle periódicamente un servicio a éste, este periodo de servicio en ningún caso deberá ser mayor al período de tiempo programado para el Watchdog.

Para la computadora de vuelo, se propone un período de tiempo de 419ms, el mayor posible, esto es porque los algoritmos de estabilización activa se ejecutan con limitantes de tiempo muy específicas, además de que ésta situación facilita la programación de comandos críticos para las operaciones del satélite.

Un mayor período de tiempo para el Watchdog, implica que se pueden realizar más instrucciones entre cada uno de los servicios.

El esquema propuesto para programar de los servicios al Watchdog en el software de operaciones de la CV, es el siguiente:

- Efectuar un servicio a Watchdog al inicio y al final de todas las rutinas y subrutinas del programa.
- Si la rutina en cuestión contiene ciclos de repetición (loop's), es recomendable efectuar un servicio al Watchdog, con la mayor frecuencia posible dentro del ciclo.
- Durante la ejecución en el modo IDLE del microprocesador, se deberá programar un temporizador para generar una interrupción, con la finalidad de despertar al microprocesador para que se efectúe un servicio al Watchdog

5.7 Desarrollo del algoritmo de autodiagnóstico

El algoritmo de autodiagnóstico forma parte del esquema de voteo bizantino, con el cual se pretenden detectar fallas bizantinas en cualquiera de las computadoras del satélite.

Las fallas que pretende cubrir el algoritmo de autodiagnóstico son:

- Fallas en la ALU.
- Fallas en memoria externa (EBC, RAM, EDAC, etc.).
- Fallas en los temporizadores (T0-T7).
- Fallas en el Watchdog.
- Fallas en los registros de trabajo.
- Fallas en memoria interna (registros de trabajo, stack, etc.).

5.2.2 Fallas en la ALU

Una falla en la unidad aritmética lógica del microprocesador, ocasionará que algunas de las operaciones lógicas (AND, OR, NOT, SHIFTL, SHIFTR,) y/o aritméticas (MUL, DIV, ADD) arrojen resultados erróneos. Una forma de saber si la ALU se encuentra trabajando de manera correcta es mediante la ejecución de operaciones lógicas y aritméticas sobre un operando conocido, de tal forma que si existiera algún error en las operaciones, este se propague a través de todo el proceso. La figura 5.8 muestra un diagrama de flujo del proceso de detección de errores en la ALU.

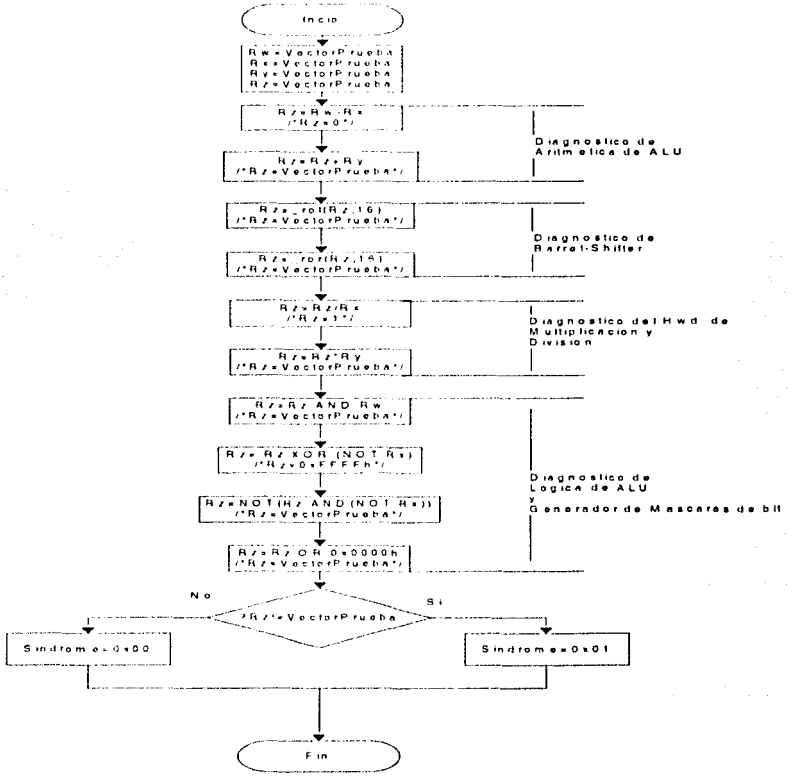


Figura 5.8 Detección de errores en ALU.

5.2.3 Fallas en memoria externa e interna

Las fallas en el acceso a la memoria externa e interna se determinan mediante los siguientes síntomas:

- Errores al efectuar accesos mediante el EBC. Lo que se lee no es lo mismo que fue escrito.
- Errores en la RAM debido a los SEU's. Si se encuentra presente la unidad EDAC, esta se encargará del manejo de errores en la memoria externa.
- Errores en la ejecución de rutinas. Cuando se presentan durante el proceso "Code Fetch[14]" del ciclo de procesamiento, el microcontrolador genera una interrupción que suspende la ejecución [SIEMENS, 1997].

La detección de fallas en memoria se efectúa de dos formas, una directa y la otra indirecta. La forma directa consiste en ejecutar un barrido[15] a toda la memoria externa excepto a la zona de memoria expandida (Sería tardado, además de que sus datos no son críticos), después de efectuar el barrido se comprueba el valor de la variable de conteo del EDAC, esta variable se incrementa automáticamente mediante una interrupción cada vez que se detecta un error en memoria RAM externa. En caso de que no se incluya el EDAC, esta variable no indicará nada en absoluto, sin embargo, si existen errores en alguna zona de memoria el barrido arrojará el resultado.

Por otro lado, además del barrido de memoria, se efectúa un muestreo en 8 zonas de memoria, distribuidas a lo largo de todo el espacio de memoria externa normal, en estas zonas de memoria se encuentran alojados 8 arreglos de 20 variables enteras (int) que contienen un valor preestablecido. La figura 5.9 muestra un diagrama de flujo del proceso de barrido para la detección de errores.

Cabe mencionar que para reducir la posibilidad de ocurrencia de SEU's en RAM, es recomendable efectuar barridos periódicos en las localidades de memoria, para reducir la acumulación de cargas debidas a la radiación.

La detección de fallas de forma indirecta es efectuada mediante el mismo proceso de ejecución del microcontrolador, es decir si se encuentra una falla durante la ejecución de instrucciones cargadas a partir de memoria RAM y estas instrucciones no son correctas, es decir no cumplen con el formato de instrucción esperado, el microprocesador lanza una interrupción de procesamiento ilegal [SIEMENS, 1997].

[14] Proceso de lectura de memoria de la siguiente instrucción que será ejecutada por el microprocesador.
[15] Un barrido consiste en un proceso de lectura, escritura y lectura y la comparación del valor escrito con el valor leído.

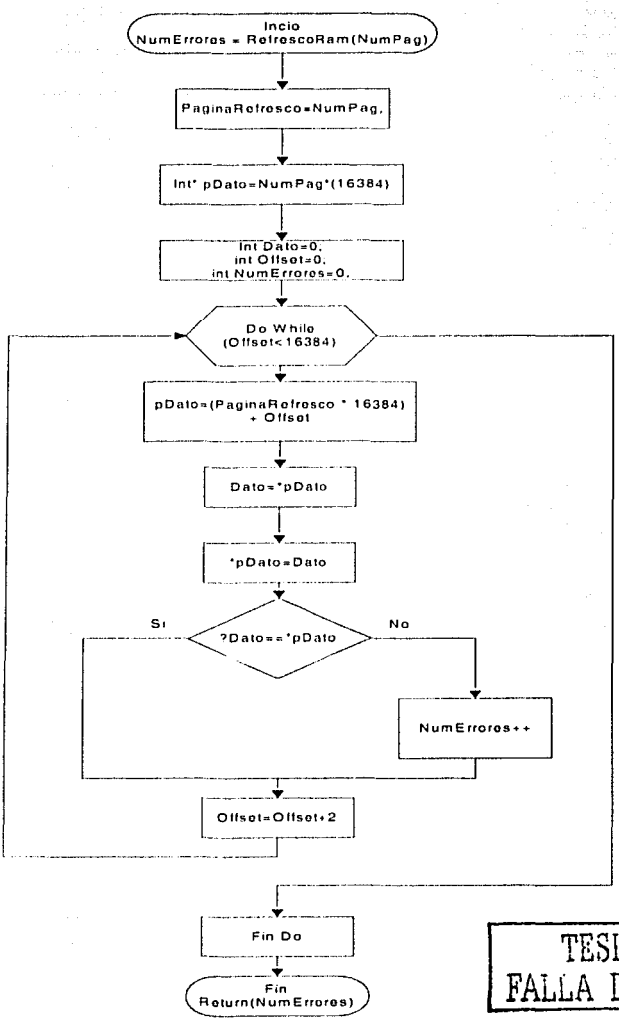


Figura 5.9: Proceso de barrido de memoria

5.2.4 Fallas en temporizadores

Las fallas en los temporizadores se pueden detectar de la siguiente forma:

- Registrar el valor del temporizador.
- Ejecutar varios comandos, teniendo en cuenta la resolución programada para los temporizadores [SIEMENS, 1997], de tal forma que se asegure que el contenido del temporizador aumente por lo menos una vez (rutina de espera)
- Al finalizar la rutina antes mencionada se obtiene el valor del temporizador y se compara con el obtenido inicialmente, si estos valores difieren se puede deducir que el temporizador está funcionando.

Para el diagnóstico de falla en los temporizadores de la computadora de vuelo, se almacena el estado actual de los temporizadores, luego se determinan cuales de los temporizadores no se utilizan en el software de vuelo (temporizadores libres), se ponen a correr solamente los temporizadores libres (los temporizadores ocupados ya están corriendo) y se toman las lecturas de valores iniciales para todos los temporizadores, esto se hace con el propósito de no interferir en la ejecución del software de vuelo. Finalmente después de la ejecución de la rutina de espera, se verifican los valores de todos los temporizadores, se generan las banderas de error correspondientes y se reestablecen los estados iniciales de los temporizadores libres.

En la figura 5.10 y 5.11 se muestra un diagrama de flujo para el diagnóstico de temporizadores.

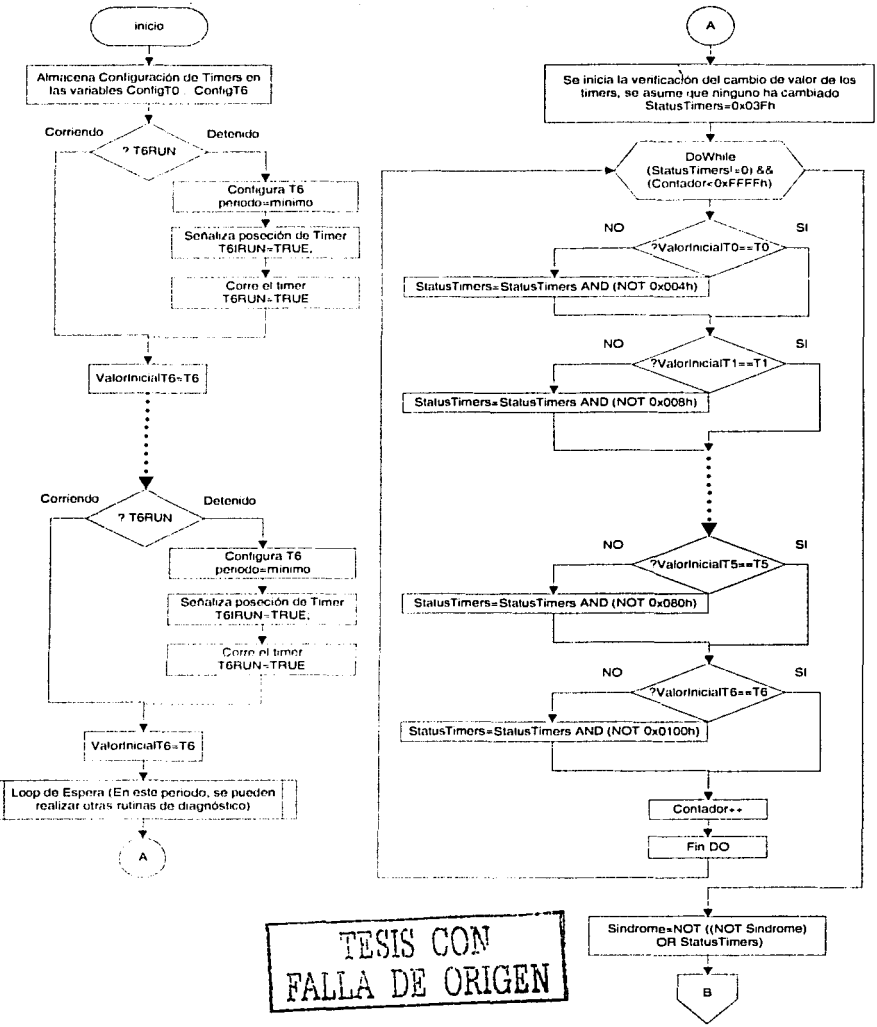


Figura 5.10: Diagnóstico de temporizadores.

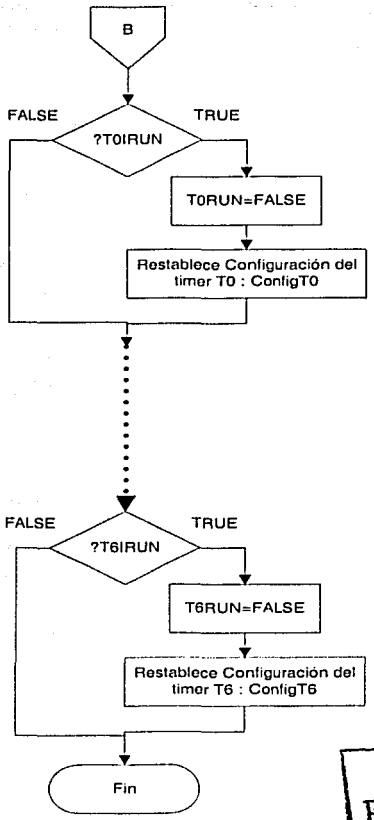


Figura 5.11: Diagnóstico de temporizadores (Continuación).

5.2.5 Fallas en el Watchdog

El diagnóstico de fallas para el Watchdog, se realiza de forma similar al diagnostico de fallas de temporizador es mencionado anteriormente, la única diferencia estriba en que para realizar el diagnóstico se nesecita habilitar el Watchdog.

5.2.6 Fallas en los registros de trabajo

El microprocesador cuenta con 16 registros de trabajo de 16bit R0-R15 que son usados por el compilador del C166 [TASKING, 1993] de la siguiente manera:

Tabla 5.3: Tabla de uso de los registros de trabajo del microprocesador.

Registro	Uso
R0	Apuntador a la pila de usuario
R1-R5, R10, R11	Registros generales (codegen, registros temporales, retorno de valores C)
R6-R9	Variables "register" de C y parámetros salvados como registros
R12-R15	Paso rápido de parámetros y variables registro en C
Registros utilizados para el retorno de parámetros de los siguientes tipos	
Tipo	Registros utilizados
Bit	PSW.6 (USR0)
Char	RL4 (parte baja de R4)
short/int	R4
Long	R4-R5 (R4-palabra menos significativa, R5-palabra más significativa)
near pointer	R4
far pointer	R4-R5 (R4-desplazamiento de página, R5 número de página)
Huge pointer	R4-R5 (R4-desplazamiento de segmento, R5 número de segmento)
structure	R4 o R4-R5 (apuntador near o far respectivamente)

Una falla en los registros del procesador se puede diagnosticar de forma indirecta, mediante la ejecución de una función que use el paso de parámetros y variables declaradas como "register". En el caso de la computadora de vuelo, para asegurar que se estuviesen utilizando los registros de trabajo, se empleó la característica del compilador C166 [TASKING, 1993] para generar código fuente ensamblador, de esta manera, del código generado para la rutina de diagnóstico, se buscaron las líneas correspondientes para las declaraciones de variables, de paso y retorno de parámetros de código C, para corroborar el uso de los registros de trabajo.

La figura 5.12 muestra un fragmento del código C de la rutina de autodiagnóstico y en la figura 5.13 se muestra el código ensamblador correspondiente. En estos fragmentos de código se muestra la declaración de las variables register, y su uso posterior en la rutina de autodiagnóstico.

```
Int DiagCPU(int VECTORP)
{

      register int Rw;  /* uso de registros R6-R9*/
      register int Rx;
      register int Ry;
      register int Rz;
            *
            *
            *
            *
            *
            *
      Rw=VECTORP;
      Rx=VECTORP;
      Ry=VECTORP;
      Rz=Rw-Rx;           /*resta*/ /*Rz=0x0*/
      Rz=Rz+Ry;           /*suma*/ /*Rz=VECTORP*/
      Rz=_rol(Rz,16);     /*shift left --rotar 16 *//Rz=VECTORP*/
      Rz=_ror(Rz,16);     /*shift right --rotar 16 *//*Rz=VECTORP*/
      Rz=Rz/Rx;           /* división*/ /*Rz=0x1*/
      Rz=Rz*Ry;           /* multiplicación*//*Rz=VECTORP*/
      Rz=Rz&Rw;           /* AND *//*Rz=VECTORP*/
      Rz=Rz^(~Rx);        /*XOR y NOT*//*Rz=0xFFFF */
      Rz=~(Rz&(~Rx));     /*NOT y AND *//*Rz=VECTORP*/
      Rz=Rz|0x0000;       /*OR*//*Rz=VECTORP*/

      if(Rz!=VECTORP)     /*verifica resultado*/
      {
            SINDROME=0x01; /*SINDROME DE FALLA DE ALU*/
      }

      else
      {
            SINDROME=0x0;      /* SINDROME SIN FALLAS */
      }
            *
            *
            *
            *
            *
            *
}
```


Figura 5.12: Fragmento de código fuente C de la rutina de autodiagnóstico.

```
        PUBLIC      _DiagCPU
        ?SYMB 'DiagCPU',_DiagCPU,37,16
        ?SYMB '',299,8,34
; Rw = R9 (automatic)
; Rx = R8 (automatic)
; Ry = R7 (automatic)
; Rz = R6 (automatic)
                *
                *
                *
                *
                *
                *
?LINE 505
        MOV     R9,[R0+#0376H]
        ?LINE 506
        MOV     R8,[R0+#0376H]
        ?LINE 507
        MOV     R7,[R0+#0376H]
        ?LINE 508
        MOV     R6,R9
        SUB     R6,R8
        ?LINE 509
        ADD     R6,R7
        ?LINE 510
        MOV     R4,R6
        ROL     R4,#00h
        MOV     R6,R4
        ?LINE 511
        MOV     R4,R6
        ROR     R4,#00h
        MOV     R6,R4
        ?LINE 512
        MOV     MDL,R6
        DIV     R8
        MOV     R6,MDL
        ?LINE 513
        MUL     R6,R7
        MOV     R6,MDL
        ?LINE 514
        AND     R6,#0AAAAh
        ?LINE 515
        MOV     R4,R8
        CPL     R4
        XOR     R6,R4
        ?LINE 516
        MOV     R4,R8
        CPL     R4
        AND     R4,R6
        CPL     R4
        MOV     R6,R4
        ?LINE 518
        MOV     R4,[R0+#0376H]
        CMP     R6,R4
        JMPR    cc_EQ,_40
        ?LINE 520
        MOV     R5,#01h
        MOV     [R0+#0192H],R5
        ?LINE 521
        JMPR    cc_UC,_41
```

```
_40:
      ?LINE 524
      MOV   R4,#00h
      MOV   [R0+#0192H],R4
      ?LINE 525
_41:
            .
            .
            .
            .
            .
      RETS
_DiagCPU    ENDP
```

Figura 5.13: Fragmento de código fuente en ensamblador de la rutina de autodiagnóstico.

De manera similar se procedió para verificar el uso de todos los registros de trabajo del microprocesador dentro de la rutina de autodiagnóstico, el código fuente en C de esta rutina completa se muestra en el apéndice C de la presente tesis.

5.8 Conformación del algoritmo de autodiagnóstico del procesador de la CV

A partir de los medios de diagnóstico que se han descrito, se conformo el algoritmo de autodiagnóstico para la CV, el diagrama de flujo correspondiente se muestra en la figura 5.14. La rutina de barrido de memoria se efectúa como una función independiente, con el propósito de que pueda utilizarse de forma periódica e independiente al autodiagnóstico para el refresco de la memoria RAM [Apartado 5.5.2].

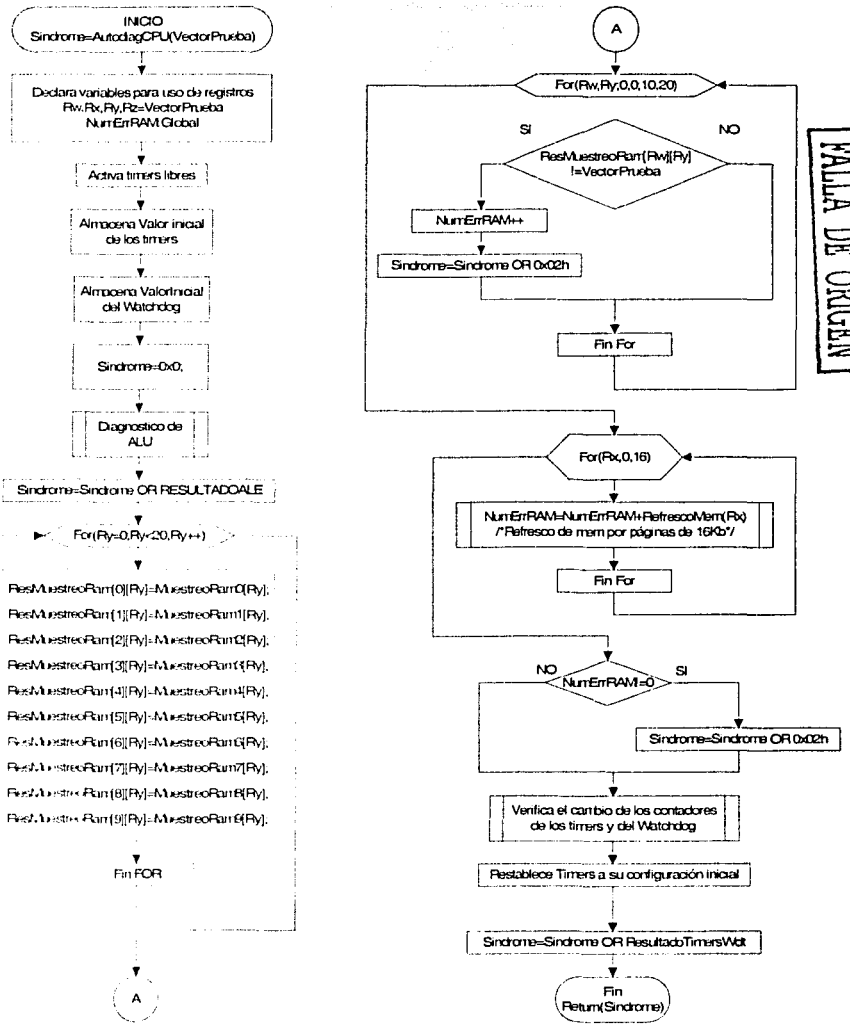


Figura 5.14: Diagrama de flujo del algoritmo de Autodiagnóstico de la CV

Capítulo 6. Diseño, validación y manufactura de circuitos impresos para la CV

6.1 Introducción

Al diseñar equipos electrónicos espaciales, deben considerarse varías restricciones, entre ellas el consumo de potencia, volumen, peso, resistencia a la vibración, etc. Para lograr estos objetivos, es común el uso de componentes reducidos y con una escala de integración cada vez mayor, además de materiales novedosos de alta resistencia y ligeros.

En el caso de SATEX, se consideraron las restricciones mencionadas, por lo cual se hizo uso extensivo de componentes eléctricos de montaje superficial o **SMT** (más del 90% de los componentes de la CV es de montaje superficial), se diseñaron circuitos impresos multicapa con componentes en ambos lados, se diseñaron contenedores ligeros, etc.

En este capítulo se describe e proceso de diseño de las tarjetas de electrónica, su captura en diagramas esquemáticos, validación de los mismos, así mismo el diseño y manufactura de los circuitos impresos.

En el capítulo siguiente, se describen el proceso de ensamble y las pruebas de validación de la computadora de vuelo.

6.2 "Protel", software de desarrollo electrónico

En un principio toda la electrónica asociada a la computadora de vuelo, fue desarrollada mediante el uso del software de diseño "Tango" [ACCEL, 1989], al inicio del desarrollo de esta tesis se analizó la posibilidad de utilizar la herramienta del software de diseño "Protel 99 SE SP6" [PROTEL, 1999] de la compañía "Protel Technologies" [www.protel.com], entre las características y factores más decisivos para la elección de esta opción se tuvieron las siguientes:

- Aplicación totalmente integrada al entorno de los sistemas operativos Windows NT4.x/9x.
- Todas las herramientas de diseño se integran bajo un mismo ambiente (esquemáticos, pcb's, simulaciones, etc.)
- Amplias bibliotecas de símbolos (más de 60,000), de componentes de diversos fabricantes, para la captura de esquemáticos.
- Simulación integrada de modo mixto, analógico y/o digital a partir del esquemático, mediante el uso de SPICE 3f5.
- Diseño de tarjetas de circuito impreso PCB's, controlado por reglas de diseño.
- Ruteador Automático, capaz de producir buenos resultados, aún en tarjetas complejas (más del 90% de ruteo)
- Ruteador interactivo, controlado por reglas de diseño.

- Analizador de integridad de señal en el PCB (Cross Talk, reflexiones, etc.).
- Amplia variedad de formatos de archivos de salida para la manufactura: Gerber, NC drill, PostScript, listas de materiales, reportes de verificación, etc.

El proceso de diseño y manufactura de un tabloide electrónico mediante el uso del software Protel, se puede describir de forma muy general como sigue:

1. Captura del esquemático.
2. Verificación del diseño eléctrico del esquemático.
3. Generación de la lista de redes del circuito.
4. Generación de la plataforma de la tarjeta impresa.
5. Cargado de la red del esquemático a la plataforma de la tarjeta impresa.
6. Posicionamiento de componentes en la tarjeta.
7. Ruteo de redes.
8. Verificación final de las reglas de diseño e integridad de señal de la tarjeta.
9. Generación de archivos de salida para la manufactura.

Es importante destacar que el proceso de diseño y manufactura de circuitos electrónicos comprende más etapas que las antes mencionadas, ya que debe tomar en cuenta diversos factores como: los costos de manufactura y componentes, disponibilidad de partes, el uso final del equipo, pruebas y reparaciones, actualizaciones, etc[10]. [GINSBERG, 1991].

Con la finalidad de simplificar la descripción del proceso de diseño y manufactura de los circuitos de la CV se adopta la forma general descrita anteriormente, en ésta se observar que los dos primeros incisos implican, el trabajo con el diagrama esquemático, los tres siguientes conforman la interfaz entre los diagramas esquemáticos y las tarjetas impresas, y finalmente los puntos restantes implican el trabajo con la tarjeta impresa y la preparación para su manufactura.

6.3 Captura de circuitos esquemáticos

Como se ha descrito en capítulos anteriores, la electrónica que conforma la computadora de vuelo se encuentra distribuida en 6 tarjetas impresas (3 TCV, 1 CONTROL, 1 MUXFIL, 1 TCVLATCH), los 4 circuitos esquemáticos correspondientes a estas tarjetas fueron capturados a partir de los esquemáticos iniciales capturados con el software "Tango Schematic" [ACCEL, 1989]. Todas las modificaciones propuestas en la presente tesis fueron incorporadas al efectuar el proceso de captura de los esquemáticos con "Protel", esté proceso se llevo a cabo mediante el cableado de los símbolos que comprenden el diseño. Los símbolos de componentes se cargan a partir de las bibliotecas incluidas, o dibujados mediante la herramienta de edición de símbolos.

[10] Una descripción más detalla del proceso de manufactura de circuitos, es encontrada en [GINSBERG,1991]

La captura de los circuitos esquemáticos, se realizó de manera jerárquica, es decir, cada uno de los esquemáticos de las tarjetas forma parte de un esquemático principal, que simboliza a la computadora de vuelo. Esto se realizó para manejar la generación de reportes de materiales y verificaciones de manera global.

En las páginas subsecuentes, las figuras 6.2 6.5 muestran los circuitos esquemáticos asociados a cada una de las tarjetas de la computadora de vuelo, la descripción del hardware contenido en éstas puede consultarse en el capítulo 2.

6.4 *Verificación del diseño eléctrico de los esquemáticos*

El proceso de validación del diseño eléctrico del esquemático de las tarjetas, consiste en verificar la conexión correcta de los componentes que lo forman, así por ejemplo, dos pines de salida de dispositivos electrónicos no pueden estar conectados a un mismo punto, dos redes no pueden tener el mismo nombre ya que esto podría implicar un corto circuito, etc. En esta etapa, Protel ofrece una herramienta para efectuar la verificación de reglas eléctricas como las descritas, generando un reporte con los errores y advertencias basado en una matriz de reglas definidas por el usuario. En la figura 6.1 se muestra un ejemplo de la matriz de reglas de chequeo.

Como fue mencionado antes, el diseño fue capturado de manera jerárquica, esto sirvió para efectuar la verificación eléctrica de forma global y de forma particular para cada tarjeta, y así poder identificar posibles errores en la compaginación de las redes de las diferentes tarjetas.

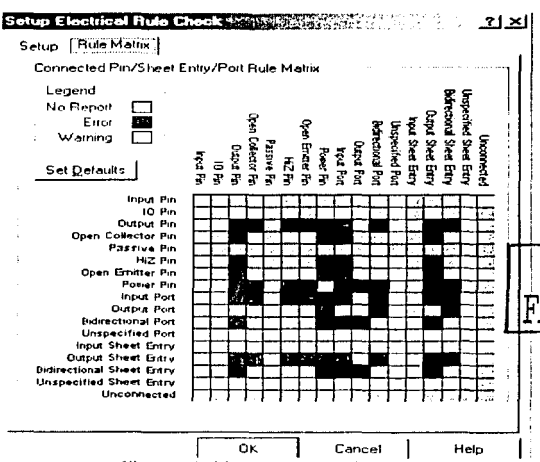


Figura 6.1: Matriz de reglas eléctricas.

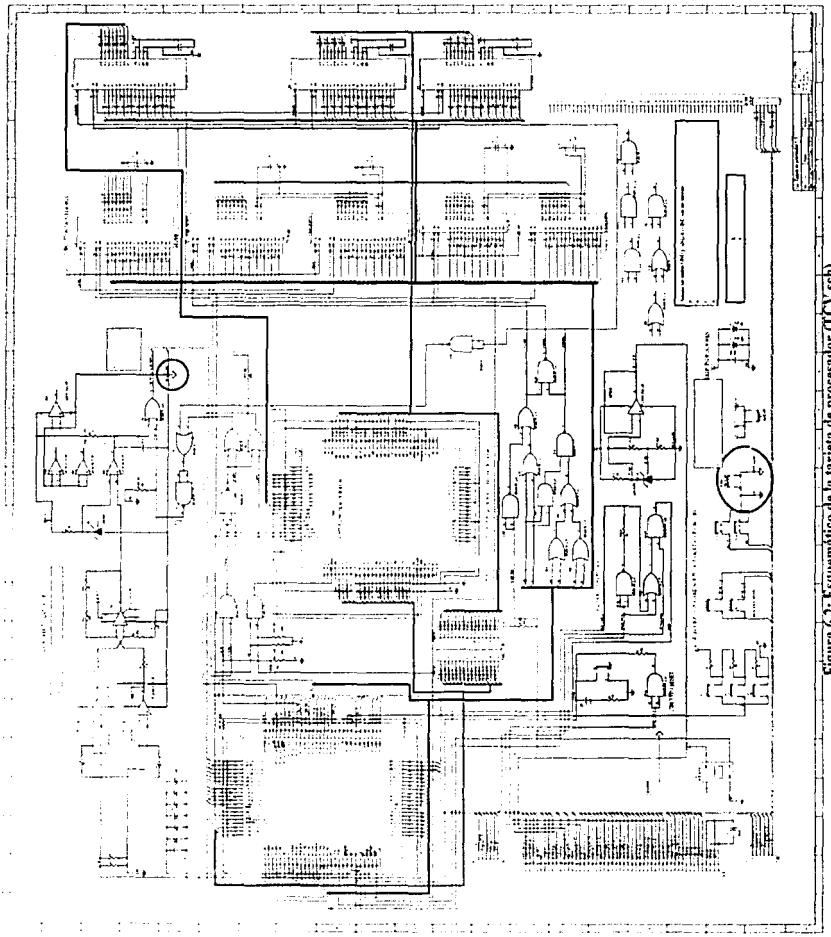


Figura 6.2: Esquemático de la tarjeta de procesador (TCV.sch)

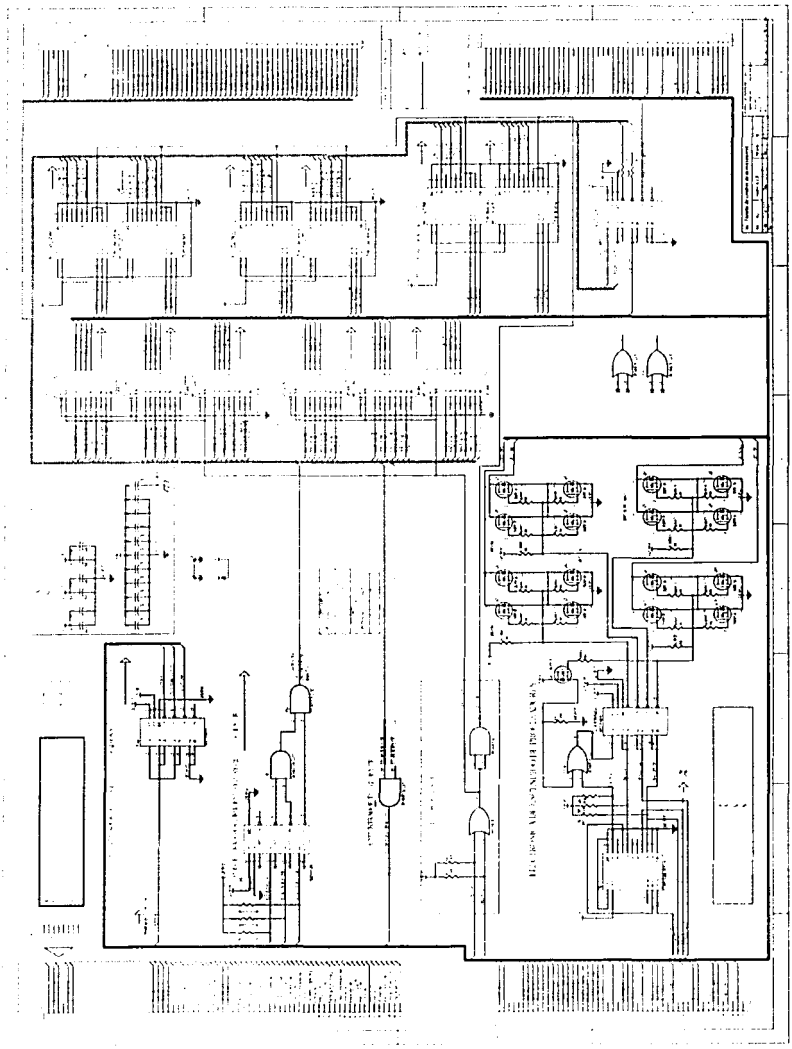


Figura 6.3: Esquemático de la tarjeta de control de procesadores (TCVCTRL.sch)

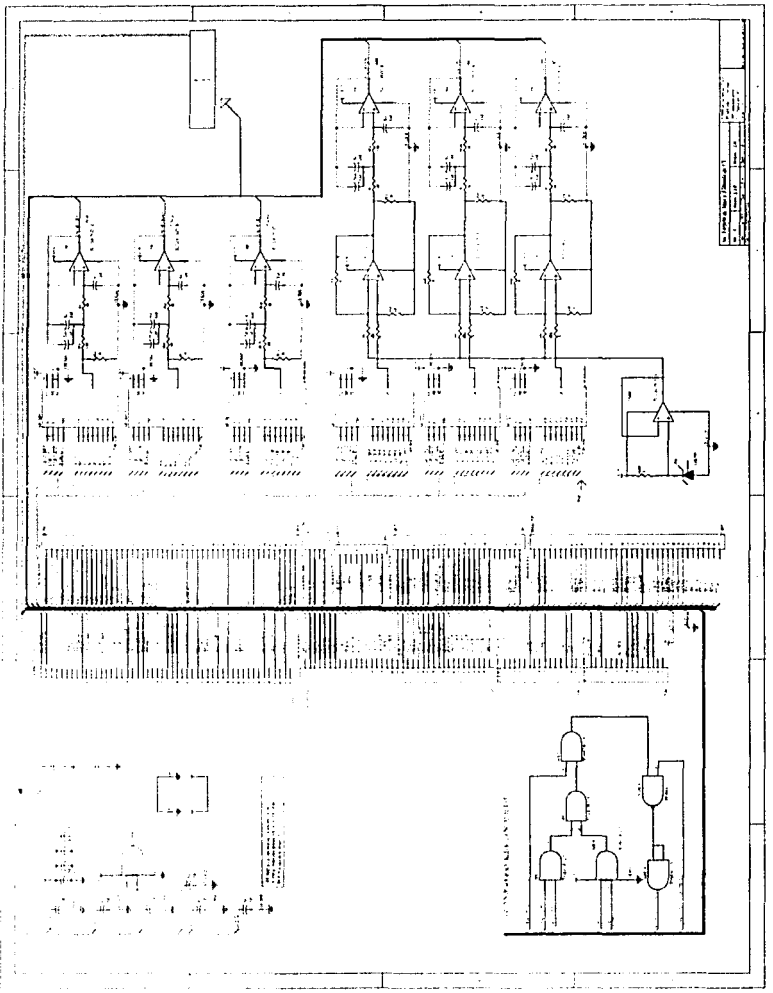


Figura 6.4: Esquemático de la tarjeta de acondicionamiento y multiplexaje de señales de sensores (TCVMUXFILT.sch)

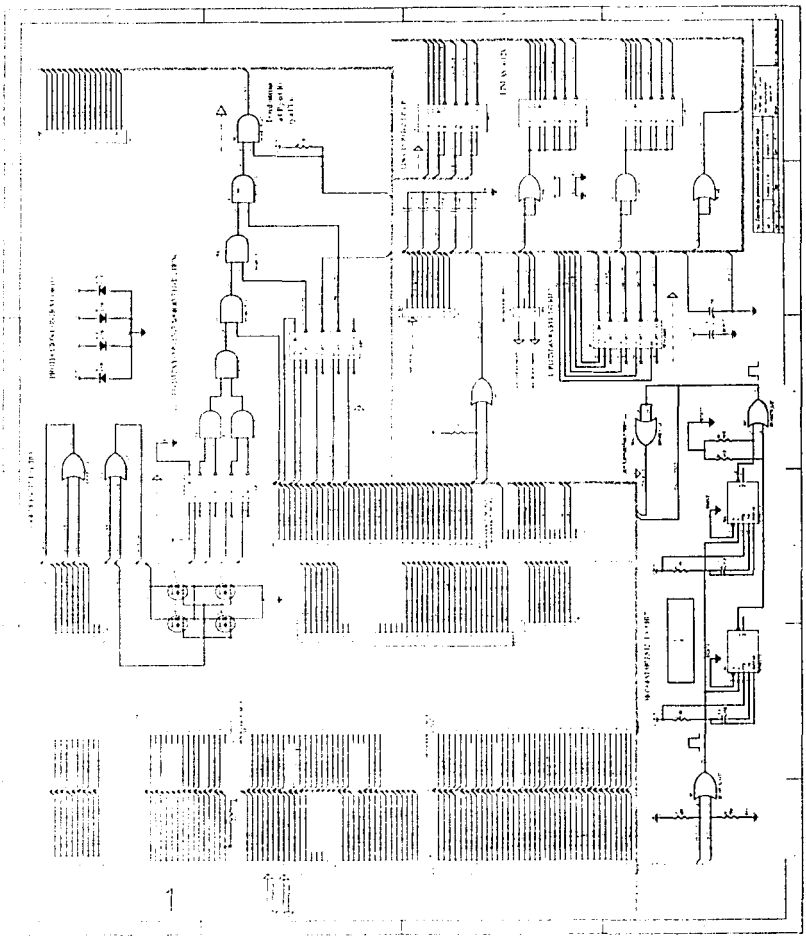


Figura 6.5: Esquemático de la tarjeta de de protección de efecto "Latch-Up" (TCVLACTHUP.sch)

6.5 Generación de las listas de redes

Esta etapa consiste en generar una lista de todos los componentes y sus conexiones eléctricas dentro del circuito, así como las huellas que representarán físicamente a los componentes en las tarjetas impresas.

La generación de la lista de redes para la electrónica de la computadora de vuelo, se efectúo de manera particular para cada una de los esquemáticos de las tarjetas de electrónica, obteniéndose así 4 listas de redes denominadas:

- TCV.NET
- TCVCTRL.NET
- TCVMUXFILT.NET
- TCVLATCH.NET

6.6 Generación de la plataforma de las tarjetas impresas

En esta etapa se genera una plataforma en la cual, por un lado se especifican las dimensiones, el número de capas, orificios de fijación y estilo de componentes de la tarjeta impresa, y por otro se especifican las reglas de diseño que deberán seguirse durante el proceso de localización de componentes, ruteado y manufactura de la tarjeta. Durante esta fase se especifican por ejemplo: ancho de pistas, separación entre objetos (pistas, componentes, vías, etc.), capas de rutéo, longitud máxima de pistas, topología de rutéo, etc. Estas reglas pueden ser aplicadas a diferentes objetos dentro de la tarjeta como son: redes o conjuntos de redes, vías, componentes, áreas, grupos formados por los anteriores, etc.

Para el caso de las tarjetas de la computadora de vuelo, se utilizó una plataforma sirvió como base para las 4 diferentes tarjetas impresas de la CV, en la figura 6.7 se muestra la plataforma base para las tarjetas impresas.

6.7 Cargado de la red del esquemático a la plataforma de las tarjetas impresas

Esta etapa consiste en ligar el listado de redes del esquemático con el diseño de la tarjeta impresa, al ejecutar este paso, el software automáticamente carga las huellas de los componentes tal y como fueron definidas en el esquemático, además se genera la vista del nido de ratas ("RAT NEST") de la tarjeta impresa, esta vista muestra las conexiones de todos los componentes del circuito impreso mediante el uso de líneas rectas.

6.8 Posicionamiento de componentes en las tarjetas

Antes de comenzar a rutear el circuito impreso, es necesario efectuar el posicionamiento de los componentes en la tarjeta, ya que al efectuar el cargado de la red, los componentes son colocados de forma aleatoria. Para colocar un ruteado eficiente de la tarjeta, es muy importante una buena colocación de los componentes, una buena práctica para lograr lo anterior es colocar los componentes de tal forma que las líneas del nido de ratas se crucen lo menos posible. Aunado a lo anterior se recomienda seguir las siguientes pautas para la colocación de los componentes:

- Colocar los componentes lógicos que trabajen con señales de mayor frecuencia, cerca del centro de la tarjeta.
- Reunir los componentes en bloques que efectúen funciones semejantes.
- Colocar los componentes analógicos juntos y de ser posible en la periferia de la tarjeta
- Colocar los capacitores de desacoplo de polarización, lo más cercano posible a los circuitos integrados de mayor frecuencia.
- No colocar componentes de alta frecuencia muy cercanos a los bordes de la tarjeta impresa.

En esta etapa, Protel ofrece una herramienta de posicionamiento automático de componentes llamada "Autoplacer". Para el caso de la computadora de vuelo, el posicionamiento de los componentes fue realizado mediante una combinación de posicionamiento automático y manual. En el diseño de la computadora de vuelo, los componentes que trabajan a mayor frecuencia son, el cristal del microcontrolador, el microcontrolador, los circuitos de memoria, el EDAC, y la lógica de decodificación de la memoria; todos estos componentes se encuentran localizados en la tarjeta del microprocesador (TCV). A continuación en las figuras 6.6 a 6.13 se muestran las salidas de nido de ratas generadas y listas para rutear, de las tarjetas impresas de la computadora de vuelo, en sus vistas superior (Top) e inferior (Bottom).

Figura 6.6: Vista Superior del nido de ratas de la tarjeta TCV.

Figura 6.7: Vista Inferior del nido de ratas de la tarjeta TCV.

Figura 6.8: Vista Superior del nido de ratas de la tarjeta TCVCTRL.

Figura 6.9: Vista Inferior del nido de ratas de la tarjeta TCVCTRL.

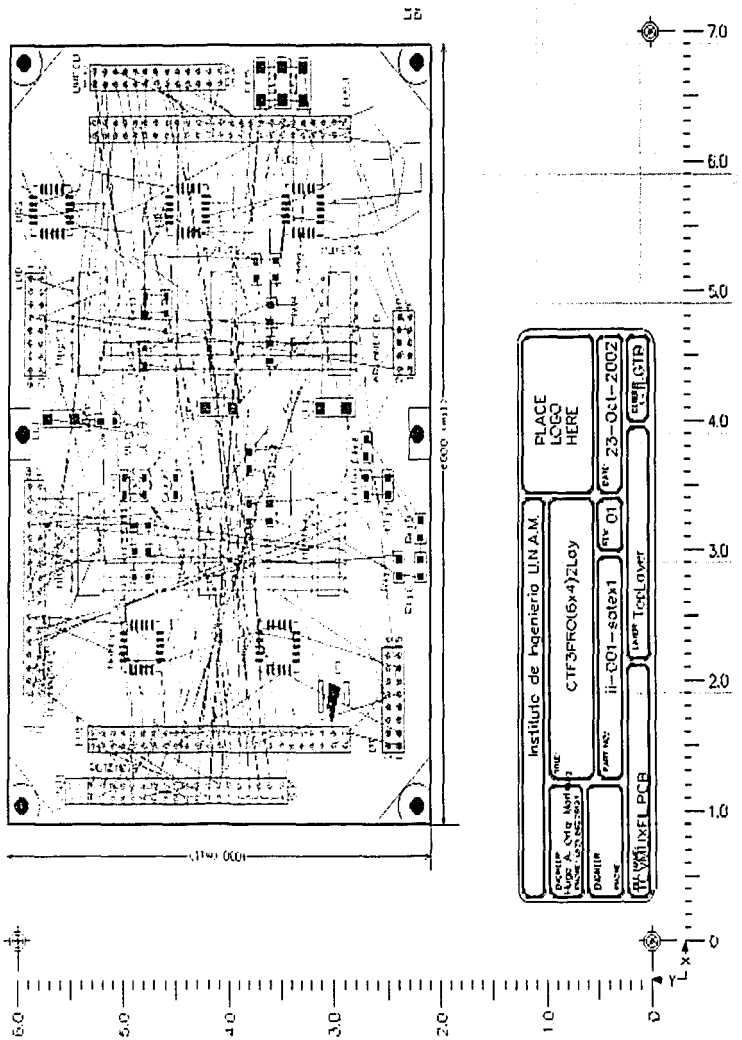


Figura 6.10: Vista Superior del nido de ratas de la tarjeta TCVMUXFILT.

Figura 6.11: Vista Inferior del nido de ratas de la tarjeta TCVMUXFILT.

Figura 6.12: Vista Superior del nido de ratas de la tarjeta TCVLATCH.

Figura 6.13: Vista Inferior del nido de ratas de la tarjeta TCVLATCH.

6.9 *Rutéo de las tarjetas impresas*

El rutéo, consiste en trazar las conexiones físicas de los componentes dentro de la tarjeta mediante el uso de pistas y vías. Regularmente, está etapa resulta muy complicada en función de la cantidad de componentes y de conexiones que contenga una tarjeta. Por tanto, esta etapa consume más tiempo en el diseño de las tarjetas impresas, sin embargo, Protel presenta una gran ayuda en este respecto, ya que su herramienta de rutéo automático ofrece muy buenos resultados en cuanto a calidad y completitud del mismo (más del 90% o 100% en todos los casos analizados).

La herramienta de rutéo Automático que emplea Protel está basada en el uso de algoritmos de rutéo por forma (Shape Based), que han demostrado ser más efectivos que los basados en laberinto (Maze Based).

6.9.1 Tarjeta de procesador

Esta tarjeta, resultó ser la más compleja para el rutéo (aproximadamente .83 in^2/componente de 14pin), debido al número de componentes y a la cantidad de conexiones, esto dio la pauta para decidir que está tarjeta fuera realizada en 3 capas, siendo necesario fabricarla en los EE.UU., ya que en México, ninguna compañía manufactura circuitos impresos con 3 o más capas en cantidades pequeñas. El método de rutéo empleado fue una combinación de rutéo automático y manual. Algunas de las reglas de diseño más importantes que se aplicaron durante el rutéo automático fueron los indicados en la tabla 6.1:

Tabla 6.1: Reglas de rutéo para la tarjeta del microprocesador.

Regla	Aplicada a	Propiedades
Regla de ancho de pistas	Redes de polarización	Min. = 10, Max. = 20, Preferido = 20
	Todas las demás redes	Min. = 10, Max. = 12, Preferido = 12
Capas de rutéo	Toda la tarjeta	Top = Horiz., Mid1 = Cualquier ángulo Bottom = Vert.
Separación entre objetos	Pads de pines de EDAC y MCU	5mils
	Entre Pads de montaje superficial y vías	40mils
	Entre polígonos de diferentes redes	30mils
	Toda la tarjeta	10mils
Prioridad de rutéo (entre más alto valor, tiene mayor prioridad)	Redes de polarización	100
	Bus de Direcciones (A1–A22)	90
	Bus de Datos del MCU	80
	Bus de Datos del EDAC	60
	Toda la tarjeta	0
Estilo de vías de rutéo	Toda la tarjeta	Ø interno = 20mil Ø externo = 40 mil

El resultado de rutéo arrojo un nivel de completitud de aproximadamente 98%, las líneas restantes fueron ruteadas de forma manual, además se agregaron polígonos de tierra y VCC, y se corrigieron algunas pistas previamente ruteadas por Protel.

En las figuras 6.14 a 6.16 se muestran las capas superior, media-1 e inferior completamente ruteadas.

6.9.2 Tarjeta de control y multiplexaje de procesadores

Esta tarjeta resultó ser la segunda en dificultad para el rutéo (1.35in^2/componente de 14 pines), debido a que contiene una cantidad mayor de componentes de montaje no superficial, además del número de conexiones que maneja (219 redes). En la tabla 6.2 se muestran las reglas de diseñó más importantes para el rutéo de la tarjeta, y en las figuras 6.17 y 6.18 se muestran las capas superior e inferior completamente ruteadas.

Tabla 6.2: Reglas de rutéo para TCVCTRL

Regla	Aplicada a	Propiedades
Regla de ancho de pistas	Redes de polarización	Min. = 10, Max. = 20, Preferido = 20
	Todas las demás redes	Min. = 10, Max. = 12, Preferido = 12
Capas de rutéo	Toda la tarjeta	Top = Horiz., Bottom = Vert.
Separación entre objetos	Entre Pads de montaje superficial y vías	30mils
	Entre polígonos de diferentes redes	30mils
	Toda la tarjeta	10mils
Topología de rutéo	Toda la tarjeta	Horizontal
Estilo de vías de rutéo	Toda la tarjeta	Ø interno = 20mil Ø externo = 40 mil

6.9.3 Tarjeta de multiplexaje y filtrado de sensores

La densidad de componentes de esta tarjeta es aproximadamente de 1.17 in^2/Componente de 14 pines, con un número de redes de 199. El ruteado de esta tarjeta resulto relativamente sencillo, debido a que no contiene un gran número de componentes de montaje no superficial. En las figuras 6.19 y 6.20 se muestran las capas superior e inferior totalmente ruteadas, las reglas que se siguieron para el rutéo de componentes fueron las mismas que para la tarjeta de control y multiplexaje de procesadores TCVCTRL (tabla 6.2)

6.9.4 Tarjeta de eliminación de efecto Latch-up

Esta tarjeta tiene una densidad de componentes aproximada de $1.35in^2$/Componente de 14 pines, y un número de redes de 207, sin embargo, la mayoría de los componentes son de montaje superficial, por lo que resulto la tarjeta más sencilla en cuanto a rutéo. Las reglas aplicadas para el rutéo de ésta tarjeta, fueron las mismas que para las tarjetas TCVMUXFIL y TCVCTRL (tabla 6.2), en las figuras 6.21 y 6.22 se muestran las capas superior e inferior totalmente ruteadas.

Figura 6.14: Capa Superior (Top) de la tarjeta TCV completamente ruteada.

Figura 6.15: Capa Media-1 (Mid1) de la tarjeta TCV completamente ruteada.

Figura 6.16: Capa Inferior (Bottom) de la tarjeta TCV completamente ruteada.

Figura 6.17: Capa Superior (Top) de la tarjeta TCVCTRL completamente ruteada.

Figura 6.18: Capa Inferior (Bottom) de la tarjeta TCVCTRL completamente ruteada

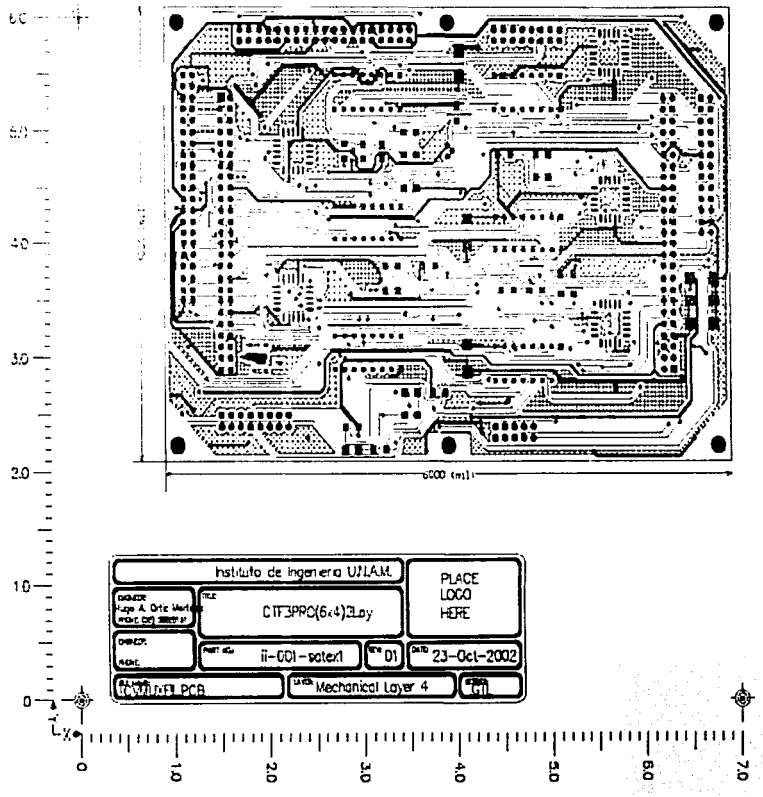


Figura 6.19: Capa Superior (Top) de la tarjeta TCVMUXFILT completamente ruteada.

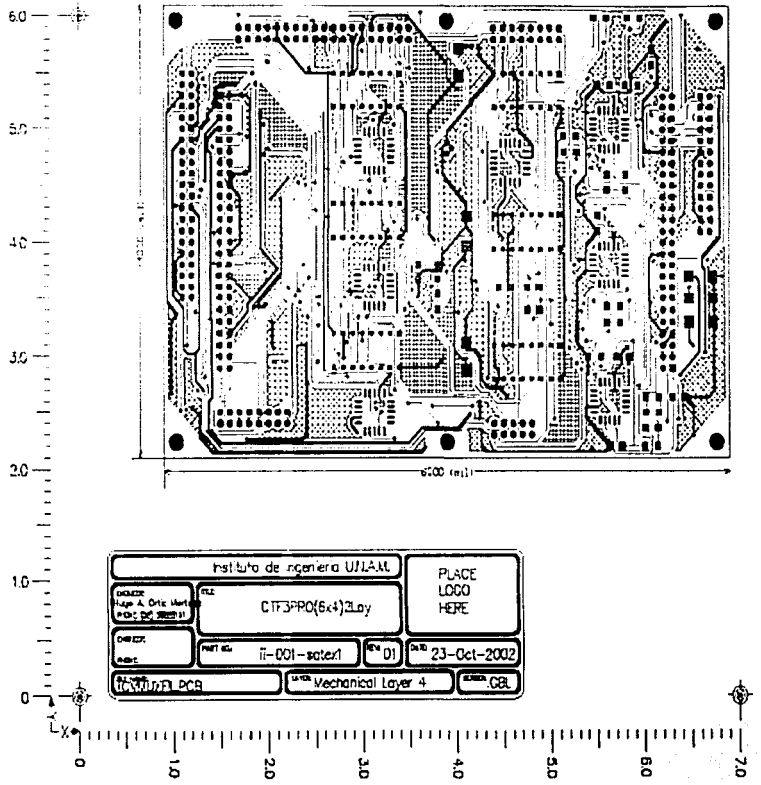


Figura 6.20: Capa Inferior (Bottom) de la tarjeta TCVMUXILT completamente ruteada.

Figura 6.21: Capa Superior (Top) de la tarjeta TCVLATCH completamente ruteada.

Figura 6.22: Capa Inferior (Bottom) de la tarjeta TCVLATCH1 completamente ruteada.

6.10 Verificación final de las reglas de diseño e integridad de señal de las tarjetas

En esta etapa se realiza un chequeo de todas las reglas aplicadas durante el diseño de la tarjeta, se verifican entre otros aspectos: ancho mínimo de pistas, dimensiones de vías, posicionamiento de componentes, se efectúan las anotaciones finales en la tarjeta, etc. Todo esto de forma previa a la manufactura. Para realizar esto basta con indicar a Protel que realice un chequeo de reglas de diseño "DRC" (Design Rule Check), al realizar este chequeo Protel indica en la pantalla, sobre la misma tarjeta, el lugar de las violaciones y su tipo, además genera un reporte a manera de listado de las violaciones de las reglas de diseño.

Un aspecto importante que debe ser tomado en cuenta durante el diseño de los circuitos impresos que trabajen con lógica de alta velocidad, es la integridad de las señales del circuito, es decir, considerar efectos como el Cross-Talk (interferencia), reflexiones de señal, retrasos de tiempo debidos a las pistas, etc. En el caso de la computadora de vuelo, la tarjeta que presenta la mayor susceptibilidad a estos efectos, es la tarjeta del procesador, principalmente en la línea de oscilación del cristal (Xtal1) cuya frecuencia de oscilación es de 40Mhz. En lo que respecta a las demás líneas de la tarjeta no es necesario llevar a cabo esta análisis, ya que trabajan a velocidades inferiores a los 10MHz (Esta frecuencia fue obtenida de las líneas del bus dirección, las cuales trabajan a la más alta frecuencia después de la línea Xtal1). Por lo anterior se decidió efectuar un análisis de la integridad de las señales que viajan a través de la red Xtal1, y de las redes que se encuentren acopladas a esta, es decir, que corran paralelas y que por esto sean mas susceptibles de presentar el efecto de Cross-Talk. Debido a que el rutéo de la línea Xtal1, se mantuvo lo más corto posible además de que fue rodeado de líneas de tierra con el propósito de minimizar el efecto de Cross-Talk que pudiese provocar, la herramienta de análisis de Protel, no encontró ninguna línea acoplada. El criterio aplicado toma en cuenta que para considerar una línea acoplada a Xtal1 su trazado deberá ser paralelo al de Xtal1 por lo menos en una distancia de 1mm , y no deberá existir una separación mayor a los 100mm entre segmentos paralelos [PROTEL,1999].

Para el análisis de integridad de señal, Protel presenta una herramienta capaz de efectuar análisis de Cross-Talk, reflexión, Formas de onda, etc. (figura 6.23). Con la ayuda de esta herramienta se procedió a efectuar el análisis de reflexión de onda en la línea Xtal1, el análisis de reflexión de onda arrojó el resultado mostrado en la figura 6.23.

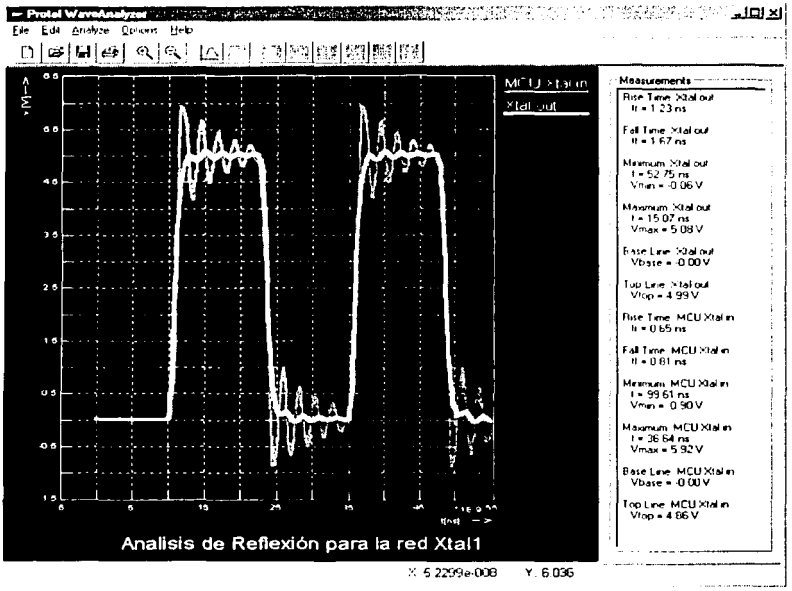


Figura 6.23: Gráfica de resultados de la reflexión de señal para Xtal1.

En la figura 6.23 se observan las características de las señales presentes en el cristal y en el alfiler de entrada del microcontrolador, si tomamos la tabla 6.3, en donde se muestran las características necesarias para la señal Xtal1 [SIEMENS, 1997], se puede observar que no existe ningún problema en cuanto a la forma de onda de la señal Xtal1 que fue arrojada por el analizador de integridad de señal de Protel.

Tabla 6.3: Características de la señal Xtal1 [SIEMENS, 1997].

Parámetro	Símbolo	Valores Límite		Unidades
		Mín.	Máx.	
Voltaje de entrada Alta	V_{IH}	0.7Vcc	Vcc+0.5	Volts
Voltaje de entrada Baja	V_{IL}	-0.5	0.2Vcc-0.1	Volts

Además del análisis de reflexión de señal efectuado en el Xtal1, se procedió a realizar un análisis general de la integridad de señal en todas las pistas de la tarjeta, este análisis realizado por Protel recibe el nombre de "Screening" y muestra aspectos como:

voltaje de rizo, voltajes pico positivo y negativo, tiempos de decaimiento y levantamiento de señal entre otros. Mediante el "Screening" se observó el comportamiento general de toda la tarjeta al aplicar una señal con una forma de onda predeterminada (forma cuadrada de 10Mhz), a partir de la tabla de resultados se pueden obtener pautas para el análisis más a fondo de las líneas que presenten problemas.

En la figura 6.24 se muestra un fragmento de la tabla de resultados obtenida del "Screening" de la tarjeta del procesador, de esta tabla de datos se observó que ninguna de las líneas presenta problemas en cuanto a voltajes de rizo, tiempos de retardo, etc.

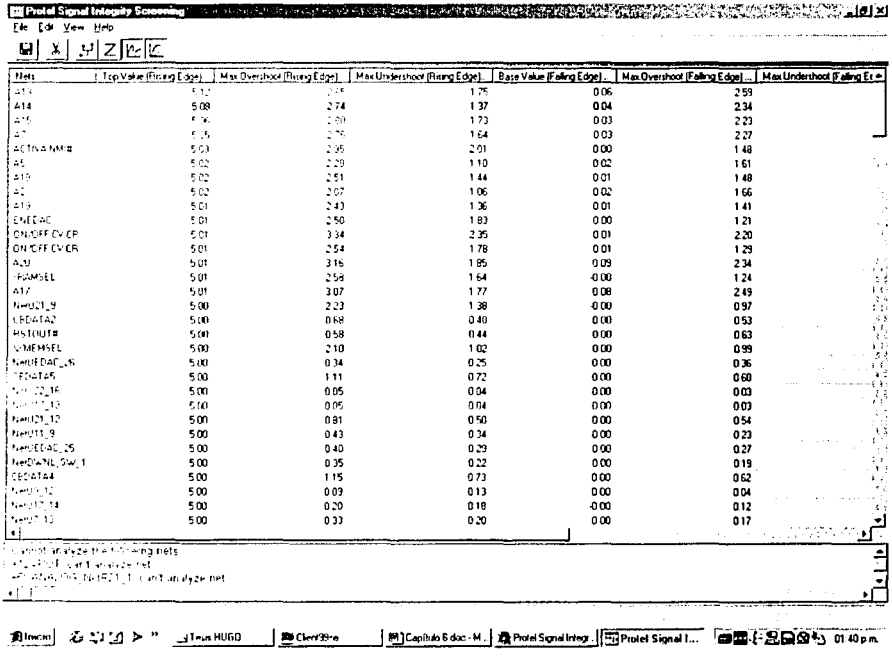


Figura 6.24: Resultado del "Screening" para la tarjeta del procesador.

6.11 Generación de archivos de salida para la manufactura de las tarjetas impresas

La generación de archivos de salida para la manufactura consiste en generar archivos de un cierto formato, determinado por el fabricante de las tarjetas impresas, éste archivo contiene la información suficiente para llevar fabricar de la tarjeta impresa. El formato elegido para la generación de archivos de salida, depende principalmente del método empleado en la fabricación de los circuitos impresos.

En el caso de los fabricantes contratados solicitaran la información que se describe a continuación:

6.11.1 Tarjetas TCVCTRL, TCVMUXFILT y TCVLATCH

Para los circuitos impresos de las tarjetas TCVCTRL, TCVMUXFILT y TCVLATCH, que son de dos caras, se optó por un fabricante en México. El proceso de manufactura empleado por este fabricante, es un proceso manual de fotolitografía, para esto es necesario tener una impresión en negativo de las diferentes caras del circuito impreso, estas impresiones fueron obtenidas por el fabricante a partir de archivos tipo Acobat generados por Protel. Las caras generadas para cada uno de los circuitos impresos fueron:

1. Cara superior de cobre (Top Layer).
2. Cara inferior de cobre (Bottom Layer).
3. Mascara de antisoldado superior (Top Solder Mask).
4. Mascara de antisoldado inferior (Bottom Solder Mask).
5. Película superior de seda (Top Silkscreen).
6. Película inferior de seda (Bottom Silkscreen).
7. Capa Mecánica (Mechanical Layer).
8. Localización de perforaciones (Drill Drawing).
9. Guía de tamaños de perforación (Drill Guide).

6.11.2 Tarjeta TCV

Como fue mencionado anteriormente, esta tarjeta fue fabricada en USA, el proceso empleado por este fabricante, es un proceso de control numérico y de fotolitografía automática, éste tipo de proceso presenta una calidad superior en cuanto al terminado de la tarjeta impresa, además una mayor precisión, con un considerable aumento del costo. Para este proceso de fabricación, el fabricante requirió de la generación de archivos "GERBER" de la tarjeta TCV. Los archivos GERBER son un conjunto de archivos en los cuales se encuentra la información necesaria para el proceso de cada una de las capas que forman el circuito impreso. Para la manufactura de la tarjeta TCV fue necesario generar los archivos GERBER para las mismas capas que

para las otras tarjetas de la CV además de generar un archivo para la capa Intermedia de cobre (Mid Layer).

En el siguiente capítulo se describen las pruebas realizadas a la las diferentes tarjetas de la CV y su proceso de integración.

Capítulo 7. Integración y pruebas del hardware de la CV

7.1 Introducción

En los capítulos anteriores se describieron las modificaciones correspondientes a la computadora de vuelo con la finalidad de proveerla con las capacidades señaladas como objetivos de la presente tesis [Apartado 1.7], también se ha descrito el proceso de diseño y desarrollo de las tarjetas electrónicas para la computadora de vuelo, que incorporan los cambios propuestos y que forman la versión final de la computadora de vuelo.

Una vez fabricados los impresos de la computadora de vuelo, se procedió a efectuar el armado de los mismos y a la integración de la computadora de vuelo. De forma casi paralela se realizaron las pruebas de validación del hardware de la CV

En este capítulo se trata el proceso de integración y pruebas del hardware de la computadora de vuelo, así también, se exponen las bases para la ejecución de pruebas operativas del hardware de la CV, durante las pruebas de vibración y termo vacío para calificar el equipo para vuelo espacial y durante las pruebas en el sitio de lanzamiento.

Por último, en el capítulo siguiente se dan las conclusiones y recomendaciones que a juicio del autor, son importantes para el mejoramiento de la computadora de vuelo y que emanan de la realización de esta tesis.

7.2 Ensamblado de las tarjetas impresas

Como se ha mencionado, la computadora d vuelo en su versión final, estará formada por 6 tarjetas electrónicas: TCVLATCH, TCVMUXFILT, TCVCTRL y tres tarjetas de procesador TCV. Debido a problemas con los proveedores de componentes electrónicos, en este momento solamente se cuenta con componentes suficientes para construir una versión de la computadora de vuelo con un solo procesador, sin embargo esta versión de la computadora podrá ser utilizada para la validación del hardware, y la única tarjeta de procesador podrá ser configurada para su trabajo como cualquiera de los procesadores.

Otro aspecto que deberá tomarse en cuenta, es que el dispositivo EDAC que contendrán las tarjetas de procesador, no esta disponible al momento de escribir de esta tesis, sin embargo, se propondrán pruebas de validación para cuando este ya se encuentre ensamblado en las tarjetas.

Una vez fabricados los circuitos impresos de la computadora de vuelo se llevó a cabo una verificación ocular y de dimensiones de las tarjetas impresas de forma previa al montaje de los componentes, con la finalidad de detectar defectos en el proceso de fabricación de las mismas, también de manera previa al soldado de los componentes, se

llevó a cabo una verificación de la continuidad entre las pistas de los impresos con la finalidad de detectar posibles cortocircuitos o fallas en el rutéo de las tarjetas.

Posteriormente se procedió a soldar los componentes de cada una de las tarjetas, este proceso fue efectuado de forma modular, siguiendo el diagrama de la arquitectura de la CV presentado en el capítulo 2 figura 2.1, de tal forma que cada una de las tarjetas estuviese completamente armada antes de comenzar con la siguiente.

Durante el proceso de ensamblado de las tarjetas, fue de suma importancia tomar ciertas precauciones y recomendaciones, entre las que podemos señalar:

- Utilizar una pulsera de aterrizamiento eléctrico, durante la manipulación de las tarjetas impresas y/o los componentes.
- Proteger las tarjetas impresas y los componentes del polvo.
- No aplicar calor excesivo durante el soldado de los componentes.
- Verificar la posición correcta de los componentes antes del soldado.
- Verificar con lupa o cuenta hilos la correcta soldadura de cada uno de los componentes.
- Utilizar en cantidades moderadas el líquido o pasta para soldar.
- Verificar constantemente el estado de la punta de cautín, ya que es una herramienta sumamente importante durante el proceso de soldado.

Además, se efectúo un proceso de limpieza profunda (Con thinner, químicos con base de freón y aire a alta presión) y de revisión bajo un microscopio estereoscópico, con equipo e instalaciones prestadas por el Departamento de Ingeniería Ambiental de la División de Estudios de Postgrado de la Facultad de Ingeniería.

En las figuras 7.1 y 7.2 se muestran fotografías de las tarjetas durante su limpieza

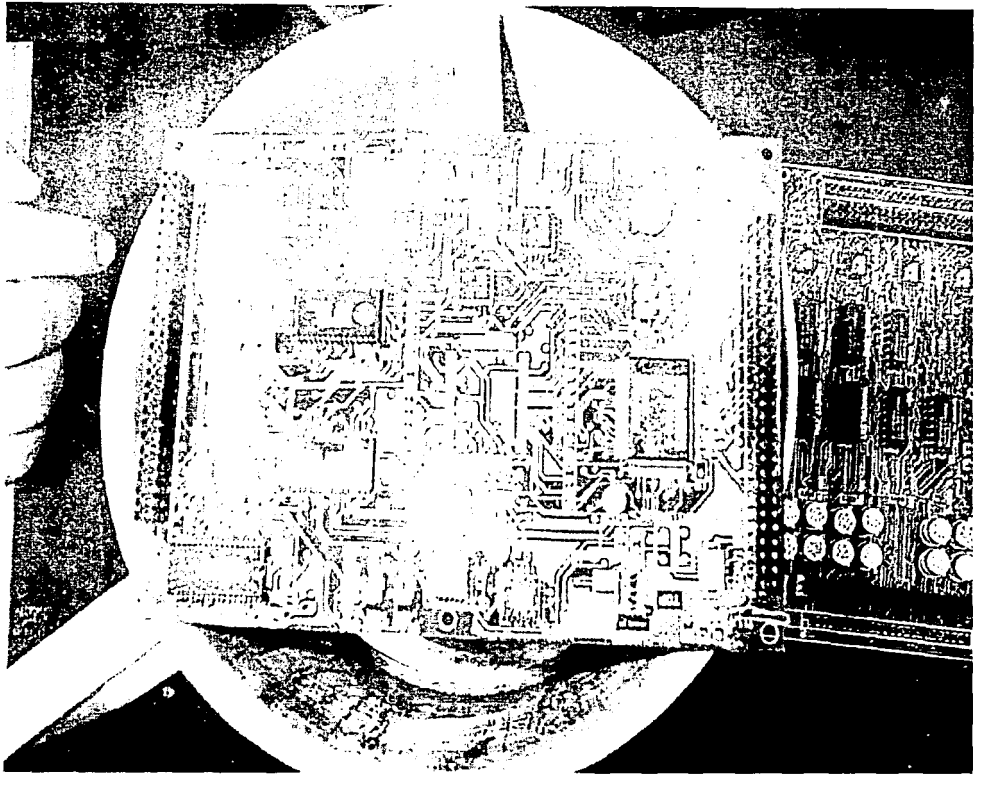


Figura 7.1: Fotografía tomada durante la limpieza de las tarjetas TCV y TCVCTRL

Figura 7.2: Fotografía de tarjetas TCLATCH y TCVMUXFILT durante la limpieza.

7.3 Pruebas realizadas a la tarjeta TCVLATCH

Esta tarjeta contiene los conectores de entrada/salida de la computadora de vuelo, el hardware de eliminación de efecto Latch-up, y parte del hardware de comunicaciones de red interna (Principal y redundante).

Una ves ensamblada la tarjeta se midió la continuidad entre +5Vpot, GND, +12Vpot y -12Vpot, con la finalidad de detectar posibles cortocircuitos. Después se procedió a efectuar la verificación del hardware de la tarjeta de forma independiente a las otras, es decir polarizando únicamente a TCVLATCH, y sin conectarle ninguna de las tarjetas que forman la CV

La operación correcta del circuito de eliminación de efecto Latch-up fue verificada mediante la generación de la señales: **ON/OFF_CV_CP** y **ON/OFF_CV_CR,** que indican mediante una transición de 0 a 5Volts la presencia del efecto Latch-up y la verificación de las señales: **ON_CR_DT#** y **ON#/OFF CP,** que indican el encendido de CR ó CP al hardware de control y encendido de procesadores, las cuales deben ser inhibidas durante un periodo de aproximadamente 14 segundos a partir de la detección, presentando un estado alto.

La verificación del hardware restante contenido en la tarjeta fue verificada hasta el momento de la integración, debido a que dicho hardware formaba parte de un módulo distribuido en varias tarjetas y era necesario contar con la contraparte del hardware para la realización de pruebas.

7.4 Pruebas realizadas a la tarjeta TCVMUXFILT

Esta tarjeta contiene el hardware de multiplexaje y filtrado de sensores, hardware de acondicionamiento para sensores de temperatura y parte del hardware de red interna.

De igual forma que con al tarjeta TCVLATCH, se procedió a medir la continuidad entre las líneas de polarización que llegan a esta tarjeta para después efectuar la verificación del hardware de ésta tarjeta solamente.

Es importante mencionar que el presente trabajo no pretende efectuar un análisis del diseño de los sensores y su hardware de acondicionamiento, ni obtener las curvas de calibración de los mismos[17], sino que pretende proponer un esquema de pruebas suficientes para la verificación de su funcionamiento.

Para la verificación del hardware de multiplexaje y filtrado de señales de sensores, se generaron las señales de control de los multiplexores: **A_MUXSENS, B_MUXSENS** y **C_MUXSENS**, se alimentaron las entradas de los multiplexores

[17] Un análisis mas profundo del diseño de los sensores y su hardware de acondicionamiento se puede encontrar en [Mejía, 2001], además lo referente a la calibración de sensores es objeto de un trabajo de tesis en desarrollo en el IIUNAM.

asignados a señales que no son de temperatura con un generador de señales y se observó la salida de cada uno de los multiplexores asociados, presentando la misma señal que la de entrada, pero limitada a voltajes de 0 a 5Volts.

Los sensores de temperatura a bordo de SATEX, utilizan el circuito integrado LM135H [NATIONAL SEMICONDUCTORS, 2000], al cual se le conectó un amplificador seguidor de voltaje, con la finalidad de acoplar impedancias. La señal que entrega el LM135H, es una señal de voltaje directamente proporcional a la temperatura absoluta, con una constante de 10mV/K, su rango de operación es de -55ºC a 150ºC, la electrónica de acondicionamiento para estas señales está compuesta por un amplificador restador cuyo diagrama se muestra en la figura 7.3.

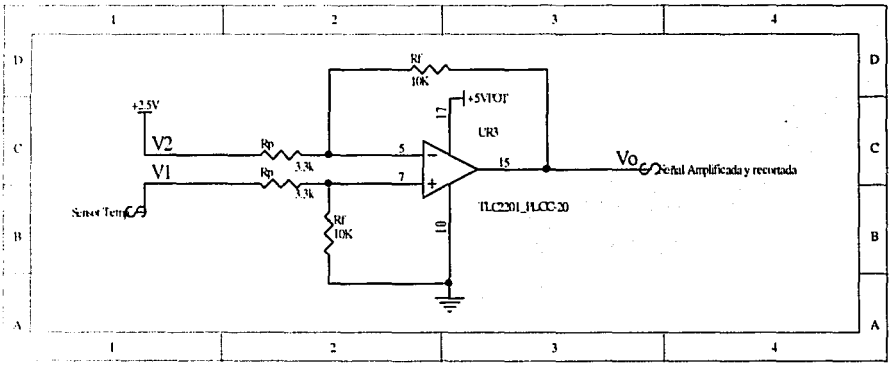


Figura 7.3: Amplificador restador de la etapa de acondicionamiento de sensores de temperatura.

La ecuación que muestra el voltaje de salida en función del voltaje de entrada de este circuito es [COUGHLIN & DRISCOLL, 1993]:

$$Vo = (V_1 - V_2) \cdot \frac{Rf}{Rp}$$

Donde:

Vo: Voltaje de salida del restador.

V_2: Voltaje de entrada al restador, referencia de voltaje fija.

V_1: Voltaje de entrada al restador, entregado por el sensor de temperatura

$\frac{Rf}{Rp}$: Ganancia del amplificador

Ahora, considerando la ecuación que relaciona la temperatura del sensor, con su voltaje de salida:

$$V_t = (T + 273.15) \cdot K_t$$

Donde:

V_t: Voltaje de salida del sensor de temperatura.
T: Temperatura en °C.
K_t: Constante de conversión del sensor = $.01 \frac{Volts}{^{o}K}$

Sustituyendo
Rf = 10KΩ
Rp = 3.3KΩ
V_2 = 2.5v
V_1= $(T - 273.15) \cdot 0.01$

Se obtiene que el voltaje de salida del restador en función de la temperatura del sensor es:

$$Vo = (((T + 273.15) \cdot 0.01) - 2.5) \cdot 3$$

Las pruebas del hardware de acondicionamiento de sensores fueron realizadas para verificar el cumplimiento de la ecuación anterior utilizando un sensor de temperatura conectado a las entradas de los multiplexores de temperatura, este sensor fue colocado en un recipiente con agua, al cual le se hizo variar la temperatura y se obtuvieron mediciones para diferentes valores de temperatura y voltaje de salida.

Para verificar de la etapa de filtrado de los sensores, se generaron las señales de entrada para cada uno de los sensores; para los sensores que no son de temperatura se generó una onda senoidal de 0 a 5volts, y para los sensores de corriente una onda senoidal de 2.5 a 5 Volts. La frecuencia de estas ondas fue variada sistemáticamente, observando al mismo tiempo la señal de salida del filtro. De la relación de la señal de entrada con la de salida, se obtuvo una gráfica similar a la mostrada en la figura 7.4.

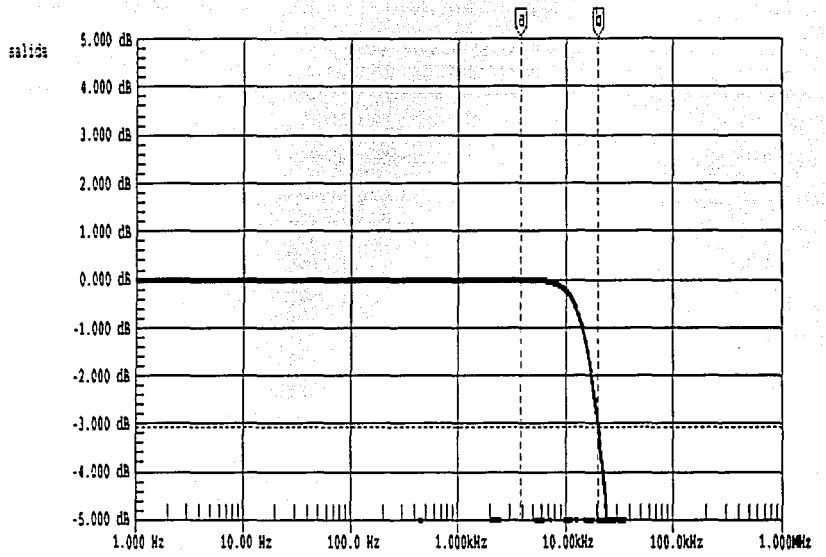


Figura 7.4: Grafica Ganancia vs. Frecuencia del filtro de sensores.

7.5 *Pruebas realizadas a la tarjeta TCVCTRL*

Esta tarjeta contiene el hardware de encendido de procesadores, hardware de encendido de electrónica de sensores, hardware de comunicaciones externas y hardware de multiplexaje del bus de instrumentación [Apartado 2.2].

Al igual que con las tarjetas anteriores, primero se efectuó la medición de continuidad entre las líneas de polarización, para seguir con la evaluación del hardware de la tarjeta.

Una de las partes más importantes del la computadora de vuelo, es el hardware de encendido de procesadores, ya que una falla en este podría ocasionar la falla total de la computadora de vuelo. La validación operativa de este hardware se efectuó de manera conjunta con la validación del hardware de multiplexaje del bus de instrumentación, generando las líneas **ON_CR_DT#**, **ON#/OFF_CP** y **SELCR0-1** de acuerdo a la tabla 7.1. Para corroborar la polarización de cada una de las tarjetas de procesador mediante las líneas **GND_CP**, **GND_CR0** y **GND_CR1**, el hardware de multiplexaje del bus de instrumentación se corroboró aplicando una señal en las

entradas de los multiplexores y verificando las salidas en el bus correcto. Un aspecto importante de la prueba del hardware de encendido fue el verificar que bajo ninguna circunstancia fuesen encendidas las tarjetas CP y CR1 al mismo tiempo, ya que estas comparten el mismo conector del bus de instrumentación y podrían surgir cortocircuitos si esto sucede.

Tabla 7.1: Tabla funcional del sistema de encendido de procesadores y selección de bus.

Modo de operación	Líneas de control			Procesadores encendidos			Bus de instrumentación conectado a
	ON#/OFF_CP	ONCRDTL	SELCR0-1	CP	CR0	CR1	
P. P.	0	0	0	ON	ON	OFF	CR0
C.P.	0	0	1	ON	OFF	OFF	CP y CR1
C.P.	0	1	0	ON	OFF	OFF	CP y CR1
C.P.	0	1	1	ON	OFF	OFF	CP y CR1
CR0	1	0	0	OFF	ON	OFF	CR0
CR1	1	0	1	OFF	OFF	ON	CP y CR1
EF. LUP.	1	1	0	OFF	OFF	OFF	CP y CR1
EF. LUP.	1	1	1	OFF	OFF	OFF	CP y CR1
Descripción del los modos de operación							
P.P.	Modo de procesamiento paralelo, CR0 actúa como procesador maestro, él posee el bus de instrumentación						
C.P.	Modo de operación del procesador principal, con este modo inicia operaciones en vuelo la CV						
CR0	Modo de operación del procesador redundante 0 (solo CR0)						
CR1	Modo de operación del procesador redundante 1 (solo CR1)						
EF. LUP.	Modo de operación con efecto "Latch-up", el modulo de detección y eliminación de efecto "Latch-up", tiene prioridad sobre el control externo de DT						

El hardware de comunicaciones externas, fue validado mediante la conexión de el puerto serie de una computadora personal a las líneas de entrada de cada uno de los canales de comunicaciones externas, luego se procedió a unir las líneas **RX_EXT_CV** con **TX1_EXTERNA,** con esto se logró que los datos transmitidos por la PC. fueran recibidos por ella misma en forma de eco.

7.6 *Pruebas realizadas a la tarjeta TCV.*

Esta tarjeta contiene al microprocesador y su memoria, hardware de detección y corrección de errores, sensor de detección de efecto Latch-up, hardware de activación de BTL, figura 3.8, y el sensor de temperatura del microprocesador.

La primera prueba de esta tarjeta, consistió en la medición de continuidad entre las líneas de polarización, después de esto se polarizó la tarjeta y se probaron los voltajes en las líneas de polarización de los componentes.

Las pruebas de polarización se efectuaron para la tarjeta en cada una de sus configuraciones, como CP, como CR0 y como CR1. la configuración de la tarjeta de procesador, se realizó mediante la colocación de puentes (jumpers) en la tarjeta, la tabla 7.2 muestra la colocación de estos puentes para cada configuración.

Tabla 7.2: Tabla de configuración de la tarjeta TCV.

Colocación de puentes para configurar a la tarjeta TCV							
Modo	Puentes indicadores de modo para MCU			Puentes de señal de efecto Latch-up		Puentes de polarización	
	J10	J5	J4	JP9ONCP	JP9ONCR	J11A	J11B
CP	OFF	ON	OFF	ON	OFF	ON	OFF
CR0	ON	OFF	OFF	OFF	ON	ON	OFF
CR1	OFF	OFF	ON	OFF	ON	OFF	ON

Además de la configuración correcta de los pines, es importante notar que cada una de las tarjetas se polariza con diferentes líneas de tierra, quedando: **GND-CP** para las tarjetas CP y CR0, y **GND-CR1** para la tarjeta CR1.

Una vez probada la polarización se efectúo la prueba del sensor de temperatura del procesador, ajustando un sensor digital de temperatura al encapsulado del circuito LM135H de la tarjeta y verificando la lectura entregada por ambos sensores midiendo con un multímetro digital el voltaje de salida del sensor de la tarjeta en la línea **TERA_CP.**

La validación del sensor de detección de efecto Latch-up fue realizada de manera previa al montaje del microprocesador, esta se realizó colocando un potenciómetro entre las pistas de polarización del microcontrolador y variando su resistencia desde varios KΩ hasta menos de 10Ω y al mismo tiempo midiendo la corriente detectada por el sensor, el cual debe de dispararse al detectar una corriente mayor a los 250mA. El disparo del sensor de efecto Latch-up se verifica en las líneas **ON/OFF_CV_CP** o **ON/OFF_CV_CR**, dependiendo si la tarjeta se encuentra configurada como principal o como redundante respectivamente. Al finalizar la prueba del sensor de efecto Latch-up se procedió a montar el microcontrolador y efectuar de nuevo las pruebas de polarización.

El siguiente paso en el proceso de pruebas de la tarjeta del microcontrolador, fue realizado con de una computadora personal conectada al puerto serial del microcontrolador, haciendo las veces de estación terrena, la prueba consistió en cargar un programa de validación que ejecutó rutinas de diagnostico del procesador y sus periféricos así como de su memoria expandida. Entre estas rutinas se encuentran el algoritmo tratado en el capitulo 5, el que fue cargado a la tarjeta del microcontrolador. Con la ejecución de este programa se valido la operación de:

- Microcontrolador.
- Memoria Externa.
- Memoria Expandida.
- Decodificación de Memoria.

- Hardware de activación de BTL.

En este punto se tiene una validación completa de la tarjeta de procesador sin el dispositivo EDAC, esta tarjeta de procesador sin el dispositivo puede ser enviada como parte de la computadora de vuelo sin que se vea afectado el comportamiento electrónico de sus componentes, la única peculiaridad de esta tarjeta es que no contaría con la capacidad de detección y corrección de errores en memoria RAM.

Para la validación del hardware de detección y corrección de errores en RAM, se propone utilizar, una de las muestras comerciales que se proveen junto con el dispositivo EDAC militar. Para estas pruebas se requeriría de una tarjeta impresa de procesador (TCV), a la cual le serán montados únicamente los componentes necesarios para el hardware de detección, como son: EDAC, compuertas lógicas, resistencias y capacitores. Una vez montado dicho hardware se procederá a generar las líneas de entrada al EDAC, según el diagrama de funcionamiento del mismo Capitulo y corroborando la generación correcta de las palabras de datos y chequeo en sus salidas correspondientes. Luego se procederá a efectuar el proceso a la forma inversa, es decir, generando la palabra de chequeo y datos, y verificando la palabra corregida y la generación de las indicaciones de error.

7.7 *Integración de la CV y pruebas finales de los módulos de hardware*

El ensamble y validación de cada una de las tarjetas fue realizado de forma modular, sin embargo, algunos de los módulos señalados en el capitulo 2, se encuentran distribuidos en más de una tarjeta de electrónica, por esto fue necesario efectuar las pruebas de estos módulos hasta el momento de la integración de la computadora de vuelo.

Para la etapa de integración de la computadora de vuelo y las pruebas en ésta, se formaron las mismas precauciones y recomendaciones para el manejo de la electrónica, que fueron mencionadas antes en este capítulo, mismas que deberán ser observadas siempre que se trabaje con equipo electrónico de la computadora de vuelo.

El primer paso en la integración de la computadora de vuelo fue efectuar la compatibilidad física de los conectores que van de una tarjeta a otra. El orden de ensamblado de las tarjetas, a partir de la tarjeta que contiene los conectores al exterior se muestra en la figura 7.5.

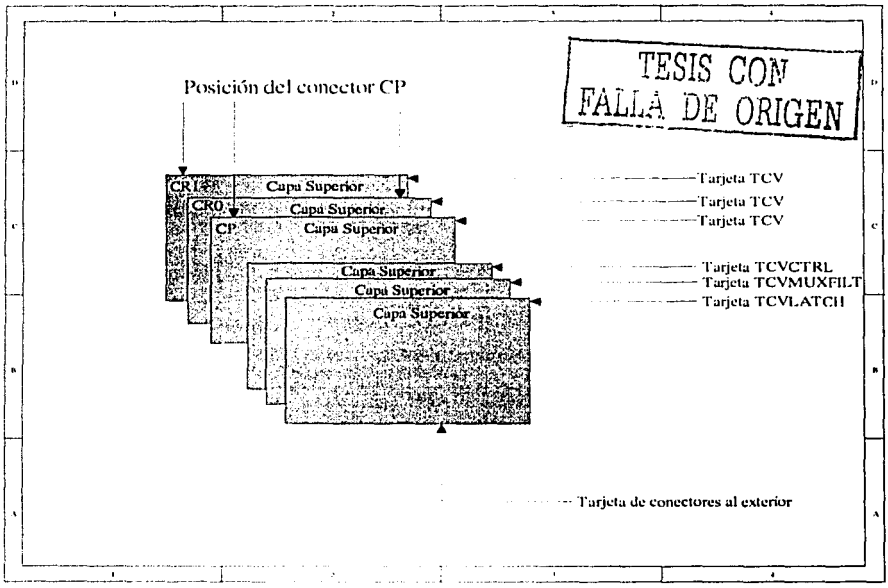


Figura 7.5: Orden de ensamblado de las tarjetas de la computadora de vuelo.

Durante el proceso de integración de la computadora de vuelo fue necesario el uso del simulador del satélite SIMSAT para la polarización, la generación de las señales de control, la simulación del hardware de bobinas de torque magnético, etc.

Después de la verificación física de los conectores, se efectuó una verificación de la concordancia de las señales de estos y se procedió a unir las tarjetas TCVLATCH con TCVMUXFILT, después, se efectuó una prueba de continuidad entre líneas de polarización. Una vez ensambladas las tarjetas antes mencionadas, se estuvo en posibilidades de verificar el funcionamiento de los canales de comunicaciones de red interna principal y redundante.

Para la verificación de la red interna, se procedió a conectar una PC, en cada uno de los nodos de red interna (principal y redundante) y a efectuar la transmisión de mensajes por medio de sus puertos seriales.

Una vez ensambladas y validadas las tarjetas TCVLATCH y TCVMUXFILT, se efectúo el ensamblado de la tarjeta TCVCTRL y se efectuaron pruebas de polarización. Por último y antes de seguir con el proceso de integración, se efectuó de nuevo la

verificación del hardware de encendido y multiplexaje de procesadores, esta vez generando las líneas de control mediante el SIMSAT.

Finalmente se ensambló la tarjeta TCV configurada como CP, en la posición correspondiente, y se efectuaron las pruebas de polarización. Una vez ensamblada la tarjeta TCV, se contaba con una versión de un solo procesador para la computadora de vuelo.

Como fue antes descrito, solamente se contaba con una tarjeta de procesador, sin embargo se efectuaron las pruebas que se mencionan a continuación para cada una las configuraciones de esta tarjeta: CP, CR0 y CR1.

La primera prueba realizada a la computadora de vuelo una vez ensamblada, consistió en cargar y ejecutar el programa de diagnóstico mencionado antes en este capítulo.

Una de las pruebas más contundentes realizadas a la computadora de vuelo, consiste en armar un esquema de simulación del satélite completo, este esquema de simulación utiliza las herramientas de simulación de satélite "SOFDEVO" [TORRES, 2002] y "SIMSAT", además utiliza el software de estación terrena.

El esquema de esta prueba en la figura 7.6 en forma de un diagrama de bloques que muestra la interacción de los componentes, así mismo, en las figuras 7.7 a 7.10 se presentan las fotografías correspondientes al proceso de pruebas con CV y a la computadora de vuelo ensamblada respectivamente.

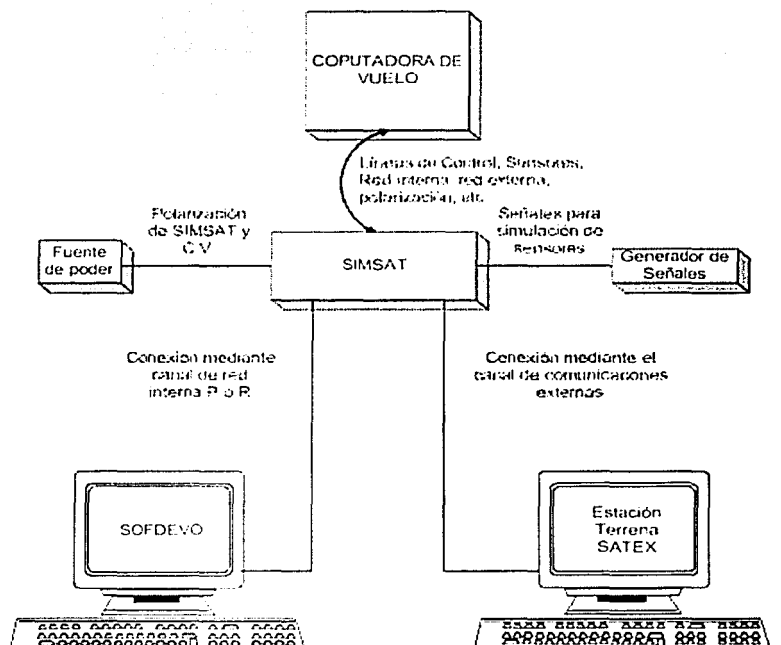


Figura 7.6: Esquema de pruebas de simulación completa del satélite.

Figura 7.7: Esquema de pruebas de simulación completa del satélite

Figura 7.8: SIMSAT conectado a CV.

Figura 7.9: ET conectada a CV.

Figura 7.10: Computadora de vuelo ensamblada.

A partir de la realización de esta prueba, se sugirió el uso de la computadora de vuelo en su versión final, para continuar con el desarrollo del software de operaciones de satélite y el software de estación terrena, para que de esta manera sirva como un proceso de validación continua de la misma.

Capítulo 8. Conclusiones y recomendaciones

8.1 Conclusiones

Del trabajo expuesto se obtuvieron las siguientes conclusiones:

1. Se cuenta con una versión final de la computadora de vuelo, validada y ensamblada con especificaciones de vuelo espacial para orbita baja (Aprox. 800 km). En esta versión todos los componentes con excepción del microcontrolador, tienen especificaciones militares. El microprocesador es de tecnología CMOS, con un rango de temperatura industrial extendido y se encuentra protegido contra el efecto Latch-up.

2. Se efectúo la revisión y el mejoramiento del hardware de la computadora de vuelo, mediante el cual se corrigieron algunos errores en el diseño previo, haciéndolo más confiable y tolerante a fallas.

3. Se implantaron técnicas de tolerancia a fallas y mejoras en el hardware de la computadora de vuelo, las cuales proveen a la computadora de vuelo con las siguientes capacidades y características nuevas:
   a. Tolerancia a errores aislados en memoria RAM, debidos principalmente a la radiación, conocidos como "Single event upsets".
   b. Capacidad para la detección y diagnóstico de fallas en los microprocesadores y sus periféricos.
   c. Capacidad de detección de fallas de procesamiento y de reconfiguración por medio de vigía de tiempo ó "Watchdog".
   d. Capacidad para desarrollar y ejecutar software completamente nuevo para el la CV, aún después de que el satélite haya sido orbitado, lo que implica la posible corrección y actualización del software de operaciones de la CV.
   e. Ampliación de la memoria de 256k bytes a 1.1875 Mbytes para el almacenamiento de imágenes o datos de telemetría.

4. Se efectuaron pruebas de validación para el hardware de la computadora de vuelo, mediante las cuales se sentaron las bases para la ejecución de pruebas durante la integración del satélite y el proceso de calificación espacial (pruebas de termovacío y vibración).

5. Con la utilización de componentes de montaje superficial y la ejecución de las revisiones del hardware de la computadora, se logró la reducción del número de tarjetas que la conforman.

8.2 Recomendaciones

Algunas de las recomendaciones para el uso o la mejora de la computadora de vuelo y de lo desarrollado la presente tesis, son las siguientes:

La computadora de vuelo puede ser utilizada también para aplicaciones industriales, con el uso de componentes comerciales, para esto se ha considerado el uso de componentes disponibles en versiones militares y comerciales, con excepción del EDAC (solo militar o espacial) y del Microprocesador (sólo comercial o industrial), para esto sería suficiente con la sustitución de todos los componentes militares con sus equivalentes comerciales. El EDAC no sería incluido en este tipo de aplicación debido a que no desempeña una función crítica dentro del ambiente industrial.

Al sustituir el conjunto del hardware formado por el EDAC, su lógica de control y el decodificador de memoria, por un solo dispositivo lógico programable de calificación militar o comercial (sin EDAC) según sea el caso, se obtendría una reducción de espacio para la tarjeta del procesador, además de que se podría reducir el número de capas y costo de fabricación del impreso.

Con la finalidad de facilitar el procesamiento de forma paralela de la computadora de vuelo, es recomendable concentrar una parte de la memoria en un solo circuito impreso, esta memoria se encontraría compartida por todos los procesadores, y su acceso estaría controlado por medio del dispositivo EDAC y el decodificador de memoria correspondiente, además de ganar en la velocidad de procesamiento, se ganaría también en la reducción del número de componentes de la computadora de vuelo en general, en el costo de la misma debido principalmente a que el EDAC es el componente más costoso de la computadora (aproximadamente $2,000 Dlls), y se estaría utilizando el 100% del potencial del EDAC

En el caso de que se requiriera el uso de la arquitectura de la computadora de vuelo para una aplicación tolerante a fallas con una necesidad mayor de procesamiento o almacenamiento de datos, es recomendable considerar el cambio del microcontrolador por otro de periféricos y arquitectura similares con mayor capacidad de direccionamiento y procesamiento, pero conservando en la medida de lo posible la misma arquitectura de la computadora de vuelo.

BIBLIOGRAFÍA

[ACCEL, 1989] Referente Manual
Tango Schematic, Tango Pcb.
Accel Technologies 1990.

[AVIZIENIS, 1981] Avizienis A.
Fault Tolerance By Means Of External Monitoring Computer Systems, Proceedings of the National Computer Conference, pp 27-40. 1981

[AVIZIENIS, 1985] Avizienis A.
The N-Versión Aproach To Fault Tolerant Software, IEEE Transactions on Software Engineering, SE-11 (12): 1491-501
1985.

[GINSBERG, 1991] Gerald L. Ginsberg,
Printed Circuit Design,
McGraw-Hill, 1991.

[IDT, 1996] Application Note AN-64
Protecting Your Data With The Idt49c465 32-Bit Flow-ThruEDC™ Unit
Integrated Device Technology, Inc., 1996.

[MAHMOOD, 1988] Mahmood A., E. J. Mc Cluskey.
Concurrent Error Detection Using Watchdog Processors, IEEE Transactions on Computers, 37 (2): 160-74
1988.

[MEJIA, 2002] Mejía Sosa Iris A.
Análisis de confiabilidad de la instrumentación de vuelo y sensores para un satélite experimental.
Tesis de licenciatura, Facultad de Estudios Superiores Cuautitlán, Enero 2002.

[MELO, 1996] Melo Cerrano Victor
Teleadquisidor de datos para aplicaciones espaciales.
Tesis de Licenciatura, Facultad de Ingeniería UNAM, 1996

[PROTEL, 1999] On-Line Manual And Documentation
Protel 99 User's Guide
Protel Technologies LTD, 1999.

[PISACANE, 1994] Pisacane L. Vincent and Moore C. Robert.
Fundamentals of Space Systems
Oxford University Press, 1994.

[SIEMENS, 1993] Advance Information 9.94
Microcomputer Components
8xC166, SAB Family ApNotes
Application Note of the On-Chip Bootstrap Loader
Siemens AG, 1993.

[SIEMENS, 1997] User's Manual Ver. 06.90/08.97
Microcomputer Componetns SAB 80C166/83C166 16-Bit Single Chip Microntrollers For Embedded Applications
Siemens AG, 1997, - 1st Ed 1990.

[TASKING, 1993] 80166 C Compiler Manual
C 80C166 Cross-Compiler
BSO/Tasking, 1993.

[TEMIC, 1997-1] Application Note ANM052
Radiation Tolerant SRAM for SPACE Applications
TEMIC Semiconductors, Rev. C (Dec-1997)

[TEMC, 1997-2] Data Sheet 29C516E
16-Bit Flor-Through EDAC
Error Detección And Correction unit
TEMIC Semiconductors, Rev D (Dec-1997)

[TORRES, 2002] Torres Fuentes Juan Ramón
Software de operaciones, de tolerancia a fallas y e telecomunicaciones para un microsatélite experimental.
Tesis de licenciatura, Facultad de Ingeniería UNAM, 1996.

[VICENTE, 1994] Vicente Vivas Esaú et. al.
Diseño y construcción del prototipo de una computadora industrial tolerante a fallas.
Informe técnico, Instituto de Ingería UNAM.
Mayo 1994.

[VICENTE, 1996] Vicente Vivas Esaú et. al.
Computadora de vuelo, programación e instrumentación para un microsatélite de órbita baja.
Informe técnico, Instituto de Ingeniería UNAM, Agosto 1996.

[VICENTE, 1998] Vicente Vivas Esaú et. al.
Validación de la Instrumentación para el microsatélite SATEX.
Informe técnico del proyecto 6143, patrocinado por el IMC, Instituto de Ingeniería UNAM, Junio 1998.

[VICENTE, 1999] Vicente Vivas Esaú
CTF-3PRO Computadora tolerante a fallas con capacidad de multiprocesamiento para microsatélites.
SOMI XIV Congreso de Instrumentación, Puebla, México.
Septiembre 1999.

[VICENTE, 2000] Vicente Vivas Esaú
Distributed Fault Tolerant Instrumentation for an Experimental Microsatellite
IASTED, First International Symposium on Advanced Distributed Systems (ISADS 2000), March 2000, Guadalajara Jalisco, México
Marzo 2000.

[VICENTE, 2001] Vicente Vivas Esaú et. al.
Instrumentación de vuelo espacial para microsatélites.
Informe técnico, Instituto de Ingeniería UNAM, Octubre 2001.

[VICENTE, 2002] Vicente Vivas Esaú, et al.
Software de Adquisición de telemetría y control de operaciones para microsatélites.
4ª Conferencia Internacional en Control, Instrumentación Virtual y Sistemas Digitales, "CICINDI 2002" Pachuca, México.
Agosto 2002.

APENDICE A: Código de precarga y carga de monitor

```
;=====================================================================+
; Archivo: STXPLOAD.ASM
; Autor: Hugo A. Ortíz
; Descripción: Archivo que continen el codigo de precarga y de carga de
; monitor para la C.V. de SATEX
;                                                                     |
;=====================================================================+

      $ABSOLUTE

Bootcode      section code at 00000h
              org    0000h

F32BYT        PROC   NEAR
              mov     R0,  #0FA60h       ;CARGA AP EN R0
W0:
              jnb     S0RIR, W0   ;ESPERA A RECIBIR UN BYTE
              movb    [R0], S0RBUF       ;GUARDA EL BYTE EN LOC APUNT POR R0
              bclr    S0RIR        ;BORRA RXINTREQUEST
              movB    S0TBUF,[R0] ;ECHO CHAR

WaitTx:
              jnb     S0TIR, WaitTx      ;ESPERA A QUE SEA TRANSMITIDO EL ECO

              bclr    S0TIR        ;BORRA TXINREQUEST

              cmpi1   R0, #0FB43h       ; read 196 bytes DIR=FA60+BINLONG-33
              jmpr    cc_NE, W0          ;CAMBIE F8 POR f6 DEBIDO A echo
              NOP                      ;AGREGA NOP PRA 32 BYTES
;------- AQUI TERMINA DE BAJARSE EL PRIMER LOOP 32 BYTES ------
F32BYT        ENDP

;=====================================================================+
;  BTL1                                                               |
;=====================================================================+
; (0xFA60)


; --- initialize bus configuration ------------------------------------
;ES MUY IMPORTANTE QUE LAS INSTRUCCIONES SE HAGAN EN EL ORDEN QUE SE ENCUENTRA
;DESCRITO EN EL DOCUMENTO 8xC166 BOSTRAP LOADER DE SIEMENS
;ADVANCE  INFORMATION 9.94,  DE  OTRA  FORMA  NO  SE  PUEDE  ACCEDER  LA  MEMORIA
EXTERNA
;EN  LOS  PRIMEROS  32K,  YA  QUE  SE  ENCONTRARÁ  MAPEADA  EN  ESTE  RANGO  LA  ROM
INTERNA


IBUSCON             PROC   NEAR
              mov     SYSCON, #0110001010000000b ; instruccion para deshab
                                                 ;la rom interna mas info ver
                                                 ;arch btl loader only 166 by
                                                 ;siemens

IBUSCON             ENDP

; --- initialize CPS ---------------------------------------------------
ICPS          PROC   FAR

              jmps    far nextp1          ; far intersegment jump to update CPS
                                        ; ojo en el .btl se debe sumar la direccion
                                         ; FA40 a la direccion codificada, para este
                                        ; este salto, ya que se compilo abs en 0
```

```
;el codigo que se debe camiar en btl aparece asi
;0xE6 0x86 0x80 0x62<-----CODICO DE CARGA DE SYSCON #6280h   DEBE SER EL EL
VALOR DE SYSC
;0xFA 0x00 0x28 0x00 <--- estos dos bytes son los que se deben cambiar se suma
;0xE6 0x00 0x00 0x00          p ejemplo si estos fueran los que se tubieran
deberian
                        ;ser 0028h+FA40h=FA68h-----> 0x68 0xFA
ICPS            ENDP


; --- initialize DPPx ------------------------------------------------------
INDPPX          PROC   NEAR
nextpl:
                org    0FA68H
next:
                ORG    00028H
                mov     DPP0, #0
                mov     DPP1, #1
                mov     DPP2, #2
                mov     DPP3, #3
INDPPX          ENDP


; (0xFA80)

; --- set up WR# as an output  am init ext. bus to system config-------------
; LA INICIALIZACION DEL BUS SE HACE AQUI DEBIDO A QUE ANTES SE DESHABILITO
;LA ROM SOLAMENTE MAS INFO EN DOC. BOOTSTRAP LOADER SIEMENS 8XC166

NOWR            PROC NEAR
                bset    P3.13                ; set WR#
                bset    DP3.13               ; make WR# an output
                mov    SYSCON,#110011011000000b  ;CON BHE# COMO I/O
NOWR            ENDP


; --- set up serial port ---------------------------------------------------
;SETSER              PROC   NEAR
;               bset    DP3.10
;               mov     S0CON, #8011h
;               mov     S0BG,  #3Fh
;               mov     S0TIC, #0
;               mov     S0RIC, #0
;               mov     S0EIC, #0
;SETSER              ENDP
; --- serial port initialized for 40MHz oscillator frequency ---


;     system initialization done -------------------------------------------
INDONE          PROC   NEAR
                DISWDT
INDONE          ENDP


;----SEND ACKNOWLEGE SYSCON EN 2 PARTES-------------------
SNDACK          PROC NEAR

;PRIMER BYTE ENVIADO ES LA PARTE BAJA DE SYSCON
W0ACK:
                jnb     S0RIR, W0ACK       ;ESPERA UNA RECEPCIÓN DE CUALQUIER
                                        ;CARACTER, NO importa YA QUE ES DESECHADO
                bclr    S0RIR              ;BORRA RX INT REQUEST
                mov     R0, SYSCON         ;ENVIA
                movb   S0TBUF, R10         ;LA PARTE BAJA DE SYSCON
W1:
                jnb     S0TIR, W1          ;ESPERA A QUE SE TRANSMITA
```

```
                bclr    S0TIR                   ;BORRA TXINTREQUEST

;SEGUNDO BYTE ES ENVIADO ES LA PARTE ALTADE SYSCON

W1ACK:
                jnb     S0RIR, W1ACK                ;ESPERA UNA RECEPCIÓN
                bclr    S0RIR                   ;BORRA RX INT REQUEST

                movb    S0TBUF, Rh0             ;ENVIA PARTE ALTA DE SYSCON
W2:
                jnb     S0TIR, W2               ;ESPERA QUE SE TRANSMITA
                NOP
                NOP
                bclr    S0TIR                   ;BORRA TXINTREQUEST
SNDACK          ENDP

;-------********FIN DE MANDA SYSCON********---------

; Recibe dir de incio de Monitor
LOADBINL        PROC NEAR
WAITRXHini:
                JNB     S0RIR,WAITRXHini
                BCLR    S0RIR
                MOVB    RH4,S0RBUF      ;GUARDA DIR EN R4
WAITRXLini:
                JNB     S0RIR,WAITRXLini
                BCLR    S0RIR
                MOVB    RL4,S0RBUF      ;GUARDA PARTE BAJA
;MANDA ECHO DE DIR
                BCLR    S0TIR
                MOVB    S0TBUF,RH4
WAITECHODIR1:
                JNB     S0TIR,WAITECHODIR1
                BCLR    S0TIR
                MOVB    S0TBUF,RL4
WAITECHODIR2:
                JNB     S0TIR,WAITECHODIR2
                BCLR    S0TIR

;--Carga Valor de generación de Baudaje en loc 402h Para monitor----
;-- el monitor se comunica  a la misma velocidad---
                mov     R1,#0402H
                mov     R2,S0BG
                mov     [R1],R2

; ------------------------------------------------------------------------------
; --- CARGA CODIGO, EN ESTE CASO EL MONITOR A PARTIR DE DIR DE INICNIO EN R4 -
.
                MOV     R6,R4           ; SALVA APUNTADOR
                MOV     R0,#0001h       ;CONTADOR R0
                BCLR    S0TIR
                BCLR    S0RIR
                NOP
                NOP
WAITCODELOW:
                JNB     S0RIR,WAITCODELOW
                MOVB    RL1,S0RBUF              ;RL1 CONTIENE DATO BAJO
                AND     R1,#000FFH              ;MASK LOW BYTE
                BCLR    S0RIR                   ;BORRA INT REQUEST
                MOVB    S0TBUF,RL1              ;MANDA ECO
WAITECHOLOW:
                JNB     S0TIR,WAITECHOLOW       ;MANDA EL ECO DEL BYTE BAJO
                BCLR    S0TIR
                NOP
                NOP
WAITCODEHIGH:
```

```
                JNB     S0RIR,WAITCODEHIGH  ;RECIBE BYTE ALTO, FORMA EL WORD Y LO
GUARDA
                ORB     RH1,S0RBUF          ;GUARDA BYTE EN PARTE ALTA DE
                BCLR    S0RIR               ;R1
                MOV     [R4],R1                     ;CONTENIDO DE R1 A DIR APUNTADA
POR R4
                                            ; WORD FORMADO

                ADD     R4,#00001H          ;R4 APUNT A PARTE ALTA
                ADD     R0,#00001H          ;INC CONTADOR
                MOVB    S0TBUF,[R4]         ;MANDA ECO

WAITECHOHIGH:
                JNB     S0TIR,WAITECHOHIGH  ;ESPERA ENVIO DE ECO
                BCLR    S0TIR
                ADD     R4,#00001H          ;INCREM APUNTDOR A SIGUIENTE WORD
                NOP
                NOP
                CMPI1   R0,#0460.           ;CONTADOR DE LONG DE MONITOR EN BYTES
                JMPR    cc_NE,WAITCODELOW
;PON INSTRUCCION DE SALTO EN CERO A LA LOC DE R6(INICIO DE MONITOR)
                MOV     R4,#00000H
                MOV     R5,#000EAH
                MOV     [R4],R5
                MOV     R4,#00002h
                MOV     R5,R6
                MOV     [R4],R5
                NOP
                NOP
                JMPA cc_UC,FININI           ;SALTA A DIRECCIÓN CERO

                ORG     00000h              ;TRUCO PARA QUE SE EFECTUE EL SALTO
FININI:                                             ;A FININI QUE ES DIR CERO

LOADBINL        ENDP
BOOTCODE        ENDS
                END
```

APÉNDICE B: Código Monitor de programas de SATEX

```
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+
;
;       Nombre: Satex Program Monitor;
;       Archivo: StxPrMon.asm
;       Autor: Hugo A. Ortíz
;       DESCR: PROGRAMA BASICO DE MONITOREO PARA LA CARGA REMOTA DE
;       PROGRMAS A CV3PRO DE SATEX, DESDE ET.
;
;
;       ¡¡¡¡IMPORTANTE!!!!:
;       ESTE PROGRMA ESTA CODIFICADO DE TAL FORMA QUE
;       NINGUNO DE LOS SALTOS EFECTUADOS, ES EJECUTADO
;       EN FORMA ABSOLUTA, PARA QUE SEA  POSIBLE SU
;       CARGADO EN CUALQUIER PARTE DE LA MEMORIA,
;       SI EL COMPILADOR MARCA UN  ERROR A ESTE RESPECTO
;       SE DEBERA EFECTUAR EL SALTO EN DOS PARTES.
;
;
;
;
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+
$ABSOLUTE
$SEGMENTED                          ; Modo de memoria Segementado
$CASE                               ; Simbolos sensibles mayusculas
; Constant definitions
$SAVE
$NOLIST

;---------------------- Acknowledges  DE INICIALIZACIÓN--------------

I_LOADER_STARTED            EQU    001h   ; Cargador iniciado
I_APPLICATION_LOADED        EQU    002h   ; Aplicación cargada
I_APPLICATION_STARTED       EQU    003h   ; Applicación iniciada

;----------------------  Function Codes ----------------------------

C_WRITE_BLOCK       EQU     084h   ; Escribe bloque de memoria
C_READ_BLOCK        EQU     085h   ; Lee bloque memoria
C_EINIT                     EQU     031h   ; Ejecuta instrucción EINIT
C_SWRESET                   EQU     032h   ; Ejecuta Software Reset
C_GO                        EQU     041h   ; Salta a progrma de usuario
C_GETCHECKSUM       EQU     033h   ; Encia el Chksum del úiltimo blq
C_TEST_COMM                 EQU     093h   ; Prueba comunicación
C_CALL_FUNCTION             EQU     09Fh   ;  Interface  de  Ext.  Monitor:  CALL
Function
C_WRITE_WORD        EQU     082h   ; Escribe Word a memoria o a registro
C_MON_EXT                   EQU     09Fh   ;  Interface  de  ext.  monitor:  CALL
Function

;  ------ ------------  Acknowledges de Monitor--------------------

A_ACK1              EQU     0AAh   ; 1er ackn: código de función ok!
A_ACK2              EQU     0EAh   ; 2o ackn:  Función ejecutada ok!

;---------------------- Valores de Error ----------------------------

E_WRITE                     EQU     011h   ; Error de Escritura

$RESTORE
```

```
NAME   MINIMON
ASSUME DPP3:system


PUBLIC MiniMonProc
GLOBAL EndMiniMonProc

MiniMonCode0 SECTION CODE AT 0000h      ;OJO LOCALIZADO EN LOCALIDAD CERO
MiniMonProc  PROC FAR                   ;PARA   QUE   SE   PUEDAN   USAR   LAS
HERRAMIENTAS
ORG    00000h                           ;HEX2BIN BTL2BIN PARA PROPOSITOS DE
                                                ;DEPURACIÓN


Init:
            DISWDT
            bset   P3.13                ; activa  WR#
            bset   DP3.13               ; activa WR# como salida
            mov          SYSCON,#110011011000000b  ;CON BHE# COMO I/O
; --- set up serial port ----------------------------------------------------
            mov          R0,#0402H              ;dirección  donde  se  guarda  la
config de
                                                ;baudios de
            bset     P3.1               ;Pin de señalización para DT en alto
            bset     DP3.1              ;dirección   de   pin   de   señalización
SALIDA
            mov      S0CON, #8011h
            mov      S0BG,  #003Fh
            mov      S0TIC, #0
            mov      S0RIC, #0
            mov      S0EIC, #0
; --- serial port initialized for 40MHz oscillator frequency ---
            ;DISWDT
            EINIT                       ; deshabilita watchdog
            BCLR   S0RIR                ; borra receive interrupt bit
            BCLR   S0TIR                ; borra transmit interrupt bit
            MOVB   RH0,#(I_APPLICATION_STARTED)
            CALLR  CSend                ; trans. ackn: aplicación iniciada
            BCLR   P3.1                 ;Avisa a DT-->Continuar



Main:
      JNB    S0RIR,Main           ;Espera Comando
      BCLR   S0RIR
      MOVB   RL0,S0RBUF           ; RL0 contiene el código de cmd.

      MOVB   RH0,#(A_ACK1)
      CALLR  CSend                ; trans. ACK1
      MOVB   RH0,#(A_ACK2); Guarda ACK2 en el registro de Tx

;*******************************************************************************
*
; Comando EINIT
;*******************************************************************************
*

CCmd1:
      CMP    RL0,#(C_EINIT)       ; comando 'EINIT' ?
      JMPR   cc_NE,CCmd2
      EINIT                       ; call EINIT
      CALLR  CSend                ; envia aknowledge ACK2
      JMPR   Main
```

```
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+
; Comando Test Communication
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+

CCmd2:
       CMP    RL0,#(C_TEST_COMM); comando 'Prueba Comunicación'
       JMPR   cc_NE,CCmd3
       CALLR  CSend              ; envia acknowledge ACK2
       JMPR   Main

;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+
; Comando Software Reset
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+

CCmd3:
       CMP    RL0,#(C_SWRESET)   ; comando 'Software Reset' ?
       JMPR   cc_NE,CCmd4
       BSET   P3.1               ; AVISA A DT EXITO, LINEA RCPROG
       CALLR  CSend              ; envia acknowledge antes de SRST
       SRST                      ; último comando: software reset


;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+
; Comando 'GetChecksum'
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+

CCmd4:
       CMP    RL0,#(C_GETCHECKSUM)
       JMPR   cc_NE,CCmd5
       MOV    RH0,RL5            ; pon el Checksum en el registro de Tx.
       CALLR  CSend              ; envia checksum
       MOV    RH0,#(A_ACK2)
       CALLR  CSend              ; envia acknowledge: hecho
       JMPR   Main

;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+
; Comando Escribe bloque a Memoria
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+

CCmd5:
       CMP    RL0,#(C_WRITE_BLOCK)      ; comando 'Write Block' ?
       JMPR   cc_NE,CCmd6
       MOVB   RL5,#0             ; Reset registro de checksum: RL5
       CALLR  CAddr1             ; recibe 3 bytes de dirección R1/R2
       CALLR  CLeng1             ; recibe 2 bytes longitud en R3

CWrtL:
       CMPD1  R3,#0000           ; checa longitud
       JMPR   cc_NE,CWrt1
       CALLR  CSend              ; envia ACK2
       JMPR   Main               ; retorna a Main

CWrt1:
       JNB    S0RIR,CWrt1        ; recibe byte de dato
       BCLR   S0RIR
       MOVB   RL4,S0RBUF         ; RL4 contiene el dato
       XORB   RL5,RL4            ; Calcula Checksum
       CALLR  CDPP               ; dirección R1,R2 en DPP2 y R6
```

```
        MOV     R12,R6          ; prepara R12 como dirección de 16bit
        BCLR    R12.0                   ; inicializa A0 a 0
        MOV     R7,[R12]                ; lowbyte: leer el byte de dato16bit en R7

        JB      R6.0,CWrHb              ; ?highbyte o lowbyte
        AND     R7,#0FF00h              ; no se modifica el highbyte
        ORB     RL7,RL4                 ; escribe dato(8 bit) en RL7
        JMPR    CWrNx
CWrHb:
        AND     R7,#00FFh               ; no se modifica lowbyte
        ORB     RH7,RL4                 ; escribe dato(8 bit) en RH7
CWrNx:
        MOV     [R12],R7                ; Escribe dato(16 Bit)de R7 a R6
        MOVB    RH4,[R6]                ; Carga a RH4
        CMP     RH4,RL4                 ; compara valor leido con el original
        JMPR    cc_NE,CWErr             ; jmp si no coinciden.
        JMPR    CWrtL
CWErr:
        MOVB    RH0,#(E_WRITE)          ; registro de Tx= Error
        JMPR    CWrtL

;*****************************************************************************
+
; Comando Lee Bloque desde Memoria
;*****************************************************************************
+

CCmd6:
        CMP     RL0,#(C_READ_BLOCK) ; comando 'Read Block' ?
        JMPR    cc_NE,CCmd7
        MOVB    RL5,#0                  ; Reset checksum: RL5
        CALLR   CAddr1                  ; recibe 3bytes dirección R1/R2
        CALLR   CLeng1                  ; recibe 2bytes longitud R3

CRdL:
        CMPD1   R3,#0000                ; checa longitud
        JMPR    cc_NE,CRd1
        MOVB    RH0,#(A_ACK2); Registro Tx = ACK2
        CALLR   CSend           ; envia acknowledge ACK2
        JMPR    Main

CRd1:
        CALLR   CDPP                    ; dirección  de R1,R2 a DPP2 y R6
        MOVB    RH0,[R6]                ; lee de memoria a RH0
        XORB    RL5,RH0                 ; Calcula Checksum
        CALLR   CSend                   ; transmite dato leido
        JMPR    CRdL

;*****************************************************************************
+
; Comando 'Go'
;*****************************************************************************
+

CCmd7:
        CMP     RL0,#(C_GO)             ; comando 'Go' ?
        JMPR    cc_NE,CCmd8
        CALLR   CAddr1          ; recibe 3bytes dirección R1/R2
        CALLR   CSend                   ; envia acknowledge entes del JMP
        PUSH    R2                      ; push R2 (como CSP)
        PUSH    R1                      ; push R1 (como IP)
        RETS                            ; activa  CSP y R1 al mismo tiempo

;*****************************************************************************
+
```

```
; Comando escribe Word
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+

CCmd8:
       CMP    RL0,#(C_WRITE_WORD) ; comando 'Write Word' ?
       JMPR   cc_NE,CCmd9

       CALLR  CAddr1              ; recibe 3bytes dirección R1/R2
       CALLR  CLengl              ; recibe 2bytes, word en R3
       CALLR  CDPP                     ; dirección R1,R2 a DPP2 y R6
       MOV    [R6],R3                  ; escribe de R3 a memoria
       CALLR  CSend                    ; envia acknowledge ACK2
MAIN1:
       JMPR   Main

;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
; CSend: envia por puerto serial el contenido de RH0
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

CSend:  NEAR
       MOVB   S0TBUF,RH0          ; transmite RH0
CSendL:
       JNB    S0TIR,CSendL ; espera fin de T::
       NOP
       BCLR   S0TIR
       RETN

;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+
; Comando Call / interface de extensión de Monitor
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+

CCmd9:
       CMP    RL0,#(C_MON_EXT)    ; comando 'Call' ?
       JMPR   cc_NE,Main
       CALLR  CAddr1       ; recibe 3bytes dirección R1/R2
       CALLR  CR16                 ; recib parametro de 16bytes en R8-R15
       CALLR  CExtN                ; salto falso para empujar al IP al Stack

CExtN: NEAR
       POP    R7                   ; almacena IP en R7
       ADD    R7,#10h              ; Offset para retornar a la linea CExtR
       PUSH   CSP                  ; push posición de retorno
       PUSH   R7
       PUSH   R2                   ; push destino
       PUSH   R1
       RETS                        ; jump a dir. destino

CExtR:
       CALLR  CS16                 ; posición de retorno
       MOV    RH0,#(A_ACK2)
       CALLR  CSend                ; envia acknowledge despues del retorno
       JMPR   MAIN1



;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
; CAddr1: recibe 3bytes de direcciones en R1 y RL2
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

CAddr1: NEAR
       JNB    S0RIR,CAddr1         ; R1=A0-A15
       BCLR   S0RIR
       MOV    RL1,S0RBUF
```

```
CAddr2:
       JNB     S0RIR,CAddr2
       BCLR    S0RIR
       MOV     RH1,S0RBUF
CAddr3:
       JNB     S0RIR,CAddr3          ; RL2=A16-A24
       BCLR    S0RIR
       MOV     RL2,S0RBUF
       MOV     RH2,#0
       RETN

;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
; CLeng1: recibe 2bytes de longitud en R3
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

CLeng1: NEAR
       JNB     S0RIR,CLeng1          ; R3=longitud
       BCLR    S0RIR
       MOV     RL3,S0RBUF
CLeng2:
       JNB     S0RIR,CLeng2
       BCLR    S0RIR
       MOV     RH3,S0RBUF
       RETN

;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
; CDPP: activa el uso del DPP2 mediante R6,de acuerdo a la dir en R1 y R2
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

CDPP:  NEAR
       MOV     R6,R1                 ; R6 contiene A0-A15
       MOV     R7,R2                 ; R7 contiene A16-A23
       SHR     R6,#14          ; R6 Bits 0-1 son A14-A15
       SHL     R7,#2                 ; R7 Bits 2-9 son A16-A23
       OR      R7,R6                 ; R7 contiene A14-A23
       MOV     DPP2,R7               ; DPP2 contiene A14-A23
       MOV     R6,R1                 ; R6 contiene A0-A15, incrementa R1
       ADD     R1,#1
       ADDC    R2,#0h          ; incrementa segmento R2 si es necesario
       AND     R6,#3FFFh             ; R6 Bits 0-13 contienen A0-A13
       OR      R6,#8000h             ; R6 Bits 14-15 contienen 10=DPP2

       RETN

;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
; CR16: recibe  parametros de cmd CALL en  R8-R15
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


CR16:  NEAR
       MOV     R6,CP
       ADD     R6,#16
       MOV     R7,#0
CRLp:
       JNB     S0RIR,CRLp
       BCLR    S0RIR
       MOVB    [R6],S0RBUF
       ADD     R6,#1
       CMPI1   R7,#15
       JMP     cc_NE,CRLp
       RETN

;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
; CS16: envia parametros de salida de CALL, en R8-R15
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
CS16:  NEAR
       MOV     R6,CP
       ADD     R6,#16
       MOV     R7,#0
CSLp:  MOVB    RH0,[R6]
       CALLR   CSend
       ADD     R6,#1
       CMPI1   R7,#15
       JMPR    cc_NE,CSLp
       RETN

       RETV                          ; RETORNO virtual, no genera codigo
                                     ; solo para evitar warnings

EndMiniMonProc:      FAR

MiniMonProc          ENDP
MiniMonCode0 ENDS

END
```

APÉNDICE C: Rutina de autodiagnóstico de CV

```
/*****************************************************************
** Archivo AuodPrb.C                                            **
**                                                              **
** Descripción: Implementación de las funciones de Autodiagnóstico **
** de la C.V., se anexan funciones de Acceso a Memoria y de control **
** de puertos serie para efectuar un monitoreo externo          **
**                                                              **

** Autor: Hugo A. Ortiz                                         **
**                                                              **
*****************************************************************/

#include <stdio.h>
#include <c166.h>
#include <reg166.h>
#include "ser9600.h"

/*
 *      Version : @(#)SER9600.c   1.3     10/18/93
 */

/*****************************************************************
**                                                              *
**  FILE         :  SER9600.c                                   *
**                                                              *
**  DESCRIPTION :  Implementation of _ioread() and _iowrite()   *
**                 using the low level I/O functions getch(), putch(), *
**                 kbhit() and init_serio() working with serial *
**                 channel 0 or channel 1 of the SAB 80C166.    *
**                 Default is serial channel 0 used for I/O.    *
**                 If serial channel 1 is wanted for I/O compile *
**                 "SER1200.c" with the -DSER_PORT_1 option.    *
**                                                              *
**  COPYRIGHT    :  1993 Tasking Software B.V., Amersfoort      *
**                                                              *
*****************************************************************/
/*#include <stdio.h>
#include <reg166.h>
#include "ser9600.h"*/

/* S0/1 port:
 * 8 bit, asynchronuous operation, one stopbit,
 * receive enable, no parity/frame/overrun check
 * baud rate generator enable
 *
#define SXCON_MOD1      0x8011
#define BAUDRAT1        0x003F          /* 9600 Baud using 39.3216 MHz  */
#define SXCON_MODE      0x8011
#define BAUDRATE        0x003F          /* 9600 Baud using 39.3216 MHz  */
#define C_control               'Z'

void
init_serio( void )
{
        _bfld( DP3, MSK_TDX_RDX, DP3_TDX_RDX ); /* direction bits       */
        _putbit( 1, P3, P3_TXD );       /* enable TXD0/TXD1 output      */
        SXBG  = BAUDRATE;
        SXTIC = 0;                              /* clear errorflags     */
        SXRIC = 0;
        SXEIC = 0;
        SXTIR = 1;
        SXCON = SXCON_MODE;
```

```
}

/*
 * Read character from serial channel
 */
int
getch( void )
{
        int c = EOF;

        if ( SXEIR )
        {
                SXPE = 0;
                SXFE = 0;
                SXEIR = 0;
                SXRIR = 0;
        }
        else if ( SXRIR )
        {
                c = SXRBUF;
                SXRIR = 0;
        }
        return ( c );
}

/*
 * Return 1 if character available, otherwise 0
 */
int
kbhit( void )
{
        if ( SXRIR )
                return ( 1 );
        return ( 0 );
}

/*
* Write character to serial channel
*/
int
putch01( int c )
{
        int cc,ii,iii;
         cc=c;
         ii=_ror(cc,8);
         iii=ii & 0x00FF;


        while ( ! SXTIR );
        SXTIR  = 0;
        SXTBUF = c;
        while ( ! SXTIR );
        SXTIR  = 0;
        SXTBUF = iii;

        return ( c );
}
int
putch( int c )
{

        while ( ! SXTIR );
        SXTIR  = 0;
        SXTBUF = c;

        return ( c );
```

```
}


void
init_serio2( void )
{
        _bfld( DP3, MSK_TDX_RDX1, DP3_TDX_RDX1 ); /* direction bits        */
        _putbit( 1, P3, P3_TXD1 );          /* enable TXD0/TXD1 output      */
        SXBG1  = BAUDRAT1;
        SXTIC1 = 0;                                 /* clear errorflags     */
        SXRIC1 = 0;
        SXEIC1 = 0;
        SXTIR1 = 1;
        SXCON1 = SXCON_MOD1;
}

/*
 * Read character from serial channel
 */
int getch2( void )
{
        int c = EOF;

        if ( SXEIR1 )
        {
                SXPE1 = 0;
                SXFE1 = 0;
                SXEIR1 = 0;
                SXRIR1 = 0;
        }
        else if ( SXRIR1 )
        {
                c = SXRBUF1;
                SXRIR1 = 0;
        }
        return ( c );
}

/*
 * Return 1 if character available, otherwise 0
 */
int
kbhit2( void )
{
        if ( SXRIR1 )
                return ( 1 );
        return ( 0 );
}

/*
 * Write character to serial channel
 */
int putch02( int c )
{
         int cc,ii,iii;
        cc=c;
        ii=_ror(cc,8);
        iii=ii & 0x00FF;

         while ( ! SXTIR1 );
         SXTIR1  = 0;
         SXTBUF1 = c;
         while ( ! SXTIR1 );
         SXTIR1  = 0;
         SXTBUF1 = iii;
```

```
        return ( c );
}

int
putch2( int c )
{

         while ( ! SXTIR1 )

         {
              _srvwdt();
         };
         SXTIR1  = 0;
         SXTBUF1 = c;

        return ( c );
}


/*** funciones retardo, prende y apaga 20veces****/
void retar(void)
{
  int i,j,k;
  for(i=0;i< 10;i++)
          {
                for(j=0;j<253;j++)
        {
         for(k=0;k<253;k++)
                {
                      _srvwdt();
                }
        }
         }
}

void ledp20(void)
       {
              int q;
              for (q=0;q<10;q++)
       {
       _putbit(1,P3,0);
       retar();
       _putbit(0,P3,0);
       retar();
       }
       retar();
       retar();
        }


/* FUNCION PARA EL REFRESCO DE LA MEMORIA RAM */


int RefrescoMem(int Pagina)
{
       #pragma combine nb=A64592

       int PaginaAsm;

       PaginaAsm=Pagina;

                                          /* respaldo de los DPP's*/
```

```
      /*en la siguiente rutina se efectua el refresco de la pag
      la pagina es la 3, no se refresca la ram interna->sfr's,--Solo stack */
      putch2('A');

      putch2('R');

      #pragma asm
                              MOV  R1,#8000h          ;Inicializa      R1      la
seleccion del dpp2 #10XXXXXXXXbin
                              MOV R4,#0FC50H       ;lee el numero de pag
                              MOV R2,[R4]
                              MOV R3,DPP2             ;respalda el DPP2
                              MOV  DPP2,R2                  ;coloca  el  num  de
pag en el DPP2


      RefRamIni:

                              MOV  R2,[R1]            ;copia  a  R2  el  Contenido
de dir Apunt por R1,
                              MOV          [R1],R2          ;copia  a  localidad
ap por R1, el conten de R2
                              SRVWDT                        ;servicio al wdt
                              CMPI2        R1,#0BBFEh   ;FIN DE STACK SI ESTOY pag
3
                              JMPR CC_NE,RefRamIni     ;EJECUTA LOOP HASTA QUE SE
ALCANCE
                              CMP          DPP2,#2          ;VERIFICA          EL
CONTENIDO DE DPP, 3--> Pag 3
                             ; JMPR        CC_EQ,FINREFRAM     ; SI ESTOY EN pag 3
->FIN
      REFN3PAG:
                              MOV          R2,[R1]          ;   SE   REPITE   EL
CICLO ANTERIOR PARA LAS LOCS SIG
                              MOV          [R1],R2          ;REESCRIBE
                              SRVWDT                        ;SERVICIO  AL
WDT
                              CMPI2        R1,#0BFFEh   ;CON EL FIN DE PAG
                              JMPR CC_NE,REFN3PAG           ;REGRESA AL LOOP
      FINREFRAM:
                              MOV          DPP2,R3          ;RESTAURA EL NUMERO
DE PAG
                              NOP                                   ;   FIN   DE
RUTINA
      #pragma endasm

      putch2('o');
      putch2('k');


      return(0);

}



/**** FUNCION DE AUTODIAGNOSTICO DE CPU Y PERIFFERICOS-***/

int DiagCPU(int VECTORP)
{

      register int Rw;    /* uso de registros R6-R9*/
```

```
        register int Rx;    /* registro temporal para dato leido*/
        register int Ry;
        register int Rz;
        register int SINDROME;      /* tambien en r12*/
        int StatusTimers;
        int i;
        int MuestreoRam0[20];
        int MuestreoRam1[20];
        int MuestreoRam2[20];
        int MuestreoRam3[20];
        int MuestreoRam4[20];
        int MuestreoRam5[20];
        int MuestreoRam6[20];
        int MuestreoRam7[20];
        int MuestreoRam8[20];
        int MuestreoRam9[20];


        int watchdogT,Timer0,Timer1,Timer2,Timer3,Timer4,Timer5,Timer6;
/* variables de control de interrupciones, recarga y configuración de los
timers */
        int T0_I_CONTROL,T1_I_CONTROL,T2_I_CONTROL,T3_I_CONTROL;
        int T4_I_CONTROL,T5_I_CONTROL,T6_I_CONTROL;
        int t0reload,t1reload,t2reload,t3reload,t4reload,t5reload,t6reload;
        int t0config,t1config,t2config,t3config,t4config,t5config,t6config;
/* variables que me indicarán que timer estoy corriendo yo*/
        int T0IRUN,T1IRUN,T2IRUN,T3IRUN,T4IRUN,T5IRUN,T6IRUN;
        int NumErrRam;
        int ResMuestreoRam[10][20];       /* matriz que guardará resultado de
todos
                                  los vectores de prueba  de los muestreos de
ram*/

        WDTCON=0X001;

for(i=0;i<=19;i++)
{
        MuestreoRam0[i]=VECTORP;
        MuestreoRam1[i]=VECTORP;
        MuestreoRam2[i]=VECTORP;
        MuestreoRam3[i]=VECTORP;
        MuestreoRam4[i]=VECTORP;
        MuestreoRam5[i]=VECTORP;
        MuestreoRam6[i]=VECTORP;
        MuestreoRam7[i]=VECTORP;
        MuestreoRam8[i]=VECTORP;
        MuestreoRam9[i]=VECTORP;
}

        _srvwdt();

        SINDROME=0x000;
/*INICIALIZA POSESIÓN DE TIMERS A 0----> NO POSEO NINGUNO*/
        T0IRUN=0,T1IRUN=0,T2IRUN=0,T3IRUN=0,T4IRUN=0,T5IRUN=0,T6IRUN=0;
        putch2('T');
/* VERIFICA QUE TIMER SE ESTAN USANDO, LOS QUE NO SE USAN YO LOS TOMO Y LOS
ACTIVO */
        if(!T0R)                          /*T0  corriendo  si->  no  modifiques
nada*/
        {                                 /* NO-> TOMA POSESIÓN DEL TIMER*/
                T0_I_CONTROL=(int)T0IC;   /* RESPALDA CONFIGURACIÓN*/
                t0reload=(int)T0REL;
                t0config=(T01CON&0x00FF);       /* SOLO QUIERO CONFIG TIMER0*/

                /* CONFIGURA EL TIMER PARA PROPOSITOS DE PRUEBA*/
```

```
            T0IC=0X0;                   /* NO INTERUPPCIÓN*/
            T0REL=0XAAAAA;              /* MAYOR PERIODO DE OVERFLOW*/
            T0M=0;

            T0R=1;                      /* SOLO MODIFICO T0 PARA MENOR PERIODO
DE TRANSISIONES */
                                        /* Y LO PONGO A CORRER */
            T0IRUN=0X1;                 /* AQUÍ INDICO QUE TOMO POSESIÓN DEL
TIMER*/
            putch2('0');

      }

      if(!T1R)                  /*T0 corriendo si-> no modifiques nada*/
      {                                 /* NO-> TOMA POSESIÓN DEL TIMER*/
            T1_I_CONTROL=(int)T1IC;         /* RESPALDA CONFIGURACIÓN*/
            t1reload=(int)T1REL;
            t1config=(T01CON&0xFF00)>>8;    /* SOLO QUIERO CONFIG TIMER1*/

            /* CONFIGURA EL TIMER PARA PROPOSITOS DE PRUEBA*/

            T1IC=0X0;                   /* NO INTERUPPCIÓN*/
            T1REL=0XAAAAA;              /* MAYOR PERIODO DE OVERFLOW*/
            T1M=0;                      /* SOLO MODIFICO T1 PARA MENOR PERIODO
DE TRANSISIONES */
                                        /* Y LO PONGO A CORRER */
            T1R=1;

            T1IRUN=0X1;                 /* AQUÍ INDICO QUE TOMO POSESIÓN DEL
TIMER*/
            putch2('1');

      }

      if(!T2R)                          /*T2 corriendo si-> no modifiques
nada*/
      {                                 /* NO-> TOMA POSESIÓN DEL TIMER*/
            T2_I_CONTROL=(int)T2IC;         /* RESPALDA CONFIGURACIÓN*/
            t2reload=T2;                /*el reload es el valor del timer */
            t2config=T2CON;                 /* configuración */

            /* CONFIGURA EL TIMER PARA PROPOSITOS DE PRUEBA*/


            T2IC=0X0;                   /* NO INTERUPPCIÓN*/
            T2=0x0;
            T2CON=0x0040;               /* SOLO MODIFICO T2 PARA MENOR PERIODO
DE TRANSISIONES */
                                        /* Y LO PONGO A CORRER */
            T2IRUN=0X1;                 /* AQUÍ INDICO QUE TOMO POSESIÓN DEL
TIMER*/
            putch2('2');

      }

      if(!T3R)                          /*T3 corriendo si-> no modifiques
nada*/
      {                                 /* NO-> TOMA POSESIÓN DEL TIMER*/
            T3_I_CONTROL=(int)T3IC;         /* RESPALDA CONFIGURACIÓN*/
            t3reload=T3;                /*el reload es el valor del timer */
            t3config=T3CON;                 /* configuración */

            /* CONFIGURA EL TIMER PARA PROPOSITOS DE PRUEBA*/

            T3IC=0X0;                   /* NO INTERUPPCIÓN*/
            T3=0x0;
```

```
            T3CON=0x0040;                   /* SOLO MODIFICO T3 PARA MENOR PERIODO
DE TRANSISIONES */
                                            /* Y LO PONGO A CORRER */
            T3IRUN=0X1;                     /* AQUÍ INDICO QUE TOMO POSESIÓN DEL
TIMER*/
            putch2('3');

      }

      if(!T4R)                              /*T4 corriendo si-> no modifiques
nada*/
      {                                     /* NO-> TOMA POSESIÓN DEL TIMER*/
            T4_I_CONTROL=(int)T4IC;               /* RESPALDA CONFIGURACIÓN*/
            t4reload=T4;                    /*el reload es el valor del timer */
            t4config=T4CON;                       /* configuración */

            /* CONFIGURA EL TIMER PARA PROPOSITOS DE PRUEBA*/

            T4IC=0X0;                       /* NO INTERUPPCIÓN*/
            T4=0x0;
            T4CON=0x0040;                   /* SOLO MODIFICO T4 PARA MENOR PERIODO
DE TRANSISIONES */
                                            /* Y LO PONGO A CORRER */
            T4IRUN=0X1;                     /* AQUÍ INDICO QUE TOMO POSESIÓN DEL
TIMER*/
            putch2('4');

      }

      if(!T5R)                               /*T5 corriendo si-> no modifiques
nada*/
      {                                      /* NO-> TOMA POSESIÓN DEL TIMER*/

       /*   WDTCON=0x01;
            _srvwdt();*/
            T5_I_CONTROL=(int)T5IC;               /* RESPALDA CONFIGURACIÓN*/
            t5reload=T5;                    /*el reload es el valor del timer */
            t5config=T5CON;                       /* configuración */

            /* CONFIGURA EL TIMER PARA PROPOSITOS DE PRUEBA*/

            T5IC=0x0;                                                   /*          NO
INTERUPPCIÓN*/
            T5=0x0;
            T5CON=0x0040;                                       /* SOLO MODIFICO T5
PARA MENOR PERIODO DE TRANSISIONES */

      /* Y LO PONGO A CORRER */
            T5IRUN=1;                                   /*  AQUÍ  INDICO  QUE  TOMO
POSESIÓN DEL TIMER*/
            putch2('5');

      }

      if(!T6R)            /*T6 corriendo si-> no modifiques nada*/
      {                             /* NO-> TOMA POSESIÓN DEL TIMER*/
      /*    WDTCON=0x01;
            _srvwdt(); */
            T6_I_CONTROL=(int)T6IC;   /* RESPALDA CONFIGURACIÓN*/
            t6reload=T6;              /*el reload es el valor del timer */
            t6config=T6CON;                 /* configuración */

            /* CONFIGURA EL TIMER PARA PROPOSITOS DE PRUEBA*/

            T6IC=0X0;                 /* NO INTERUPPCIÓN*/
            T6=0x0;
```

```
            T6CON=0x0040;                   /* SOLO MODIFICO T6 PARA MENOR PERIODO
DE TRANSISIONES */
                                            /* Y LO PONGO A CORRER */
            T6IRUN=0X1;                     /* AQUÍ INDICO QUE TOMO POSESIÓN DEL
TIMER*/
            putch2('6');

      }
      putch2('T');
      WDTCON=0x0001;

      _srvwdt();                            /* servico al wdt al iniciar rutina*/
      watchdogT=WDT;                              /*  lee  la  Configuración  del
watchdog*/
      Timer0=0;                             /*LEE los valores de todos los timers*/
      Timer1=0;
      Timer2=T2;
      Timer3=T3;
      Timer4=T4;
      Timer5=T5;
      Timer6=T6;

/* al final se deben restaurar todos las config  de los timers que tome */


/*************************************************************************
*****
**   PRUEBA   PARA   VERIFICAR   ALU,   SE   EFECTUAN   +,-,*,/,SHIFT<-,SHIFT-
>,AND,OR,NOT,XOR,**
** SI ESTA PRUEBA NO ES EFECTUADA SATISFACTORIAMENTE, ES MUY POSIBLE QUE EL
**
** PROCESADOR SE ENCUENTRE ATASCADO, O QUE LA RESPUESTA DE FALLA SE REALIZE DE
**
**                              FORMA                                INCORRECTA,
**
*************************************************************************
****/

      putch2('A');
      putch2('L');
      putch2('U');

      Rw=VECTORP;
      Rx=VECTORP;
      Ry=VECTORP;
      Rz=Rw-Rx;           /*resta*/               /*Rz=0x0*/
      Rz=Rz+Ry;     /*suma*/                  /*Rz=VECTORP*/
      Rz=_rol(Rz,16); /*shift left --rotar 16 */    /*Rz=VECTORP*/
      Rz=_ror(Rz,16);     /*shift right --rotar 16 */     /*Rz=VECTORP*/
      Rz=Rz/Rx;     /* división*/                 /*Rz=0x1*/
      Rz=Rz*Ry;     /* multiplicación*/         /*Rz=VECTORP*/
      Rz=Rz&Rx;     /* AND */                   /*Rz=VECTORP*/
      Rz=Rz^(~Rx); /*XOR y NOT*/               /*Rz=0xFFFF */
      Rz=~(Rz&(~Rx));     /*NOT y AND */                 /*Rz=VECTORP*/
      Rz=Rz|0x0000;/*OR*/                     /*Rz=VECTORP*/
      if(Rz!=VECTORP)     /*verifica resultado*/
      {
            SINDROME=0x01; /*SINDROME DE FALLA DE ALU*/
      }
      else
      {
            SINDROME=0x0;/* SINDROME SIN FALLAS */
      }

      putch2('O');
      putch2('K');
```

```
/* SIGUIENTE RUTINA EFECTUA UN MUESTREO DE LA MEMORIA RAM, EN ARREGLOS
PREDEFINIDOS DE 20 INT's */
/* DISTRIBUIDOS A LO LARGO DE TODA LA MEMORIA EXCEPTO MEMORIA INTERNA */
/* CON ESTE SE PRUEBA EL FUNCIONAMIENTO DEL CONTROLADOR DEL BUS EXTERNO,
INTERNO Y DPP's */
/*si hay edac, el mismo los detecta y los corrige, si no hay edac reporta el
numero de errores*/
        NumErrRam=0;
        WDTCON=0x001;
        _srvwdt();


        for(Ry=0;Ry<20;Ry++)
        {
                ResMuestreoRam[0][Ry]=MuestreoRam0[Ry];
                ResMuestreoRam[1][Ry]=MuestreoRam1[Ry];
                ResMuestreoRam[2][Ry]=MuestreoRam2[Ry];
                ResMuestreoRam[3][Ry]=MuestreoRam3[Ry];
                ResMuestreoRam[4][Ry]=MuestreoRam4[Ry];
                ResMuestreoRam[5][Ry]=MuestreoRam5[Ry];
                ResMuestreoRam[6][Ry]=MuestreoRam6[Ry];
                ResMuestreoRam[7][Ry]=MuestreoRam7[Ry];
                ResMuestreoRam[8][Ry]=MuestreoRam8[Ry];
                ResMuestreoRam[9][Ry]=MuestreoRam9[Ry];
         }
        for(Ry=0;Ry<20;Ry++)
        {
                for(Rw=0;Rw<10;Rw++)
                {
                        if(ResMuestreoRam[Rw][Ry]!=VECTORP) /* verifica cada uno de
los errores*/
                        {
                                NumErrRam++;
                                SINDROME=SINDROME|0x2;  /*enmascara el 2 bit del
sindrome, errores en ram
                                                        SIGNIFICA QUE ACUMULA
ERRORES ANTERIORES*/
                        }

                }
        }
        putch2('M');
        putch2('R');
        putch2('O');
        putch2('K');

/* ciclo de refresco de ram se refresca una pagina a la vez */

        for(Rx=0;Rx<16;Rx++)
        {
                NumErrRam=RefrescoMem(Rx)+NumErrRam;
                putch2(('0'+Rx));
        }
        if(NumErrRam!=0)
        {
                Sindrome=Sindrome|0x2;
        }


/* compruebo que los timers hayan cambiado su valor lo que nos indica que
funcionan*/
```

```
        Rx=0x0; /*inicio contador */
        putch2('T');
        StatusTimers=0x3FC;
        do
        {
                if(Timer0!=T0)
                {   StatusTimers=StatusTimers&(~0x4);

                }
                if(Timer1!=T1)
                {   StatusTimers=StatusTimers&(~0x8);
                }
                if(Timer2!=T2)
                {   StatusTimers=StatusTimers&(~0x10);

                }
                if(Timer3!=T3)
                {   StatusTimers=StatusTimers&(~0x20);

                }
                if(Timer4!=T4)
                {   StatusTimers=StatusTimers&(~0x40);

                }
                if(Timer5!=T5)
                {   StatusTimers=StatusTimers&(~0x80);

                }
                if(Timer6!=T6)
                {   StatusTimers=StatusTimers&(~0x100);

                }
                if(watchdogT!=WDT)
                {
                        StatusTimers=StatusTimers&(~0x200);
                        _srvwdt();
                }
                WDTCON=0x01;
                _srvwdt();

                Rx++;

        }while(!((StatusTimers==0x0)||(Rx==0xFFFF)));
                if(StatusTimers!=0X0)
                {
                      putch2('o');
                      putch2('R');
                }
                putch2('T');

                putch2('O');
                putch2('K');
                _srvwdt();


        SINDROME=(~SINDROME)|(StatusTimers);

/* reconfigura los timers*/

        if(T0IRUN)                          /* si poseo el timer restaura*/
        {
                T0IC=T0_1_CONTROL&0x00FF; /* RESTAURA  CONFIGURACIÓN*/
                T0REL=t0reload;
                T01CON=T01CON&(t0config|0xFF00); /* SOLO QUIERO CONFIG TIMER0*/
        }
```

```
        if(T1IRUN)                          /* si poseo el timer restaura*/
        {
                T1IC=T1_I_CONTROL&0x00FF;           /* RESTAURA  CONFIGURACIÓN*/
                T1REL=t1reload;
                T01CON=T01CON&((t1config<<8)|0x00FF);   /*   SOLO   QUIERO   CONFIG
TIMER1*/
        }


        if(T2IRUN)                          /*si poseo el timer restaura */
        {
                T2IC=T2_I_CONTROL&0x00FF;  /* RESTAURA CONFIGURACIÓN*/
                T2=t2reload;               /*el reload es el valor del timer */
                T2CON=t2config;                   /* configuración */
        }


        if(T3IRUN)                          /*si poseo el timer restaura */
        {
                T3IC=T3_I_CONTROL&0x00FF;  /* RESTAURA CONFIGURACIÓN*/
                T3=t3reload;               /*el reload es el valor del timer */
                T3CON=t3config;                   /* configuración */
        }


        if(T4IRUN)                          /*si poseo el timer restaura */
        {
                T4IC=T4_I_CONTROL&0x00FF;  /* RESTAURA CONFIGURACIÓN*/
                T4=t4reload;               /*el reload es el valor del timer */
                T4CON=t4config;                   /* configuración */
        }


        if(T5IRUN)                          /*si poseo el timer restaura */
        {
                T5IC=T5_I_CONTROL&0x00FF;  /* RESTAURA CONFIGURACIÓN*/
                T5=t5reload;               /*el reload es el valor del timer */
                T5CON=t5config;                   /* configuración */
        }


        if(T6IRUN)                          /*si poseo el timer restaura */
        {
                T6IC=T6_I_CONTROL&0x00FF;  /* RESTAURA CONFIGURACIÓN*/
                T6=t6reload;               /*el reload es el valor del timer */
                T6CON=t6config;                   /* configuración */
        }


        return(~SINDROME);

}



/**funcion main--**********************/
void main(void)
{

        int SINDROME;
        int ciclosW,i;

        int VecPru;
```

```
        WDTCON=0x001;
        _putbit(1,DP3,0);
        _putbit(1,DP3,1);


        ledp20();
        init_serio2();/*inicializa comunicaci¢n serial***/
        _srvwdt();
        putch2('a');
        putch2('u');
        _srvwdt();
        putch2('t');
        putch2('o');
        _srvwdt();
        putch2('d');
        putch2('i');
        _srvwdt();
        putch2('a');
        putch2('g');
        _srvwdt();
        putch2(' ');
        putch2('i');
        _srvwdt();
        putch2('n');
        putch2('i');
        _srvwdt();

        SINDROME=0x0;

        VecPru=0xAAAA;
        SINDROME=DiagCPU(VecPru);
        _srvwdt();
        DP2=0x0028;    /*Config. del Puerto P3.6 como salida,P3.7 como salida
(LED)          */

        _srvwdt();
        putch2('S');
        putch2('I');
        _srvwdt();
        putch2('N');
        putch2('D');
        putch2('R');
        _srvwdt();
        putch2('O');
        putch2('M');
        putch2('E');
        _srvwdt();
        putch2('o');
        putch2('K');
        _srvwdt();

        ledp20();
        putch2('L');
        _srvwdt();
        putch2('0'+(SINDROME&0x000F));
        putch2('0'+((SINDROME&0x00F0)>>4));
        _srvwdt();
        putch2('H');
        putch2('0'+((SINDROME>>8)&0x00F));
        _srvwdt();
        putch2('0'+((SINDROME>>12)&0x0F));
        _srvwdt();
        _putbit(1,P3,0);
        retar();
```

```
        putch2(' ');
        putch2('W');
        putch2('W');
        putch2('W');
        for(i=1;i<=100;i++)
        {
                putch2('0'+i);
                _srvwdt();
                for(ciclosW=0;ciclosW<(2000)+(i*10);ciclosW++)
                {
                        _nop();
                        _nop();
                        _nop();
                        _nop();
                }
                _srvwdt();
        }


}
```

PAGINACIÓN DISCONTINUA

APÉNDICE D: Rutinas de acceso y control de memoria expandida

```
/*****************************************************************************
** Archivo Textxm.c                                                         **
** Autor: Hugo A. Ortíz M.                                                  **
**                                                                          **
** Descripción: Implementación de la rutina de refresco de memoria RAM      **
** expandida, y programa de prueba de la memoria expandida                  **
*****************************************************************************/


#include <stdio.h>
#include <c166.h>
#include <reg166.h>
#include "ser9600.h"

/*
 *      Version : @(#)SER9600.c   1.3     10/18/93
 */

/*****************************************************************************
**                                                                          **
**  FILE        :  SER9600.c                                                **
**                                                                          **
**  DESCRIPTION :  Implementation of _ioread() and _iowrite()               **
**                 using the low level I/O functions getch(), putch(),      **
**                 kbhit() and init_serio() working with serial             **
**                 channel 0 or channel 1 of the SAB 80C166.                **
**                 Default is serial channel 0 used for I/O.                **
**                 If serial channel 1 is wanted for I/O compile            **
**                 "SER1200.c" with the -DSER_PORT_1 option.                **
**                                                                          **
**  COPYRIGHT   :  1993 Tasking Software B.V., Amersfoort                   **
**                                                                          **
*****************************************************************************/
/*#include <stdio.h>
#include <reg166.h>
#include "ser9600.h"*/

/* S0/1 port:
 * 8 bit, asynchronuous operation, one stopbit,
 * receive enable, no parity/frame/overrun check
 * baud rate generator enable
 *
#define SXCON_MOD1      0x8011
#define BAUDRAT1        0x003F          /* 9600 Baud using 39.3216 MHz  */
#define SXCON_MODE      0x8011
#define BAUDRATE        0x003F          /* 9600 Baud using 39.3216 MHz  */
#define C_control               'Z'


void
init_serio( void )
{
        _bfld( DP3, MSK_TDX_RDX, DP3_TDX_RDX ); /* direction bits        */
        _putbit( 1, P3, P3_TXD );       /* enable TXD0/TXD1 output      */
        SXBG  = BAUDRATE;
        SXTIC = 0;                              /* clear errorflags     */
        SXRIC = 0;
        SXEIC = 0;
        SXTIR = 1;
```

```
        SXCON = SXCON_MODE;
}

/*
 * Read character from serial channel
 */
int
getch( void )
{
        int c = EOF;

        if ( SXEIR )
        {
                SXPE = 0;
                SXFE = 0;
                SXEIR = 0;
                SXRIR = 0;
        }
        else if ( SXRIR )
        {
                c = SXRBUF;
                SXRIR = 0;
        }
        return ( c );
}

/*
 * Return 1 if character available, otherwise 0
 */
int
kbhit( void )
{
        if ( SXRIR )
                return ( 1 );
        return ( 0 );
}

/*
* Write character to serial channel
*/
int
putch01( int c )
{
        int cc,ii,iii;
         cc=c;
         ii=_ror(cc,8);
         iii=ii & 0x00FF;


        while ( ! SXTIR );
        SXTIR  = 0;
        SXTBUF = c;
        while ( ! SXTIR );
        SXTIR  = 0;
        SXTBUF = iii;

        return ( c );
}
int
putch( int c )
{

        while ( ! SXTIR );
        SXTIR  = 0;
        SXTBUF = c;
```

```
        return ( c );
}


void
init_serio2( void )
{
        _bfld( DP3, MSK_TDX_RDX1, DP3_TDX_RDX1 ); /* direction bits        */
        _putbit( 1, P3, P3_TXD1 );          /* enable TXD0/TXD1 output      */
        SXBG1  = BAUDRAT1;
        SXTIC1 = 0;                                 /* clear errorflags     */
        SXRIC1 = 0;
        SXEIC1 = 0;
        SXTIR1 = 1;
        SXCON1 = SXCON_MOD1;
}

/*
 * Read character from serial channel
 */
int getch2( void )
{
        int c = EOF;

        if ( SXEIR1 )
        {
                SXPE1 = 0;
                SXFE1 = 0;
                SXEIR1 = 0;
                SXRIR1 = 0;
        }
        else if ( SXRIR1 )
        {
                c = SXRBUF1;
                SXRIR1 = 0;
        }
        return ( c );
}

/*
 * Return 1 if character available, otherwise 0
 */
int
kbhit2( void )
{
        if ( SXRIR1 )
                return ( 1 );
        return ( 0 );
}

/*
 * Write character to serial channel
 */
int putch02( int c )
{
         int cc,ii,iii;
        cc=c;
        ii=_ror(cc,8);
        iii=ii & 0x00FF;

         while ( ! SXTIR1 );
         SXTIR1  = 0;
         SXTBUF1 = c;
         while ( ! SXTIR1 );
         SXTIR1  = 0;
```

```
        SXTBUF1 = iii;

      return ( c );
}

int
putch2( int c )
{

        while ( ! SXTIR1 )
        {
            _srvwdt();
        };
        SXTIR1  = 0;
        SXTBUF1 = c;

      return ( c );
}


/*** funciones retardo, prende y apaga 20veces****/
void retar(void)
{
  int i,j,k;
  for(i=0;i< 10;i++)
        {
            for(j=0;j<253;j++)
       {
        for(k=0;k<253;k++)
            {
                _srvwdt();
            }
       }
        }
}

void ledp20(void)
      {
            int q;
            for (q=0;q<10;q++)
      {
      _putbit(1,P3,0);
      retar();
      _putbit(0,P3,0);
      retar();
      }
      retar();
      retar();
       }


/*************************************************
** rutinas de activación de mem expandida       **
*************************************************/
/* Variable global que indica el numero de segmento**
** expandido actual gSegExpAct                  **
*************************************************/

int gSegExpAct;
int RefrescoMem(int Pagina);
void InitRamExp(void);
void ActivaRamExp(int NumSeg);
int  RefrescoRamExp(int NumSeg);
```

```
/*****************************************************
** función para inicialización de dir. Mem. Exp     **
** Inicializa los puertos de salida para las        **
** direcciones de la memoria Expandida A18-A20      **
*****************************************************/
void InitRamExp(void)
{
   _putbit(1,DP3,15);
   _putbit(1,DP3,14);
   _putbit(1,DP3,7);
   _putbit(1,DP2,6);
   ActivaRamExp(0);
}

/*****************************************************
** funcion que activa un segmento(64Kbytes) de la ram **
** expandid, el numero permitido es de 0 a 15, si es  **
** 0 se accesa al segemnto 03 de la memoria no exp    **
*****************************************************/
void ActivaRamExp(int NumSeg)
{
   int A18,A19,A20,A21; /*variables de cada direccion*/
   A18=NumSeg&(0x01);
   A19=(NumSeg&(0x02))>>1;
   A20=(NumSeg&(0x04))>>2;
   A21=(NumSeg&(0x08))>>3;
/*****************************************************
** en las siguientes líneas se habilitan las líneas E/s **
** del MCU, correspondientes a las direcciones A18-A21  **
*****************************************************/
   _putbit(A18,P3,15);
   _putbit(A19,P3,14);
   _putbit(A20,P3,7);
   _putbit(A21,P2,6);
   gSegExpAct=NumSeg;
}

/* funcion de refresco de memoria exp */
int  RefrescoRamExp(int NumSeg)
{
   int pagina;
   int SegExpAct;
   int NumErr=0;
   SegExpAct=gSegExpAct;
   ActivaRamExp(NumSeg);
   for(pagina=12;pagina<16;pagina++)
   {
      NumErr=RefrescoMem(pagina)+NumErr;
   }
   ActivaRamExp(SegExpAct);
   return(NumErr);

}


/* funcion de refresco de memoria por pagina **
** de 16kbytes, es independiente del control **
** de la ram expandida                       **
*********************************************/
int RefrescoMem(int Pagina)
{
      #pragma combine nb=A64592
```

```
        int PaginaAsm;
        PaginaAsm=Pagina;

                                        /* respaldo de los DPP's*/

        /*en la siguiente rutina se efectua el refresco de la pag
        la pagina es la 3, no se refresca la ram interna->sfr's,--Solo stack */
        /*putch2('A');

        putch2('R'); */

        #pragma asm
                                MOV R1,#8000h       ;Inicializa  R1  la  seleccion
del dpp2 #10XXXXXXXXbin
                                MOV R4,#0FC50H ;R4=&PaginaAsm
                                MOV R2,[R4]    ;R2=PaginaAsm
                                MOV R3,DPP2    ;respalda el DPP2
                                MOV DPP2,R2    ;coloca el num de pag en el DPP2
                                MOV [R4],ZEROS       ;PaginaAsm=0


        RefRamIni:

                                MOV  R2,[R1]              ;R2=*R1
                                MOV  [R1],R2              ;*R1=R2
                                CMP     R2,[R1]           ;?*R1==R2
                                JMPR    CC_NE,ERRAM
        RetERRAM:
                                SRVWDT                     ;servicio al wdt
                                CMPI2       R1,#0BFFEh    ;FIN DE PAGINA
                                JMPR CC_NE,RefRamIni      ;EJECUTA LOOP HASTA QUE SE
ALCANCE
                                JMPR CC_UC,FINREFRAM ;SALTA AL FIN DE REFRESCO RAM
        ERRAM:
                               MOV R2,[R4]          ;R2=PaginaAsm
                               ADD  R2,#01          ;R2++
                               MOV [R4],R2          ;PaginaAsm=R2=Numero de Err
                               JMPR CC_UC,RetERRAM   ;Continua con ciclo



        FINREFRAM:
                                MOV          DPP2,R3              ;RESTAURA  EL  NUMERO
DE PAG
                                NOP                                    ;    FIN    DE
RUTINA
        #pragma endasm

/*      putch2('o');
        putch2('k'); */



        return(PaginaAsm);

}

/**funcion main--**********************/
void main(void)
{
        int i,IndSegExp;
        unsigned int IndOffset;

        int ByteH,ByteB;
        int ciclosW;
        int huge * hpMemExp;
```

```
        WDTCON=0x001;
        _putbit(1,DP3,0);
        _putbit(1,DP3,1);

        ledp20();
        init_serio2();      /*inicializa comunicacion serial***/
        _srvwdt();
        InitRamExp();       /*inicializa ram expandida */
        putch2('R');
        putch2('x');
        for(i=0;i<16;i++)
        {
              putch2('0'+i);
              _srvwdt();
              RefrescoRamExp(i);
        }
        _srvwdt();
        hpMemExp=_mkhp(0,03); /* forma el app al segmento 03 */
        putch2('R');
        for(i=0;i<16;i++)
        {
              _srvwdt();
              putch2('e');
              ActivaRamExp(i);
              *hpMemExp=i; /* escribe el numero de segmento exp dentro de cada
seg*/
              putch2('0'+i);
        }
        for(i=0;i<16;i++)
        {
              _srvwdt();
              ActivaRamExp(i);
              putch2('L');
              putch2('0'+(*hpMemExp));

        }

        _srvwdt();
        putch2('M');
        putch2('E');
        _srvwdt();
        putch2('M');
        putch2('E');
        putch2('X');
        _srvwdt();
        putch2('P');
        putch2(' ');
        putch2('O');
        _srvwdt();
        putch2('K');
        putch2(' ');
        _srvwdt();
        ledp20();
        ledp20();
        for(IndSegExp=0;IndSegExp<16;IndSegExp++)

              ActivaRamExp(IndSegExp);
              _putbit(0,P3,0);
              for(IndOffset=0;IndOffset<0x534;IndOffset)
              {

                    hpMemExp=_mkhp(IndOffset,03);
                    if*
                    {

                          _srvwdt();
                          while(kbhit2()==0)
```

```
                        ByteB=getch2();
                        IndOffset++;
                        do
                        {
                                _srvwdt();
                        }while(kbhit2()==0 );
                        _srvwdt();
                        ByteH=getch2();
                        IndOffset++;
                        *hpMemExp= ((ByteH<<8)&0xFF00)+(ByteB&0x0FF);
                        _srvwdt();

                }

        _putbit(1,P3,0);

        }

        ledp20();
        ledp20();
        ledp20();
        for(IndSegExp=0;IndSegExp<16;IndSegExp++)
        {
                ActivaRamExp(IndSegExp);
                for(IndOffset=0;IndOffset<65534;IndOffset)
                {
                        hpMemExp=_mkhp(IndOffset,03);
                        putch2(*hpMemExp);
                        putch2((*hpMemExp>>8)&0xFF);
                        IndOffset=IndOffset+2;
                        _srvwdt();

                }

        }

        ledp20();
/*      putch2('W');
        putch2('W');
        putch2('W');
        for(i=1;i<=100;i++)
        {
                putch2('0'+i);
                _srvwdt();
                for(ciclosW=0;ciclosW<(4000)+(i*10);ciclosW++)
                {
                        _nop();
                        _nop();
                        _nop();
                        _nop();
                }
                _srvwdt();
        }

*/



}
```