132
2ej.

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

IMPLEMENTACIÓN CON DSPs DE UN ANALIZADOR Y VISUALIZADOR DE POSICIÓN DE PULSOS DE UN SISTEMA MEPSICRÓN

TESIS PROFESIONAL
QUE PARA OBTENER EL TÍTULO DE

INGENIERO MECÁNICO ELECTRICISTA
EN EL ÁREA ELÉCTRICA Y ELECTRÓNICA

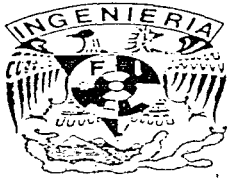
PRESENTA

JUSTO SEIJI OSHINO ORTIZ

DIRECTOR DE TESIS:

FÍS. ARTURO IRIARTE VALVERDE

MÉXICO, D. F. 1997



AGRADECIMIENTOS

A mis padres
por su invaluable apoyo

A mis maestros
por su ejemplo

DEDICATORIA

ご指導のために
佐々木先生に
著書を捧げます

(Al profesor Sasaki
por su guia)

A Marcela Esparza
por todo

ÍNDICE

Introducción 3

Capítulo primero. Antecedentes 9

1.1. Conceptos de óptica 9
- 1.1.1. Observaciones astronómicas
- 1.1.2. Difracción
- 1.1.3. Diafragmación

1.2. El análisis espectral 11
- 1.2.1. El espectro electromagnético
- 1.2.2. El mecanismo de radiación
- 1.2.3. El espectro de líneas
  - 1.2.3.1. Espectro de emisión
  - 1.2.3.2. Espectro de absorción
- 1.2.4. Las líneas espectrales
  - 1.2.4.1. Posición
  - 1.2.4.2. Perfil de línea
- 1.2.5. Los espectros continuos
- 1.2.6. El átomo de hidrógeno

1.3. Conceptos de astronomía 15
- 1.3.1. Densidad de flujo
- 1.3.2. Magnitudes aparentes

1.4. Parámetros en el empleo de detectores 16
- 1.4.1. Los fotones
- 1.4.2. Detección de fotones
- 1.4.3. La eficiencia cuántica

1.5. Detectores 18
- 1.5.1. La placa fotográfica
- 1.5.2. Los espectrógrafos
- 1.5.3. Los fotocátodos y fotomultiplicadores
- 1.5.4. Los fotómetros y polarímetros
- 1.5.5. Detectores basados en semiconductores
  - 1.5.5.1. Intensificadores
  - 1.5.5.2. La cámara Vidicon
  - 1.5.5.3. El CCD
- 1.5.6. Interferómetros

Capítulo segundo. El sistema Mepsicrón 25

2.1. Descripción general del equipo 25
2.1.1. El fotocátodo
2.1.2. El sistema de placas microcanal
2.1.3. El ánodo resistivo

2.2. Características básicas del detector 28

2.3. Observaciones 28
2.3.1. El intervalo dinámico
2.3.2. Limitación por el APP
2.3.3. Recepción de señales débiles
2.3.4. El viñeteo

2.4. Descripción de la electrónica 29

2.5. Ventajas del Mepsicrón sobre el CCD 32

Capítulo tercero. Los procesadores digitales de señales . . . 37

3.1. El procesamiento paralelo 38

3.2. La arquitectura del TMS320C40 38
3.2.1. CPU
3.2.1.1. Registros del CPU
3.2.2. Memoria
3.2.2.1. La organización de la memoria
3.2.2.2. RAM, ROM y caché del TMS320C4x
3.2.2.3. Modos de direccionamiento de la memoria
3.2.3. Interrupciones
3.2.4. Periféricos
3.2.4.1. Los puertos de comunicaciones de la generación TMS320C4x
3.2.4.2. Los puertos de comunicaciones del TMS320C40 en el sistema de procesamiento paralelo (PPDS)
3.2.4.3. El acceso directo a memoria (DMA)
3.2.4.4. Timers

3.3. Datos técnicos del PPDS 48

3.4. Tipos de direccionamiento 49
3.4.1. Direccionamiento indirecto
3.4.2. Direccionamiento circular

3.4.3. Direccionamiento de bit revertido
3.4.3.1. Direccionamiento de bit revertido en CPU
3.4.3.2. Direccionamiento de bit revertido en DMA

Capítulo cuarto. Desarrollo de algoritmos 55

4.1. Descripción general 55
4.1.1. Configuración de la memoria
4.1.2. Coordenadas del punto de incidencia

4.2. El lenguaje ensamblador del TMS320 58
4.2.1. Modelo de enunciados fuente
4.2.2. Introducción al COFF
4.2.2.1. Secciones
4.2.2.2. Descripción del ensamblador
4.2.3. Herramientas para programación de los DSPs
4.2.3.1. Características del compilador de C
4.2.3.2. Comandos para las herramientas

4.3. El lenguaje de programación C 64
4.3.1. Turbo C

4.4. Manejo de la información 66

4.5. Especificaciones del visualizador 66
4.5.1. Pantalla de inicio
4.5.2. Pantalla de ventanas
4.5.2.1. Ventana de ayuda
4.5.2.2. Ventana de menú
4.5.2.3. Ventana de señal
4.5.2.4. Ventana de estado

Capítulo quinto. Conclusiones 77

Apéndice A. Conceptos de computación . . . . . A - 1

A.1. Memoria RAM . . . . . A - 1

A.2. Tipos de memoria . . . . . A - 2
A.2.1. Memoria convencional del DOS
A.2.2. Memoria reservada
A.2.3. Memoria expandida o paginada

A.2.4. Memoria genérica extendida
A.2.5. Especificación de memoria extendida
A.2.5.1. Área de memoria superior
A.2.5.2. Área de memoria alta
A.2.5.3. Área de memoria extendida
A.2.6. Memoria virtual

A.3. Software del DOS para memoria . . . . . A - 4
A.3.1. Activación de controladores de dispositivos
A.3.2. Administradores de memoria
A.3.2.1. HIMEM.SYS
A.3.2.2. EMS
A.3.2.3. EMM386.EXE
A.3.2.4. EMM386
A.3.2.5. RAMDRIVE.SYS

A.4. Software del OS/2 para memoria . . . . . A - 7
A.4.1. Conmutación entre sistemas operativos
A.4.1.1. PROTECTONLY
A.4.2. Controladores de dispositivos (para memoria)
A.4.2.1. VEMM.SYS
A.4.2.2. VXMS.SYS
A.4.2.3. DISK.SYS

A.5. Manejo de memoria por diferentes sistemas operativos . . . A - 9

Apéndice B. Datos técnicos . . . . . B - 1

B.1. Diagrama a bloques del TMS320C40
B.2. Mapas de memoria
B.3. Organización de la memoria
B.4. Módulos periféricos
B.5. Diagrama a bloques del TMS320C4x PPDS
B.6. Mapa de memoria de los periféricos
B.7. Diagrama de la tarjeta del PPDS
B.8. Puertos externos de comunicaciones del PPDS

Apéndice C. Listado de programas . . . . . C - 1

Primera etapa en los DSPs . . . . . C - 1
Archivo de proceso por lotes . . . . . C - 2
Visualizador . . . . . C - 3

Biblioteca o cabecera del visualizador . . . . . C - 17
Archivo de ayuda . . . . . C - 18

Apéndice D. Descripción de pruebas . . . . . D - 1

D.1. Variables de ambiente . . . . . D - 1

D.2. Compilación en C para el TMS320 . . . . . D - 2
D.2.1. El descriptor gramatical
D.2.1.1. Opciones del descriptor gramatical
D.2.2. Opciones del ensamblador
D.2.3. Opciones generales del compilador
D.2.4. El optimizador
D.2.4.1. Opciones del optimizador
D.2.5. La utilería de entrelistado
D.2.5.1. Opciones para entrelistado

D.3. Pruebas de compilación con el comando asm30 . . . . D - 5
D.3.1. Prueba 1
D.3.2. Prueba 2

D.4. Pruebas de compilación con el comando cl30 . . . . D - 5
D.4.1. Prueba 1
D.4.2. Prueba 2
D.4.3. Prueba 3
D.4.4. Prueba 4

D.5. El enlazador . . . . . D - 7
D.5.1. Opciones del enlazador
D.5.1.1. Notas sobre las opciones
D.5.2. El archivo de comandos del enlazador

D.6. Pruebas con el enlazador y su archivo de comandos . . . D - 9
D.6.1. Prueba 1
D.6.2. Prueba 2
D.6.3. Prueba 3
D.6.4. Prueba 4
D.6.5. Prueba 5

D.7. Bibliotecas con soporte en tiempo de ejecución . . . . D - 11
D.7.1. Utilería para construcción de bibliotecas

D.8. Pruebas con la utilería . . . . . D - 12
D.8.1. Prueba 1
D.8.2. Prueba 2
D.8.3. Prueba 3
D.8.4. Prueba 4
D.8.5. Prueba 5
D.8.6. Prueba 6

D.9. Otras opciones de programación . . . . . D - 15

Bibliografía . . . . . BIB - 1

INTRODUCCIÓN

La astronomía se ha servido de muchas áreas del conocimiento para lograr resultados de mayor confiabilidad en sus campos de investigación, al tiempo que ha recurrido a los diversos avances tecnológicos para similares propósitos, dada la importancia que reviste no permanecer al margen del progreso en el mundo actual.

Los conocimientos sobre esta área de la ciencia han sido incrementados desde tiempos muy antiguos; podría decirse que el ser humano siempre ha tenido inquietud por conocer mejor las cosas que observa al dirigir la vista al cielo. Y es dicha inquietud la que ha provocado la acuñación de términos específicos para el desarrollo de la astronomía.

Por supuesto, el avance de la misma ha ido añadiendo conceptos, cambiando y mejorando parámetros, hasta llegar en la actualidad a un estado tal que no pueden estudiarse las relaciones de la astronomía con las demás áreas científicas, ni comprender sus objetivos de conocimiento ni sus problemas si no se han analizado antes dichos términos.

Por esta razón, en el primer capítulo se hace una exposición de conceptos, definición de parámetros y otros aspectos, relativos a la astronomía, que han sido considerados indispensables para una adecuada interpretación del problema que se plantea en este trabajo de tesis.

Para efectos de la investigación astronómica, una de las principales fuentes de información acerca de los cuerpos celestes está constituida por las partículas que éstos emiten, mismas que son analizadas mediante técnicas físicas y químicas. Sin embargo, es fácil ver que antes de cualquier análisis debe tenerse en cuenta la forma en que esas partículas serán recibidas: los resultados de los análisis posteriores serán prácticamente dependientes del sistema encargado de recibir las partículas.

A lo largo de la historia se han manejado diversos sistemas de observación y detección de diferentes parámetros astronómicos, desde el ojo humano mismo: la placa fotográfica, espectrógrafos, fotocátodos, fotomultiplicadores, fotómetros, polarímetros, intensificadores de imágenes y detectores basados en semiconductores, entre otros. No obstante, es conveniente indicar que la variedad de detectores existentes se debe a que ninguno de ellos es ideal para la observación astronómica.

También en el primer capítulo se hace una presentación de las características de todos los dispositivos enunciados, con el fin de hacer posteriormente una evaluación entre los mismos para así justificar el empleo de alguno de ellos. Empero, siempre se toma en cuenta la no idealidad indicada previamente.

En el Instituto de Astronomía de la UNAM se ha trabajado desde 1985 con un detector denominado Mepsicrón, que es del tipo de los fotocátodos. Actualmente se busca desarrollar en el Instituto mejores componentes de dicho detector que los que se tenían inicialmente: debido a los problemas de construcción de los fotocátodos, la empresa fabricante (ITT) dejó de producirlos.

El detector de las partículas de interés (los fotones) emitidos por algún evento, e.g. un cuerpo celeste, es solo la primera etapa del sistema Mepsicrón: una vez realizada la detección de una partícula debe procederse a su análisis, que corresponderá con el tipo de información que se pretenda obtener. En el primer capítulo se expone la gran cantidad de parámetros que pueden analizarse; no obstante, el aspecto de interés en el presente trabajo es el conocimiento de la posición exacta del evento emisor, por lo que los datos que proporcione el detector deben ser llevados a la siguiente etapa, en donde habrán de ser manejados para la obtención del propósito descrito.

No se abandona la idea de desarrollar los Mepsicrones en México, antes todo lo contrario, debido a la gran cantidad de ventajas que se les hallan al compararlos con otros sistemas. Estas ventajas, en conjunto con las especificaciones correspondientes, son presentadas en el segundo capítulo.

Es en este punto en el que es encontrado el objetivo general del presente proyecto de tesis: Hacer las mejoras necesarias al sistema Mepsicrón en lo que al aspecto electrónico se refiere, en forma tal que sean aprovechadas las características propias del mismo, es decir, desarrollar un detector astronómico eficiente, que utilice tecnología actualizada para ser capaz de competir con otros basados en diferentes principios.

La etapa de manejo de datos está constituida por un sistema de desarrollo de procesamiento paralelo con DSPs (procesadores digitales de señales), estando en el tercer capítulo la información correspondiente a este sistema, tanto de tipo individual como de conjunto.

El sistema basado en los DSPs constituye la principal mejora que se le hace a este detector en cuanto a electrónica corresponde, como se refiere en el tercer capítulo, en el que puede verse la gran cantidad de diferencias con respecto a los datos propios del segundo. En otras palabras, el tercer capítulo presenta las especificaciones de los avances buscados en hardware para el manejo de la información astronómica que atañe al sistema Mepsicrón.

Una vez hecho el análisis del problema y la forma en que se pretende realizar su solución, se hace, al inicio del capítulo cuarto, una descripción general que comprende y sintetiza estos aspectos, para inmediatamente después pasar a la descripción de las partes del sistema y los algoritmos que habrán de ser empleados en el mencionado proceso de solución.

En éste capítulo son expuestos los detalles propios de la etapa del desarrollo e implementación del procesamiento por software (en forma global e individual) en el sistema. No obstante, se hace también una presentación de las características propias de las herramientas empleadas; estas herramientas son los lenguajes de programación que fueron considerados convenientes para el proyecto.

Las especificaciones propias del programa son ampliamente tratadas tras la presentación anterior: todos los aspectos propios de la información de tipo preventivo y de auxilio para el usuario, la forma de adquisición de datos, el manejo de la información, los distintos comandos y operaciones disponibles, las maneras opcionales de despliegue de datos y, finalmente, las consideraciones necesarias para la correcta interpretación de los mismos.

Las conclusiones sobre el trabajo son presentadas en el quinto capítulo, en que se analizan y comentan los resultados obtenidos al evaluar el sistema teniendo como referencia el objetivo medular ya descrito. Se hace asimismo una descripción de las expectativas que se tienen sobre el proyecto, al tiempo que se enuncian las mejoras que fueron contempladas al realizar los diferentes procedimientos y pruebas que corresponden al mismo.

En la sección de apéndices se incluyen, con el propósito de detallar algunos aspectos, diversos datos que corresponden a dispositivos, software empleado y demás medios manejados en el cuerpo del trabajo.

El apéndice A maneja información referente al sistema operativo empleado en el proyecto, lo compara con otros más populares y, en esta forma, pretende una comprensión más adecuada de las especificaciones del mismo.

En el apéndice B se presentan diversos diagramas que complementan los datos hallados en el capítulo tercero. Asimismo, son proporcionados otros datos técnicos que fueron contemplados en el proyecto.

Los listados finales de los programas realizados se encuentran en el apéndice C. En estos listados no solamente se encuentran los elementos de programación propios de cada caso, sino que, en aquellos puntos donde se consideró adecuado o necesario, se incluye una descripción del comando, serie de comandos o rutina que se esté efectuando.

Los resultados producto de las diferentes pruebas realizadas con el conjunto de software de desarrollo para los DSPs son expuestos en el apéndice D. Se detalla más dicho software para un mejor aprovechamiento posterior de las experiencias expuestas.

Finalmente, en la bibliografía se proporcionan los datos de las diversas fuentes de información consultadas, entre las que es posible hallar libros, artículos y documentos del WWW o Internet.

CAPÍTULO PRIMERO

ANTECEDENTES

1.1. CONCEPTOS DE ÓPTICA

1.1.1. OBSERVACIONES ASTRONÓMICAS

Durante mucho tiempo (hasta el fin de la Edad Media, aproximadamente), la astronomía se valió solo del ojo humano, pero posteriormente habría de emplear diversos instrumentos, iniciando por el telescopio (principio del s. XVII) y más adelante técnicas como la fotografía astronómica (fin del s. XIX). Más recientemente, se ha auxiliado de sistemas con detectores electrónicos para estudiar la radiación electromagnética que puede recibirse del espacio. Mediante satélites y aeronaves existe, y se aprovecha, la posibilidad de realizar la gran mayoría de las observaciones astronómicas fuera de la atmósfera terrestre.

Lo anterior se basa en que en la atmósfera se presentan diversos factores que afectan las observaciones de muchas formas (la inestabilidad del aire, los cambios de temperatura y densidad entre capas, etc.). El paso de la luz de una estrella es afectado de tal forma que la cantidad de luz recibida por un detector varía al grado de decir que las estrellas centellean.

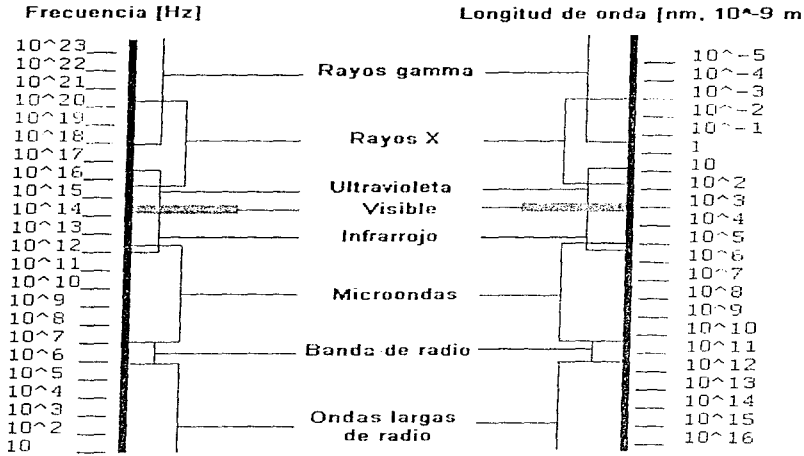
El problema del centelleo es disminuido gracias a la gran área en la que un telescopio recibe luz, pues dicha área iguala los cambios rápidos; sin embargo, existe otro problema, como describe Karttunen: "Las diferencias en refracción a lo largo de diferentes trayectorias de luz a través de la atmósfera manchan la imagen y las fuentes puntuales son vistas en el telescopio como pecas vibrantes. A este fenómeno se le llama 'seeing' y el tamaño del así llamado 'disco seeing' puede variar desde menos de un segundo de arco hasta varias decenas de segundos de arco. Tanto el seeing como el centelleo tienden a 'tachar' pequeños detalles cuando alguien vé a través del telescopio, por ejemplo, a un planeta"[1].

Además, la atmósfera absorbe algunas regiones de longitud de onda del espectro electromagnético. Puede definirse en éste la 'ventana óptica', importante intervalo comprendido entre 300 y 800 nm y que coincide con la región de sensibilidad del ojo humano (alrededor de 400 a 700 nm).

---

[1]Karttunen, Hannu et al. Fundamental Astronomy, pág. 49

En la siguiente figura, Sears[2] muestra el espectro electromagnético. La zona sombreada corresponde a la mencionada "ventana óptica".



En astronomía, muchas veces es necesario observar objetos muy borrosos, por lo que es importante que el cielo de fondo sea tan oscuro y la atmósfera tan transparente como sea posible. Por estas razones los grandes observatorios son construidos en cimas de montañas lejos de las ciudades; además, el aire sobre el sitio de un observatorio debe ser muy seco, debe haber muy pocas noches nubladas y la vista ha de ser buena.

1.1.2. DIFRACCIÓN

Sobre este aspecto, Hecht menciona: "Un cuerpo opaco a medio camino entre una pantalla y una fuente puntual forma una sombra intrincada hecha de regiones claras y oscuras. Francesco Grimaldi escribió en el siglo XVII: 'El efecto es una característica general de los fenómenos ondulatorios que ocurren donde quiera que una porción de un frente de onda sea sonido, onda material o luz, es obstruido de alguna manera'. Si en el

[2]Sears, Francis W. Fundamentos de Física III. Óptica, pág. 24

transcurso del encuentro con un objeto transparente u opaco se altera una región del frente de onda en amplitud o en fase, ocurrirá difracción. Los varios segmentos del frente de onda que se propagan más allá del obstáculo interfieren para producir la distribución de densidad de energía particular conocida como patrón de interferencia"[3].

Hecht también maneja los conceptos de interferencia y difracción sin distinción significativa entre sí. No obstante, aclara que se ha vuelto común, aunque no siempre apropiado, hablar de interferencia cuando se está considerando la superposición de solo unas cuantas ondas y de difracción cuando se está tratando un gran número de ondas.

La importancia de la difracción reside en el hecho de que los instrumentos ópticos (lentes, espejos, diafragmas, rendijas, etc.) utilizan solo una porción del frente de onda incidente completo, de tal forma que la nitidez final en un sistema óptico sin defectos estaría limitada por la difracción.

1.1.3. DIAFRAGMACIÓN

La diafragmación es la interposición de un diafragma entre la lente y el objeto, de tal modo que solo deje libre la parte central de aquella. Esto, naturalmente, ocasiona una disminución de la cantidad de luz transmitida.

1.2. EL ANÁLISIS ESPECTRAL

1.2.1. EL ESPECTRO ELECTROMAGNÉTICO

Fundamentalmente, la naturaleza de las ondas luminosas y la de otras ondas electromagnéticas es la misma. De este modo, el término *espectro* designa todo el intervalo de ondas electromagnéticas, en tanto que el espectro visible se refiere en forma particular a aquellas ondas capaces de estimular el sentido de la vista.

"Las ondas procedentes de las fuentes luminosas son emitidas por las moléculas y los átomos y abarcan desde las ondas infrarrojas relativamente largas, pasando por el espectro visible, hasta el ultravioleta. Para que una molécula o átomo pueda emitir energía radiante es necesario algún tipo de excitación. Esta excitación se produce por la agitación térmica de las moléculas, o bien es adquirida en un proceso de choque en una descarga eléctrica. La molécula almacena temporalmente energía y la cede en forma de ondas electromagnéticas. La emisión es un fenómeno cuántico y la energía ligada al proceso es tanto mayor cuanto más elevada es la frecuencia, o sea, más corta la longitud de onda emitida"[4].

---

[3]Hecht, Eugene; Zajac, Alfred. Óptica, pág. 350

[4]Sears, ídem, pág. 22

En la actualidad no se reconoce interrupción en el espectro electromagnético, pudiendo producirse y analizarse prácticamente todas las frecuencias del mismo.

1.2.2. EL MECANISMO DE RADIACIÓN

Karttunen explica este proceso de la siguiente manera:

"La radiación electromagnética es emitida o absorbida cuando un átomo o una molécula se mueve de un nivel de energía a otro. Si la energía de un átomo se reduce en una cantidad ΔE, el átomo emite o radia un cuanto de radiación electromagnética, llamado fotón, cuya frecuencia v está dada por la ecuación

$$\Delta E = hv \quad (h = \text{constante de Planck}, h = 6.62620 \text{ Js}).$$

En forma similar, si el átomo recibe o absorbe un fotón de una frecuencia v, su energía se incrementa en $\Delta E = hv$.

Un nivel de energía de un átomo usualmente se refiere al nivel de energía de sus electrones. La energía E de un electrón no puede tomar valores arbitrarios; solo son permitidos ciertos valores: los niveles de energía están cuantizados. Un átomo puede emitir o absorber radiación solamente a ciertas frecuencias v_{if} correspondientes a las diferencias de energía entre los niveles i y f: $|E_i - E_f| = hv_{if}$. Esto origina el espectro de líneas, específico para cada elemento"[5]. De este modo, el espectro obtenido gracias a los cuantos detectados permite conocer el elemento en particular de que han sido emitidos.

1.2.3. EL ESPECTRO DE LÍNEAS

De acuerdo con las condiciones de formación del espectro, pueden distinguirse dos tipos de espectro, Karttunen[6] presenta el siguiente ejemplo para explicarlos:

1.2.3.1. ESPECTRO DE EMISIÓN

Los átomos de un gas caliente regresando de sus estados excitados a los estados más bajos emiten fotones con frecuencias correspondientes a la diferencia de energía de los estados. Cada elemento emite sus propias longitudes de onda características, que pueden ser medidas expandiendo la luz a un espectro con un prisma o una rejilla de difracción.

---

[5]Karttunen, ídem, pág. 108

[6]Karttunen, ídem, pág. 107

1.2.3.2. ESPECTRO DE ABSORCIÓN

Cuando la luz blanca, que contiene todas las longitudes de onda (espectro continuo), pasa a través del gas, las longitudes de onda características del gas son absorbidas.

1.2.4. LAS LÍNEAS ESPECTRALES

Las siguientes características de estas líneas se basan en definiciones de Léna[7]:

1.2.4.1. POSICIÓN

Puede darse la posición en el espectro de una línea específica del mismo por su frecuencia *v* ó *f* (Hz), por su longitud de onda λ, en número de onda $\sigma = 1/\lambda$ (cm^{-1}) o por su energía *hv* (eV).

Las líneas observadas son muchas veces afectadas por los corrimientos Doppler, que en términos de astronomía implica movimientos de determinado tipo del evento emisor, por lo que conocer la frecuencia absoluta (rest) no es suficiente. Se acostumbra indicar la posición observada reducida al LSR (Local Standard of Rest) y medida en kms^{-1}.

1.2.4.2. PERFIL DE LÍNEA

Varios parámetros son usados para representar perfiles complejos del espectro observado de un modo simplificado, uno de ellos es *el ancho total a la mitad del máximo de la altura* o a media profundidad (FWHM: full width at half-maximum; half-depth), que permite determinar un valor del ancho del perfil cuando éste asemeja al que corresponde a un filtro pasabanda y no se tiene un ancho constante.

1.2.5. LOS ESPECTROS CONTINUOS

"Debido al principio de incertidumbre de Heinsenberg, cada línea espectral tiene un ancho natural. Los átomos también tienen movimientos térmicos y los corrimientos de Doppler producidos por esos movimientos aleatorios ensanchan los perfiles de línea"[8].

Las longitudes de onda de cada fotón están definidas, por lo que cada espectro está compuesto básicamente de líneas separadas. Algunas veces, sin embargo, las líneas están tan pegadas y tan anchas, que el espectro parece continuo (sin líneas).

---

[7]Léna, Pierre. Observational Astrophysics, pág. 275

[8]Karttunen, idem, pág. 114

1.2.6. EL ÁTOMO DE HIDRÓGENO

Una de las referencias más empleadas al trabajar con las líneas espectrales es el espectro propio del átomo de hidrógeno, del cual se pueden hacer diversos análisis, que son presentados por Karttunen[9], y que se resumen a continuación:

El átomo de hidrógeno es el más simple y la interpretación mecánica cuántica de Bohr para el mismo indica que se emite radiación solo cuando un electrón salta de un estado de mayor energía a otro de menor energía, teniendo el cuanto emitido una energía *hv*, igual a la diferencia de energía de esos estados:

$$h\nu = E_{n_2} - E_{n_1}$$

El modelo de Bohr, en conjunto con otras relaciones básicas de física, permite encontrar la energía de un electrón en el estado E_n.

$$h\nu = E_{n_2} - E_{n_1} = C\left(\frac{1}{n_1^2} - \frac{1}{n_2^2}\right)$$

donde C es una constante. En términos de la longitud de onda λ, se tiene:

$$\frac{1}{\lambda} = \frac{\nu}{c} = \frac{C}{hc}\left(\frac{1}{n_1^2} - \frac{1}{n_2^2}\right) = R\left(\frac{1}{n_1^2} - \frac{1}{n_2^2}\right)$$

donde R es la constante de Rydberg, $R = 1.097 \times 10^7\ m^{-1}$ y c es la velocidad de la luz, $c=300 \times 10^6$ m/s.

La última expresión fue obtenida experimentalmente por Johann Jakob Balmer en 1885, por lo que el conjunto de líneas espectrales producido por las transiciones $E_n - E_2$ es llamado la serie de Balmer; estas líneas se hallan en la parte visible del espectro. Si el electrón regresa a su estado normal ($E_n - E_1$) se tiene la serie de Lyman, que se encuentra en la zona del ultravioleta.

Las otras series con nombres específicos son la serie de Paschen ($n_1 = 3$), la serie de Brackett ($n_1 = 4$) y la serie de Pfund ($n_1 = 5$). Por razones históricas las líneas de Balmer se denotan usualmente por H_α , H_β , H_γ , etc.

---

[9]Karttunen, ídem, pág. 109

1.3. CONCEPTOS DE ASTRONOMÍA

1.3.1. DENSIDAD DE FLUJO

"La densidad de flujo indica la potencia de radiación por unidad de área y su dimensión es [W m^{-2} Hz^{-1}] ó [W m^{-2}] dependiendo de que se trate de la densidad de flujo a cierta frecuencia o de la densidad total de flujo"[10].

La utilidad de este concepto reside en que sirve para definir parámetros empleados para clasificar a las estrellas, como se presenta en seguida.

1.3.2. MAGNITUDES APARENTES

Uno de los intentos de clasificación de las estrellas visibles fue realizada en el siglo II A.C. por Hipparchos, quien consideró seis clases de acuerdo con el brillo aparente. La primera clase contenía las estrellas más brillantes y la sexta las más borrosas que aún eran visibles para el ojo humano descubierto.

No obstante, la respuesta del ojo humano al brillo de la luz no es lineal, dado que si la densidad de flujo de tres estrellas están en la proporción 1:10:100, la diferencia de brillo entre la primera y la segunda parece ser igual a la diferencia entre la segunda y la tercera. Iguales relaciones de brillo corresponden a iguales diferencias aparentes de brillo: la percepción humana del brillo es logarítmica. La clasificación de Hipparchos fue reemplazada en 1856 por la de Norman R. Pogson, que no se aleja mucho de la anterior. Pogson definió la relación de brillo de las clases n y $n + 1$ como $\sqrt[5]{100} \approx 2.5$.

La clase de brillo o magnitud puede ser definida con exactitud en términos de la densidad de flujo observada F: Si se fija la magnitud 0 como correspondiente a alguna densidad de flujo preseleccionada F_0, todas las demás magnitudes están definidas entonces por la ecuación

$$m = -2.5 \log \frac{F}{F_0}$$

Las magnitudes se extienden en ambos sentidos desde los seis valores originales: La magnitud del Sol, es de hecho negativa, -26.8, y los objetos más borrosos observados tienen magnitudes de alrededor de 25. [11]

---

[10]Karttunen, Idem, pág. 93

[11]Karttunen, Idem, pág. 93

Aunque pueden tomarse diversas referencias para los sistemas correspondientes, actualmente es común considerar el flujo de la estrella Vega como F_0.

1.4. PARÁMETROS EN EL EMPLEO DE DETECTORES

Como se mencionó anteriormente, la mayoría de las observaciones astronómicas utilizan la radiación electromagnética en una forma o en otra, ya que es posible obtener información de la naturaleza física de una fuente de radiación estudiando la distribución de energía de dicha radiación.

1.4.1. LOS FOTONES

La siguiente tabla[12] resume las propiedades de toda la radiación electromagnética. Cada una de estas propiedades proporciona un cierto tipo de información que no puede ser proporcionado por las otras, por lo que cada una tiene una correspondiente estrategia de observación, cuyo análisis particular queda fuera de los propósitos de esta tesis.

Propiedad del fotón	**Estrategia de observación**
Energía, longitud de onda, frecuencia	Tratamiento espectral Transmisión de la atmósfera terrestre Selección del detector adecuado
Número de fotones recibidos (flujo)	Tamaño del área de recepción (telescopios)
Intensidad de radiación	Sensibilidad del detector Fotometría
Dependencia del tiempo ($t \gtrsim 1/\nu$)	Análisis espectral Resolución espectral
Coherencia temporal Dependencia del tiempo ($t \gg 1/\nu$)	Resolución en el tiempo Fotometría rápida ($t \lesssim 1$ s)
Dependencia espacial (angular)	Resolución espacial (angular) del "mapeo" y en la obtención de imágenes
Spin	Polarimetría

[12] Léna, ídem, pág. 7

1.4.2. DETECCIÓN DE FOTONES

Es necesario especificar las condiciones necesarias para que la detección de fotones sea realizada en forma conveniente para los propósitos de observación. Léna presenta los siguientes aspectos:

"La detección de fotones individuales es posible si sus energías exceden varios MeV, i.e., a longitudes de onda más cortas que 200 μm aproximadamente. A longitudes de onda mayores las fluctuaciones térmicas dominan y evitan la detección sencilla de fotones a radiofrecuencias. Es claro que esto no es una limitación física, pero los detectores de radio (y milimétricos y centimétricos) prácticos detectan el campo eléctrico de la onda incidente. Finalmente la detección calorimétrica de la energía radiante es un tercer método que es importante en varias regiones ampliamente separadas: submilimétricas, rayos X y gamma"[13].

1.4.3. LA EFICIENCIA CUÁNTICA

Léna presenta este término de la manera siguiente:

"La superficie de un sólido representa una barrera potencial contra la extracción de electrones del sólido. La altura E_s de la barrera es llamada la función de trabajo de la superficie. Solo los fotones con energía mayor que este umbral fotoeléctrico pueden producir un fotoelectrón. El umbral depende de la estructura cristalina y el estado de la superficie del material. Los umbrales fotoeléctricos en los metales son del orden de 1 eV, lo que limita el uso del efecto fotoeléctrico en vacío en la detección de longitudes de onda menores que aproximadamente un micrón (rayos visibles, UV, X).

Si la energía del fotón está sobre el umbral, la probabilidad de producir un fotoelectrón se conoce como la *eficiencia cuántica* η, definida por

$$\eta = \frac{\textit{número promedio de fotoelectrones producidos}}{\textit{número promedio de fotones incidentes}}$$

Esta probabilidad (por supuesto menor que uno) puede ser determinada teóricamente a partir de la estructura atómica del material o del grosor de la capa emisora"[14].

---

[13] Léna, ídem, pág. 146

[14] Léna, ídem, pág. 146

Cabe añadir que el efecto fotoeléctrico en vacío tiene aplicación en detectores como los fotomultiplicadores, las cámaras electrónicas y los amplificadores de placa microcanal, manejándose en ellos la eficiencia cuántica como un parámetro de desempeño del dispositivo.

1.5. DETECTORES

Hasta fines del siglo XIX, no fue posible obtener más que una pequeña cantidad de información al emplear solo un telescopio para observación. La introducción de detectores semiconductores a mediados de la década de los 70s ha sido una gran ayuda pues su sensibilidad ha mejorado tanto que la capacidad de los instrumentos de observación ha variado en proporción inversa con su tamaño.

1.5.1. LA PLACA FOTOGRÁFICA

La fotografía sigue teniendo un lugar importante en la observación astronómica debido a las diversas ventajas que presenta sobre otros instrumentos. Karttunen menciona algunas de estas características:

"La placa puede registrar millones de estrellas (elementos de la fotografía) de una sola vez, mientras el ojo puede observar un máximo de uno o dos objetos a un tiempo. La imagen de una placa es prácticamente permanente, puede ser estudiada en cualquier momento; además, la placa fotográfica es barata y fácil de usar, comparada con muchos otros detectores. Su característica más importante es su capacidad de recibir luz en un tiempo extenso"[15].

Sin embargo, es necesario mostrar también la principal desventaja que presenta: su baja sensibilidad. La naturaleza de la placa provoca que solo reaccione ante un fotón en un millar; algunos tratamientos químicos pueden sensibilizar la placa antes de la exposición, pero la eficiencia cuántica aumenta muy ligeramente. Otra desventaja es que no permite realizar una segunda exposición, debido a que se alcanza un punto de saturación.

1.5.2. LOS ESPECTRÓGRAFOS

"El espectrógrafo más simple es un prisma que es colocado en frente de un telescopio. Este tipo de dispositivo es llamado un *espectrógrafo de prisma de objetivo.* El prisma separa las diferentes longitudes de onda de la luz en un espectro que puede ser registrado, e.g. en una placa fotográfica. Durante la exposición, el telescopio es

---

[15]Karttunen, ídem, pág. 66

usualmente movido ligeramente en forma perpendicular al espectro, para incrementar el ancho del espectro. Con un espectrógrafo de prisma de objetivo, grandes cantidades de espectros pueden ser fotografiados, e.g. para clasificación espectral"[16].

Existe, además del prisma, la rejilla de difracción, con el que también es posible formar el espectro. Este instrumento presenta ranuras angostas juntas (rayado), cuyo número sirve para identificarlo (se tienen típicamente varios cientos de ranuras por milímetro). Al ser reflejada la luz por las paredes de las ranuras, los rayos contiguos interfieren entre sí para formar distintos espectros. Karttunen abunda en este aspecto:

"Hay dos tipos de rayados: *de reflexión* y *de transmisión*. En un rayado de reflexión, la luz no es absorbida por el vidrio como en el prisma o en el rayado de transmisión. Una rejilla usualmente tiene mayor dispersión, o capacidad para esparcir el espectro, que un prisma. La dispersión puede ser incrementada al incrementar la densidad de las ranuras de la rejilla"[17].

Conviene aquí indicar la definición del colimador, un instrumento que no es precisamente un detector, pero auxilia a algunos de ellos: el colimador es un dispositivo empleado en los aparatos ópticos, por ejemplo en el espectroscopio, para conseguir luz paralela; está compuesto por una lente y por una ranura en el foco de la misma.

1.5.3. LOS FOTOCÁTODOS Y FOTOMULTIPLICADORES

La importancia de analizar las características del fotocátodo reside en que el detector del sistema de interés en el presente trabajo está basado en él, aprovechando las ventajas sobre otros dispositivos, como la placa fotográfica: la eficiencia cuántica del fotocátodo supera entre 10 y 20 veces a la de la placa fotográfica, pudiendo alcanzarse una eficiencia de 30% en el caso óptimo.

El fotocátodo tiene como principio de funcionamiento al efecto fotoeléctrico, que indica que la luz incidente en el instrumento provoca una corriente eléctrica que puede ser medida. Las características del dispositivo en este aspecto también se traducen en ventajas, particularmente de linealidad, pues la corriente aportada por el fotocátodo se duplicará si el número de electrones es duplicado.

Más adelante, al estudiar el sistema Mepsicrón, se proporcionarán otros parámetros sobre el fotocátodo; a su vez, el fotomultiplicador es una importante aplicación del fotocátodo que interviene en forma también decisiva en el sistema mencionado.

---

[16]Karttunen, ídem, pág. 66

[17]Karttunen, ídem, pág. 66

En el fotomultiplicador, los electrones que abandonan el fotocátodo golpean un dínodo (electrodo empleado en los multiplicadores, con alto rendimiento de electrones secundarios), liberando varios electrones de él por cada electrón incidente; así, un conjunto apropiado de dínodos puede aumentar una débil corriente original en un millón de veces, de tal forma que pueda ser medida por un amplificador de carga o electrómetro. La sensibilidad del fotomultiplicador es prácticamente óptima al medir toda la luz que entra a él, aunque sin formar una imagen; se les utiliza con frecuencia en fotometría y llegan a tener una exactitud de 0.1 a 1 %.

"Como usa el efecto fotoeléctrico, no existe umbral de detección (puede detectar un fotón solo) y es perfectamente lineal ante flujos que sean tan grandes que puedan provocar cascadas y destruir el fotocátodo. Su sensibilidad está limitada a la región que comprende desde 20 hasta 12000 nm. por efectos de cascada en los UV lejanos y por la respuesta del fotocátodo en los IR cercanos. Prácticamente no tiene selectividad espectral y su tiempo de respuesta es extremadamente corto (el tiempo de paso de los electrones entre los electrodos es del orden de unos cuantos ns). Es claro que no tiene una capacidad multicanal y la respuesta del fotocátodo podría variar algo a través de su superficie"[18].

Estas propiedades hacen a este dispositivo muy conveniente para la aplicación en la cual se le ocupa, y no solo al compararlo con los anteriores, sino con los que a continuación se presentan.

1.5.4. LOS FOTÓMETROS Y POLARÍMETROS

La siguiente explicación está apoyada en la proporcionada por Karttunen[19]:

Un telescopio recibe la luz del objeto en estudio y ésta entra al fotómetro (detector que mide brillo) a través de un pequeño hoyo en el plano focal, el diafragma, para llegar, a través de un filtro y guiada por una lente de campo, al fotocátodo del multiplicador, en el que cada electrón emitido del cátodo provoca un pulso de alrededor de 10^8 electrones al ánodo; el pulso es amplificado y registrado por un contador de pulsos, para así contar los fotones de la estrella.

Es conveniente agregar que en los diferentes sistemas de observación o detección astronómicos se recurre muchas veces al empleo de un dispositivo llamado *fotopolarímetro*, en el que se usa un *filtro de polarización* ya sea solo o en combinación con otros filtros. El grado y dirección de polarización puede ser hallado midiendo la intensidad de la radiación con diferentes orientaciones de los polarizadores.

---

[18]Léna, idem, pág. 161

[19]Karttunen, idem, pág. 69

1.5.5. DETECTORES BASADOS EN SEMICONDUCTORES

El empleo de detectores fabricados a partir de semiconductores ha crecido desde mediados de los 70s. Con estos detectores es factible conseguir valores del factor de eficiencia cuántica de alrededor de 70 u 80 %, de forma tal que no es sencillo lograr una mejor eficiencia.

Estos dispositivos presentan diversas ventajas ante la placa fotográfica como las regiones de longitud de onda que manejan, que son mucho más amplias que las de la placa; además su respuesta es lineal y, mediante computadoras, los datos de salida disponibles en forma digital pueden ser recibidos, salvados y analizados.

1.5.5.1. INTENSIFICADORES

Los dispositivos conocidos como intensificadores de imágenes se basan en el fotocátodo y en ellos es posible conservar la información sobre el punto de incidencia del electrón en el fotocátodo para posteriormente formar y presentar la imagen intensificada en una pantalla fluorescente; dicha imagen puede ser almacenada entonces por otro medio, por ejemplo, la fotografía.

Entre las ventajas de estos dispositivos se cuenta la capacidad de distinguir aun objetos borrosos con exposiciones relativamente cortas, además, las observaciones pueden hacerse en longitudes de onda en las cuales las placas fotográficas son insensibles.

1.5.5.2. LA CÁMARA VIDICON

Otra aplicación del fotocátodo se halla en detectores cuyo principio de funcionamiento es similar al de la cámara de TV. Los electrones que son recibidos a la salida del fotocátodo son acelerados para después incidir en un electrodo y así lograr la imagen con forma de una distribución de carga eléctrica. Posteriormente se lee la carga en diferentes puntos en la superficie del electrodo mediante un barrido (scanning) con un rayo de electrones, renglón por renglón.

El resultado es una señal de video que puede ser finalmente vista empleando un tubo de TV, o bien, salvada digitalmente. En los sistemas más avanzados se realiza un registro, en una memoria de computadora, de la información de centelleos causados por electrones solitarios en la pantalla fluorescente del intensificador de imágenes. Un procedimiento similar se sigue en este proyecto y será detallado en el capítulo cuarto.

1.5.5.3. EL CCD

El CCD (Charge Coupled Device, dispositivo acoplado por carga) es un circuito integrado de silicio capaz de detectar luz, compuesto por un arreglo de elementos con propiedades capacitivas. Cuando la imagen de un objeto celeste es detectada por el CCD,

la luz se transforma en señales eléctricas. El nombre de este dispositivo describe la forma como la señal del mismo es leída (la distribución de cargas acumuladas es leída por la transferencia de las mismas a lo largo del arreglo).

El CCD se aplica en el procesamiento de imágenes y en sistemas de procesamiento digital de señales, donde se aprovecha la naturaleza "serial" de alta densidad (high-density serial nature, hace referencia a características de construcción) del dispositivo. La aplicación más sencilla de un CCD es una cámara para la formación de imágenes en el foco de un telescopio; no obstante, los CCDs tienen muchas ventajas sobre los otros detectores electrónicos y fotográficos, entre las que se encuentran las siguientes:

Son dispositivos muy pequeños, compactos, de baja potencia y estables en un amplio intervalo de longitudes de onda; esto significa una gran cobertura del espectro, gracias a que el material de silicio de un CCD es sensible a la luz de una gran variedad de colores. Tienen excepcional linealidad y no existe dispersión en la imagen, ya que todos los pixeles en el silicio están en posiciones ordenadas por construcción. Presentan muy alta sensibilidad: una enorme fracción de la luz que cae sobre el CCD se convierte en un voltaje eléctrico medible. Tienen compatibilidad inmediata con las computadoras: una salida del CCD es una corriente de pulsos eléctricos listos para transferirse a una computadora y desplegar instantáneamente la imagen.

En el presente proyecto se ha considerado el empleo de los fotomultiplicadores, que son más económicos que los CCDs[20] y presentan las características suficientes para un buen desempeño en el sistema. En el siguiente capítulo se hablará comparativamente de las características de ambos.

1.5.6. INTERFERÓMETROS

La resolución de un telescopio grande está limitada en la práctica por seeing y, por tanto, incrementar la apertura no mejora necesariamente la resolución. Se pueden usar diferentes interferómetros para obtener una resolución más cercana al límite teórico fijado por la difracción.

Hay dos tipos de interferómetros ópticos. Un tipo usa un gran telescopio existente; el otro, un sistema de dos o más telescopios separados. En ambos casos, a los rayos de luz se les permite interferir. Mediante el análisis del patrón de interferencia resultante, pueden ser estudiadas las estructuras de binarias cercanas, pueden ser medidos los diámetros angulares aparentes de las estrellas, etc.

---

[20]En 1996, el costo aproximado del mejor CCD era US $40, 000

CAPÍTULO SEGUNDO

EL SISTEMA MEPSICRÓN

2.1. DESCRIPCIÓN GENERAL DEL EQUIPO

A partir de los reportes técnicos del Instituto de Astronomía de la UNAM [[1] y [2]], se obtuvieron los datos siguientes sobre el funcionamiento del sistema contador de fotones bidimensional que posee resolución temporal, denominado Mepsicrón:

Al incidir un fotón sobre el fotocátodo multialcalino, se desprende un electrón que será acelerado por un campo eléctrico externo, haciéndolo llegar a las placas microcanal, que están formadas por microcanales capilares multiplicadores de electrones: cada uno consiste en un tubo de vidrio en una inclinación tal que asegura la colisión del electrón contra su pared, de la que se desprenden electrones secundarios, los cuales a su vez repiten el proceso creando un haz de aproximadamente 10^5 electrones que, al salir por la última placa, choca contra el ánodo resistivo, lo que establece un paquete de carga lo suficientemente grande para poder ser detectado.

Es importante hacer notar que el centroide de masa de la nube electrónica producto del electrón entrante conserva la posición del punto de impacto del fotón sobre el fotocátodo.

El Mepsicrón no ha tenido cambios importantes desde 1985 y en seguida se describe su estado al principio del presente trabajo de tesis:

El sistema contiene tres elementos básicos:

- el detector
- un instrumento optoelectrónico en donde se analiza la posición y el tiempo de arribo de cada evento (el analizador de posición de pulsos o APP)
- y una memoria en donde se guarda la información.

Existe además un paquete de software con el que se maneja la información que el sistema va acumulando. En 1985 el APP analizaba hasta 30 000 eventos por segundo y se esperaba tener una capacidad diez veces superior.

---

[1] Bohigas, J. Reporte técnico No. 19

[2] Ruiz, E. Martín; Iriarte, Arturo. Reporte técnico No. 21

La figura 1 es una representación simplificada del detector. Sus componentes principales son descritos a continuación.

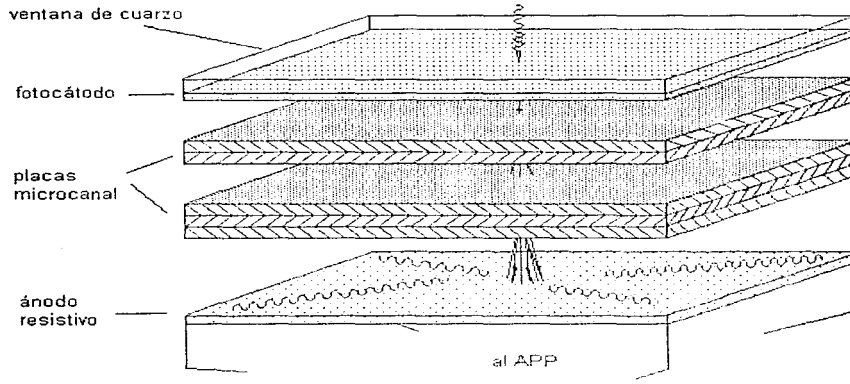


Figura 1. Representación esquemática del detector

2.1.1. EL FOTOCÁTODO

Es un elemento que transforma algún fotón incidente en un electrón. Actualmente se utiliza un fotocátodo multialcalino S25 con extensión al rojo depositado sobre una placa de cuarzo, y en la figura 2 se muestra su curva de respuesta. Como se puede ver, su máxima eficiencia cuántica es del 24% en 4000 Å, decayendo al 14% en 3000 Å y al 3.6% en H_α. En el fotocátodo están contenidas dos de las limitaciones más importantes del sistema:

- Por una parte, los fotocátodos son muy poco sensibles hacia el rojo, de modo que el sistema es inservible a longitudes de onda mayores que 9000 Å.
- En segundo lugar, es posible que una zona del detector quede temporalmente "ciega" al recibir una señal demasiado intensa. Por ejemplo, trabajando con un filtro de banda ancha (50 Å centrado en 4670 Å), se encontró que una estrella de m=7.5 dejó una mancha sobre la superficie del detector durante unos días.

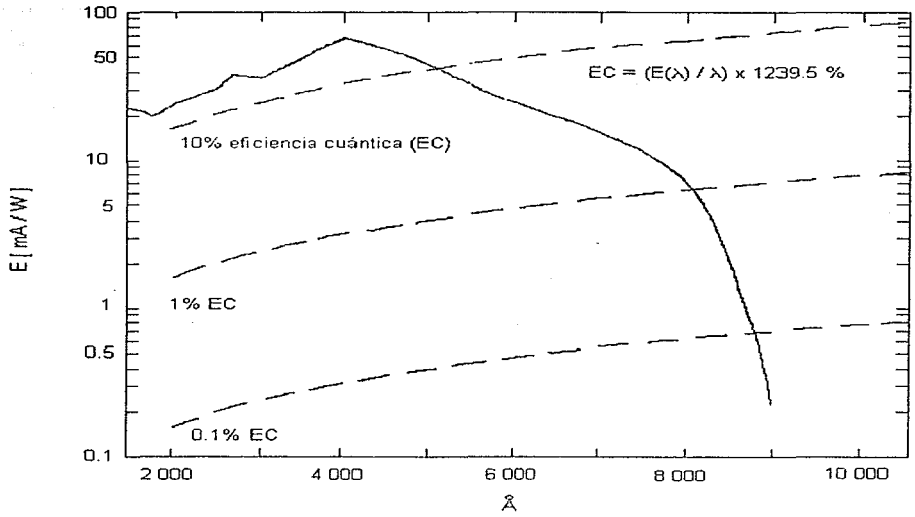


Figura 2. Curva de respuesta espectral del fotocátodo del detector

2.1.2. EL SISTEMA DE PLACAS MICROCANAL

Este sistema está compuesto por una serie de 5 placas microcanal con una configuración V-Z y convierte al electrón que emerge del fotocátodo en una cascada de aproximadamente 10^8 electrones.

2.1.3. EL ÁNODO RESISTIVO

El ánodo resistivo de baja distorsión es una placa de mediana resistividad que tiene en sus cuatro extremos conexiones para poder instalar un circuito electrónico que decodifique, a partir de las corrientes eléctricas que salen del área de incidencia de la nube electrónica (producida por el sistema de placas microcanal) sobre él, para lo que es necesario sensar la carga en cada una de las esquinas y preamplificarlas a niveles de voltaje reconocibles por la etapa subsecuente del sistema de detección de imágenes.

2.2. CARACTERÍSTICAS BÁSICAS DEL DETECTOR

En la tabla siguiente se resumen las principales características del detector.

Tamaño	25x25 mm
Número de pixeles	$8x10^5$ (cada pixel mide 25 µm)
Resolución espacial	42 µm a FWHM
Resolución temporal	200 ns
Ruido de fondo	$6x10^{-5}$ conteos / (pixel seg) a -30 °C
Máximo número de conteos	100 conteos / (pixel seg)
Rango dinámico	$\geq 10^6$
Ganancia	10^5 electrones / evento
Fotocátodo	S25 extendido al rojo
Variación de sensibilidad de pixel a pixel	≤5%
Corrección por linealidad	Uniforme
Observación en tiempo real	Sí
Límite a objetos brillantes	Por el fotocátodo: m= 7.5 de baja dispersión o en imagen directa. Por el APP: m= 8.
Tiempo de exposición	Espectroscopía: m=16 en 1 hora, con señal a ruido = 5 y 0.4 Å / FWHM. Imagen directa: m=21 en 10 minutos, con filtros de 80 Å de ancho de banda.

2.3. OBSERVACIONES

2.3.1. EL INTERVALO DINÁMICO

El intervalo dinámico está definido por el contraste entre el ruido de fondo y el máximo número de conteos que es capaz de asimilar el detector. Un elemento (pixel) del detector solo puede contar un número limitado de eventos por segundo, ya que usa un

tiempo definido en restablecer la carga del microcanal empleado, durante el cual se halla deshabilitado. Así, el detector no puede observar señales puntuales demasiado intensas.

Vale la pena señalar que la amplitud del intervalo dinámico ofrece la posibilidad de estudiar cocientes de líneas de emisión sumamente contrastadas, en particular las que son sensibles a la temperatura.

2.3.2. LIMITACIÓN POR EL APP

Ya se han mencionado dos limitaciones a señales muy intensas: el propio fotocátodo y el tiempo de respuesta de cada uno de los pixeles del detector. Una tercera limitación se halla en el APP por el que, de no poder analizar más de 30,000 eventos por segundo, no se observarán estrellas más brillantes que magnitud 8. Un APP 10 veces más rápido posibilitará la observación de estrellas de hasta magnitud 5, siempre y cuando no se presente el problema del fotocátodo.

2.3.3. RECEPCIÓN DE SEÑALES DÉBILES

La experiencia ha demostrado que en espectroscopía de alta dispersión (alta resolución espectral, mayor número de líneas en el espectro) toma una hora observar una estrella de magnitud 16 (en el máximo de la respuesta instrumental, con una señal a ruido igual a 5 y 0.4 Å FWHM). En imagen directa aparecen estrellas de magnitud 21 ó 21.5 en 10 minutos, utilizando filtros ópticos de 80 Å de ancho de banda.

2.3.4. EL VIÑETEO

El término *viñeteo* se refiere a un efecto similar a la difracción provocado por un obstáculo entre el evento de interés y el detector. Este efecto es fuerte en el detector, ya que su sensibilidad disminuye hacia las orillas; más aún, varía con la longitud de onda, siendo más pronunciado en el rojo que en el azul. Es probable que se deba a que el depósito de diversos materiales sobre la superficie del fotocátodo no es uniforme.

2.4. DESCRIPCIÓN DE LA ELECTRÓNICA

La figura 3 es un diagrama a bloques del sistema. De atenerse a la señal que emerge del ánodo resistivo, pueden distinguirse las siguientes etapas en el proceso de adquisición de datos:

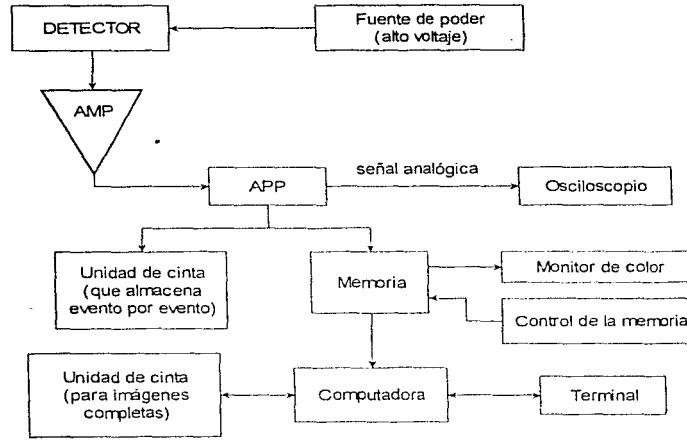


Figura 3. Diagrama a bloques del sistema

i Al salir del detector, la señal pasa por varias fases de amplificación hasta llegar al APP, sitio en donde es analizada y codificada la posición y el tiempo de arribo de cada evento.

ii Del APP sale una señal al frecuencímetro, aparato en donde se registra el número de detecciones por unidad de tiempo. Con la electrónica en 1985 no era posible analizar más de 30,000 eventos por segundo. De hecho, era recomendable no trabajar a más de 20,000 cuentas por segundo, y si esta condición no se verificaba, era necesario reducir el número de eventos que llegaban al detector, con métodos tales como cambiar la rendija que limita la luz que llega al detector o diafragmando el telescopio.

La rendija actúa conjuntamente con mascarillas para controlar la forma en que es recibida la luz. Cerrando la rendija se tiene mayor definición de líneas espectrales, aunque se recibe menor cantidad de luz.

iii El APP genera una señal analógica que va a dar a un osciloscopio colocado en la misma platina que el detector. Ahí se puede llevar a cabo un análisis inmediato sobre el funcionamiento del sistema.

iv El APP también produce una señal digital que es enviada a las memorias encontradas en la sala de cómputo.

v Una de las memorias era una cinta magnética que guardaba la información acerca de la posición y el tiempo de arribo de cada evento. Esta forma de guardar la información puede ser útil en relación con problemas de señales periódicas de alta frecuencia.

vi También es posible contabilizar el número de conteos recibidos por cada pixel del detector, y guardar esta información en una memoria adecuada. En este caso se pierde la información temporalmente. Esta memoria cuenta con 1024x1024 elementos, cada uno de ellos con una profundidad de 16 bits. Es decir, la memoria puede guardar hasta 2 MB de información, y cada uno de sus elementos puede registrar hasta 64,000 conteos.

vii La memoria está conectada a un monitor de color y a un panel de control. En el monitor se observa en tiempo real la imagen que se está formando en la memoria. Con el panel de control se pueden realizar diversas operaciones:

- iniciar o interrumpir la integración
- decidir si se corta o no el último bit significativo
- escoger uno de los cuatro cuadrantes en los que se puede fraccionar la memoria (en caso de cortar el último bit significativo)
- amplificar la imagen
- colocar el cursor

en fin, llevar a cabo tareas de análisis sobre la imagen, al mismo tiempo que ésta se está formando.

El sistema Mepsicrón prácticamente termina aquí, era entonces necesario vaciar los datos acumulados en un registro más permanente y móvil, como por ejemplo una cinta. Hecho esto se vaciaría la memoria, que de esta manera estaría en condiciones de recibir nuevos datos.

viii La transferencia de datos entre la memoria y la unidad de cinta se realiza a través de una computadora, con la que también es posible manipular los datos que están siendo recibidos o que se hayan recibido en el pasado. Para llevar a cabo estas tareas la computadora está conectada con las secciones siguentes:

a) La memoria y su panel de control
b) Terminal ADM, con su control de cursor
c) DATARAM, que es la memoria asociada a la computadora

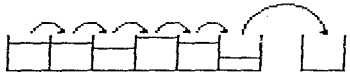
En el capítulo cuarto son descritos los cambios hechos a esta configuración, basados en el empleo de los DSPs, de los cuales se exponen diversos detalles en el tercer capítulo.

2.5. VENTAJAS DEL MEPSICRÓN SOBRE EL CCD

A continuación se realiza una evaluación cualitativa de las diferencias entre el CCD y el Mepsicrón y se exponen las ventajas de este último.

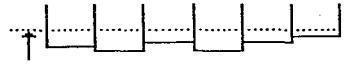
Es posible hacer una analogía de los CCDs con un recipiente para cubos de hielo. Los "pozos" o pixeles (con áreas típicas de 7, 18 ó 21 μm^2; altamente densos comparados con, por ejemplo, los detectores infrarrojos, de 40 μm^2) se llenan con las partículas (fotones) que llegan a ellos.

La información (cantidad contenida) en cada pozo es "leída" pixel por pixel y renglón por renglón, de la siguiente manera: Se hace un corrimiento de pixeles con la ayuda de un registro, de tal forma que los datos son transmitidos en serie pixel por pixel; al terminar un renglón se pasa al siguiente para hacer así el barrido de todo el arreglo. El tiempo empleado en dicho barrido o lectura es solo una parte del tiempo total de proceso en el CCD.



El último es un registro auxiliar o de corrimiento

Al no existir arreglos de pixeles ideales, los pozos son, en general, de diferente profundidad, por lo que se hace una "lectura de sesgo" (bias) como parte de la inicialización del CCD. Consiste en llenar todos los pozos hasta un cierto nivel, i.e., hacer una exposición uniforme durante un tiempo determinado, para guardar posteriormente su lectura.

Nivel de bias 

Como es fácil ver, al hacer después una exposición cualquiera, bastaría restar la exposición de bias para superar el obstáculo de diferencias de profundidad.

Otro problema consecuencia de la no idealidad de los pixeles es la diferencia de sensibilidad entre ellos, que es resuelto con una exposición de campo plano (flat field). A semejanza de lo que ocurre con la exposición de bias, la exposición final deberá después confrontarse con la de campo plano (no en una diferencia, sino en convolución), para hallar la imagen real.

El tiempo de proceso incluye entonces al de barrido, al de resta con la exposición de bias y al de convolución con la exposición de campo plano.

Durante el tiempo de lectura el obturador del CCD debe permanecer cerrado, pues puede verse que la recepción de fotones en ese momento alteraría la información recibida. Ésta es una limitante del CCD: no pueden hacerse lecturas en tiempo real; es necesario esperar a hacer el barrido de los pixeles para tomar después la siguiente lectura. Por tal motivo, es conveniente que el tiempo de barrido sea el menor posible.

Hay modelos recientes de CCDs que, a semejanza de las cámaras de TV, no "barren" pixel por pixel directamente, sino que pasan el cuadro completo a un segundo arreglo en un tiempo mínimo y así barren este segundo arreglo mientras el primero sigue recibiendo fotones. El tiempo de exposición en este caso no puede ser menor que el tiempo de barrido.

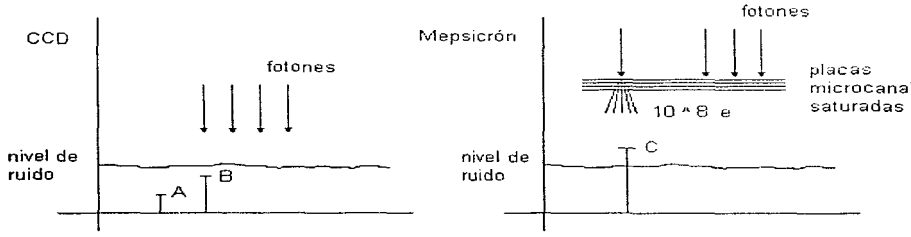
El CCD integra imágenes y la mínima señal observable es aquella que tenga un nivel superior al del ruido. El ruido asociado al CCD es causado, entre otras cosas, por la denominada "corriente oscura", propia del detector. Esta corriente se define como la señal obtenida en el CCD no expuesto a la luz, i.e., sin recibir fotones, debida a la temperatura ambiente, a los rayos cósmicos y al efecto de "bias".

La imagen de bias (o sesgo) del arreglo de un CCD es la diferencia de voltaje con respecto a 0 V de cada pixel del arreglo sin luz en el arreglo inmediatamente después de que es limpiado.[3]

---

[3] Spectra Source Instruments. LYNXX/LYNXX PLUS CCD Digital Imaging System User's Manual; pág. 3-16.

El Mepsicrón no integra sino que cuenta fotones, en él la corriente oscura es mínima y, por tanto, también el ruido provocado por ella; no obstante, sí tiene un nivel de ruido causado por la incertidumbre de localización de fotones (llamado "ruido de fotones") y por la temperatura a la que se encuentre (ruido térmico o de Johnson).



A Amplitud "lograda" por un fotón. No es observable debido a que es menor que el nivel del ruido.
B Amplitud "lograda" al integrar varios fotones. Tampoco observable.
C Amplitud detectable, superior al nivel de ruido.

Cuando se logra una amplitud observable, se ha integrado ya muchas veces, ha pasado ya un cierto tiempo y, además, se ha integrado involuntariamente la corriente oscura (el CCD es un integrador de corriente), lo que propicia la saturación del detector.

Un fotón no llega al detector del Mepsicrón directamente, sino que es antes amplificado por un factor de hasta 10^8 por el fotomultiplicador, cantidad que sí producirá un nivel superior al del ruido y, por tanto, observable. De este modo podría concluirse que el Mepsicrón sí detecta fotones individuales y cambiando la referencia de voltaje puede asumirse que no tiene ruido (la relación señal a ruido es muy alta, como se ha visto). La ventaja en velocidad ante el CCD es evidente, solo puede hablarse de tiempo real en el Mepsicrón.

CAPÍTULO TERCERO

LOS PROCESADORES DIGITALES DE SEÑALES

Los procesadores digitales de señales, comúnmente conocidos como DSPs, han tenido un avance muy rápido en lo que a desarrollo y mejoras corresponde, dada la gran cantidad de aplicaciones que los requieren. Los DSPs son ciertamente microcontroladores con un uso específico: el manejo de señales digitales; esto se entiende al observar la estructura de los DSPs y compararla con los microcontroladores, pues ambos poseen una unidad aritmética lógica, bloques de memoria RAM y ROM, buses[1] para direcciones y para datos, puertos de comunicación con otros dispositivos y demás características o elementos propios de dichos sistemas.

La información que a continuación se presenta, referente a datos técnicos, características de operación y demás especificaciones acerca de los sistemas desarrollados por Texas Instruments ha tenido como fuente principal sus manuales [2 y 3], a los que será necesario acudir en el caso de requerirse mayores detalles sobre algún aspecto en particular, para efecto de lo cual se han hecho las indicaciones convenientes, ya que este capítulo no pretende un análisis exhaustivo sobre el sistema constituido por los DSPs, sino una descripción general que, si bien amplia, solo profundiza donde los requerimientos del proyecto en cuestión lo han hecho necesario. Además, es necesario mencionar que en el desarrollo del capítulo se indican, donde se ha considerado necesario, en notas al pie de página o entre paréntesis, los términos originales y las acepciones manejadas en este texto.

Aunque son disponibles comercialmente DSPs de punto fijo y de punto flotante, existen muchas restricciones en cuanto a la aplicación del grupo de punto fijo, cuya principal ventaja es, no obstante, su costo menor que los restantes. En cuanto a los mencionados en segundo término (los de punto flotante), conviene indicar que la característica del punto flotante manejada por los DSPs en este proyecto queda definida de la siguiente manera:

Exponente	8 bits
Signo	1 bit
Fracción	31 bits

---

[1] Buses; ductos. Se manejará el término original.

[2] Texas Instruments. TMS320C4x User's Guide; abreviado como UG.

[3] Texas Instruments. TMS320C4x Parallel Processing Development System, Technical Reference; abreviado como PPDSTR.

3.1. EL PROCESAMIENTO PARALELO

La necesidad de procesamiento paralelo está creciendo rápidamente. Al tiempo que los requerimientos de trabajo con punto flotante crecen exponencialmente, los fabricantes de semiconductores no logran encontrar más necesidades de elementos de procesamiento sencillo. Los procesadores no diseñados para procesamiento paralelo se vuelven inadecuados porque la comunicación entre procesadores rápidamente satura el dispositivo de I/O (entrada/salida) y afecta contrariamente la eficiencia de cómputo.

Los productos en la generación C3x de Texas Instruments comenzaron a mostrar la necesidad de procesamiento paralelo al proveer a los diseñadores dos puertos externos de interfaz, lo que se traduce en una cantidad inmensa de ancho de banda de I/O.

Los dispositivos de la generación C4x van varios pasos más allá al incorporar hardware integrado para facilitar la comunicación de alta velocidad entre procesadores además de operación I/O concurrente (simultánea) sin disminuir el desempeño del CPU. Estas características, en conjunto con un "anfitrión" (host) de sofisticadas herramientas de desarrollo de procesamiento paralelo, hacen a la generación de procesadores de punto flotante ideal para aplicaciones en tiempo real.

El sistema de procesamiento paralelo está formado por un conjunto de cuatro DSPs TMS320C40; esto es, pertenecientes a la familia de procesadores de punto flotante.

El empleo de este tipo de hardware requiere también un sistema operativo particular, capaz de realizar procesos multitareas (multitasking), para lo cual el fabricante recomienda el denominado OS/2 de la casa IBM. En realidad, el DSP TMS320C40 puede operar confiablemente en forma individual con sistemas operativos más comunes, como DOS o Windows, pero no es el caso de un sistema trabajando con múltiples DSPs simultáneamente. [*]

3.2. LA ARQUITECTURA DEL TMS320C40

La generación de procesadores de punto flotante TMS320C4x está diseñada específicamente para resolver las necesidades de procesamiento paralelo y otras aplicaciones del tipo de tiempo real.

El fabricante menciona como las características principales del procesador paralelo TMS320C40 las siguientes:

- Seis puertos de comunicaciones (com0 - com5) para comunicación de alta velocidad entre los procesadores.

---

*Véase el apéndice A para detalles

PAGINACION VARIA

COMPLETA LA INFORMACION

- Un coprocesador DMA (direct memory access) de seis canales para operación I/O concurrente (simultánea) y del CPU, maximizando el desempeño aportado por el CPU al aliviar a éste de operaciones difíciles.
- Un CPU de alto desempeño capaz de manejar 320 Mbytes por segundo y 275 MPOS (millones de operaciones por segundo).
- Dos buses idénticos de datos externos y direcciones que manejan sistemas de memoria compartida y transferencias de ciclo simple y alto índice de datos.
- Programa de caché[5] contenido en el chip y RAM de acceso dual y ciclo simple para desempeño incrementado de acceso de memoria.
- Buses separados para programas internos, datos y el DMA para manejo de una capacidad de transferencia I/O de programas y datos masiva y concurrente.
- Interfaz JTAG para conexión a sistemas estándares.
- Un módulo de análisis contenido en el chip que maneja el más novedoso depurador de proceso paralelo. [6]

3.2.1. CPU

El C4x tiene una arquitectura de CPU basada en registros. Dicho CPU está compuesto por los siguientes elementos:

- Un multiplicador de enteros (32 bits) y de punto flotante (40 bits).
- Una ALU (unidad aritmética lógica) para llevar a cabo las operaciones aritméticas: de punto flotante (40 bits), enteras (32 bits) y lógicas (32 bits).
- Un cambiador de barril de 32 bits[7].
- Buses internos.
- Unidades aritméticas de registros auxiliares (ARAUs).
- Un archivo de registros de CPU [8].

---

[5] Memoria oculta; se manejará el término común caché

[6] UG, pág. 1-4, puede analizarse el diagrama a bloques del TMS320C40 en el apéndice B

[7] 32-bit barrel shifter

[8] UG, pág. 2-6

3.2.1.1. REGISTROS DEL CPU

La descripción de los registros que a continuación se hace tiene la intención de permitir una mejor comprensión de temas vistos más adelante.

- *Los registros de precisión extendida (R0 - R11)* son capaces de guardar y soportar operaciones de números enteros de 32 bits y de punto flotante de 40 bits. Cualquier instrucción que suponga que los operandos son números de punto flotante usa los bits de 39 a 0. Si los operandos son enteros con ó sin signo, solo son usados los bits de 31 a 0 y los restantes permanecen sin cambio.

- *Los registros auxiliares de 32 bits (AR0 - AR7)* pueden ser accesados por el CPU y modificados por las dos unidades aritméticas de registros auxiliares (ARAUs). La principal función de los registros auxiliares es la generación de direcciones de 32 bits. También pueden ser usados como contadores de lazo (loop counters) o como registros de propósito general de 32 bits que pueden ser modificados por la ALU y el multiplicador.

- *El apuntador a página de datos* (data-page pointer, *DP*) es un registro de 32 bits cuyos 16 LSBs son usados por el modo de direccionamiento directo como un apuntador a la página de datos que está siendo direccionada. El C4x puede direccionar hasta 64 "kpáginas", cada página conteniendo 64 "kpalabras".

- *Los registros índice de 32 bits (IR0, IR1)* contienen el valor usado por la ARAU para procesar una dirección indexada.

- *El registro de tamaño de bloque (BK)* es empleado por la ARAU en direccionamiento circular para especificar el tamaño del bloque de datos.

- *El apuntador a la pila del sistema* (stack pointer, *SP*) es un registro de 32 bits que contiene la dirección de la cima de la pila del sistema. El SP siempre apunta al último elemento introducido en la pila.

- *El registro de estado (ST)* contiene información global relacionada con el estado del CPU

- *El registro de bandera de IIOF (IIF)* controla la función (I/O de propósito general o interrupción) de los cuatro pines externos (IIOF0 a IIOF3). Las interrupciones pueden ser dispuestas por nivel o por flanco. También contiene las banderas de interrupción del timer y del DMA.

- *El registro de habilitación de interrupciones al coprocesador DMA (DIE)* es un registro de 32 bits que contiene campos de 2 y 3 bits para designar el esquema de sincronización de interrupciones para cada uno de los canales del DMA.

- *El registro de habilitación de interrupciones internas del CPU (IIE)* es también un registro de 32 bits que habilita o deshabilita las interrupciones para los seis puertos de comunicaciones, ambos timers[9] y los seis canales del coprocesador DMA.

- *El registro contador de repetición (RC)* de 32 bits especifica el número de veces que un bloque de código va a ser repetido cuando se ejecuta una repetición de bloque.

- *El contador de programa (PC)* es un registro de 32 bits que contiene la dirección de la instrucción próxima a ser traída[10].

- *El registro IVTP* es un registro especial que apunta a la tabla del vector de interrupciones (IVT).

- *El registro TVT* es un registro especial que apunta a la tabla del vector de trampas (traps) (TVT). [11]

3.2.2. MEMORIA

3.2.2.1. LA ORGANIZACIÓN DE LA MEMORIA

El alcance total de la memoria del C4x es 4 G (gigas o miles de millones, 1 G = 2^{30}) de palabras de 32 bits (16 Gbytes). Dentro de este espacio están contenidos tanto la memoria de programa (RAM o ROM integrada, on-chip, y memoria externa) como los registros que afectan a los timers, los puertos de comunicaciones y los canales de DMA. Esto permite que tablas, coeficientes, código de programa y datos sean guardados ya sea en RAM o en ROM. De este modo, el uso de memoria es maximizado y el espacio de memoria asignado[12] como se desee.

Mediante la manipulación de un pin externo (ROMEN), es posible configurar la primer área de memoria de un megaword (0000 0000h a 00FF FFFFh) para ser parte del bus local de direcciones o para direccionar la ROM integrada cuando sea usado el cargador del boot (con el espacio restante reservado)[13].

---

[9]A los timers también se les llama temporizadores o relojes. Se manejará el término original.

[10]Fetch; traer, buscar

[11]UG, pág. 2-6 y 3-3

[12]Allocated; asignado, alojado, "direccionado"

[13]Puede consultarse la referencia UG y la figura de mapas de memoria en el apéndice B

3.2.2.2. RAM, ROM y caché del TMS320C4x

Los bloques de RAM 0 y 1 son de 4 kbytes (1 k x 32 bits) cada uno. El bloque de ROM está reservado y contiene un cargador de boot. Cada bloque de RAM y ROM es capaz de soportar dos accesos en un solo ciclo. Los buses independientes de programa, datos y DMA permiten traídas[14] de programa paralelas, lecturas y escrituras de datos y operaciones de DMA. Por ejemplo: el CPU puede accesar dos valores de datos en un bloque RAM y realizar una búsqueda externa de programa en paralelo con el coprocesador DMA cargando otro bloque RAM, todo dentro de un solo ciclo.

El bloque ROM reservado contiene un cargador de boot, que soporta cargas de programa y datos al momento del reset, como ya se había indicado en el inciso anterior. La carga es de las memorias de 8, 16 ó 32 bits o de alguno de los seis puertos de comunicaciones.

Un caché de instrucciones de 128 x 32 bits se provee para guardar secciones de código frecuentemente repetidas, reduciendo así en gran medida el número de accesos fuera del chip necesarios. Esto permite al código ser almacenado en memorias, más pequeñas y de menos costo, fuera del chip. Los buses externos están también liberados para ser usados por el DMA, traídas de memoria externas o por otros dispositivos en el sistema[15].

3.2.2.3. MODOS DE DIRECCIONAMIENTO DE LA MEMORIA

Se dispone de cuatro grupos de modo de direccionamiento en el C4x, cada uno de los cuales tiene dos o más diferentes tipos de direccionamiento, como se indica a continuación:

- *Modos de direccionamiento general:*
  - i Registro: el operando es un registro de CPU.
  - ii Inmediato: el operando es un valor "inmediato" de 16 bits.
  - iii Directo: el operando es el contenido de una dirección de 32 bits (una concatenación de los 16 bits del apuntador de página de datos y un operando de 16 bits).
  - iv Indirecto: Un registro auxiliar de 32 bits indica la dirección del operando.

- *Modos de direccionamiento de 3 operandos:*
  - i Registro (el mismo que en el modo de direccionamiento general).
  - ii Indirecto (el mismo que en el modo de direccionamiento general).
  - iii Inmediato: el operando es un valor "inmediato" de 8 ó 16 bits.

---

[14]Fetches; traídas, búsquedas

[15]Puede consultarse la figura de organización de la memoria en el apéndice B

- *Modos de direccionamiento paralelo:*
  i Registro (el mismo que en el modo de direccionamiento general).
  ii Indirecto (el mismo que en el modo de direccionamiento general).

- *Modos de direccionamiento de saltos:*
  i Registro (el mismo que en el modo de direccionamiento general).
  ii Relativo al PC (contador de programa): Un desplazamiento de 16 bits con signo ó un desplazamiento de 24 bits es añadido al PC. [16]

3.2.3. INTERRUPCIONES

El C4x maneja las siguientes interrupciones:

- Cuatro interrupciones externas (IIOF3 - 0)
- Un número de interrupciones internas
- Una interrupción externa "no mascarable" NMI
- Una señal externa no mascarable RESET, que lleva el procesador a un estado conocido.

El DMA y los puertos de comunicaciones tienen sus propias interrupciones internas. [17]

3.2.4. PERIFÉRICOS

Todos los periféricos del C4x están controlados a través de registros "mapeados" en memoria en un bus periférico dedicado. Este bus periférico permite una comunicación directa con los periféricos. Los periféricos del C4x incluyen dos timers y seis puertos de comunicaciones. [18]

3.2.4.1. LOS PUERTOS DE COMUNICACIONES DE LA GENERACIÓN TMS320C4x

El capítulo 8 del manual UG analiza con detalle los puertos del C4x, en tanto que en el apéndice B se presenta más información con respecto a datos técnicos; a continuación se indican características de interés de los mismos: Para proporcionar la

---

[16]UG, pág. 2-15

[17]UG, pág. 2-28

[18]UG, pág. 2-29; puede consultarse la figura de módulos periféricos en el apéndice B

comunicación sencilla entre procesadores, el C4x tiene seis puertos de comunicación paralela bidireccional de alta velocidad, cuyas características se enuncian a continuación:

- Operaciones de transferencia bidireccional de datos (a un tiempo de ciclo de 40 ns) de 160 MBps (20 Mbytes o 5 Mwords por segundo).
- Comunicación simple entre procesadores vía ocho líneas de datos y cuatro líneas de control.
- Memoria intermedia (buffering) de tipo FIFO de todas las transferencias de datos, tanto entrada como salida.
- "Arbitraje" automático para asegurar la sincronización de la comunicación.
- Sincronización entre el CPU ó el coprocesador de acceso directo a memoria (DMA) y los seis puertos de comunicaciones vía interrupciones internas y señales de disponibilidad internas (ready signals).

Ya que se tienen unidades de arbitraje de puertos en ellos para manejar la pertenencia del bus de datos de los puertos de comunicaciones entre los procesadores, es conveniente atender sólo la operación interna de dichos puertos. Para software, pueden ser tratados como buffers integrados (on-chip) de 32 bits de datos I/O FIFO (First In-First Out). La lectura o escritura de datos del procesador para comunicación es simple:

```
LDI    @comm_port0_input, R0          ; Lee datos del puerto de com. 0
o
STI    R0, @comm_port0_output         ; Escribe datos en el puerto de com. 0
```

Si el CPU o el DMA lee o escribe en el puerto de comunicaciones I/O FIFO y el I/O FIFO está ya vacío (en una lectura) o lleno (en una escritura), la ejecución de la lectura o escritura será extendida ya sea hasta que el dato esté disponible en la entrada FIFO para una lectura o hasta que el espacio esté disponible en la salida FIFO para una escritura. Algunas veces puede usarse esta característica para sincronizar los dispositivos. De todos modos, esto puede ser atrasar la velocidad de procesamiento y aun "colgar", detener, el procesador.

Se previenen tales situaciones mediante la sincronización de los accesos CPU/DMA con las siguientes banderas que indican el estado del puerto:

ICRDY (input channel ready)
=0, el canal de entrada está vacío y no está listo para ser leído
=1, el canal de entrada contiene datos y está listo para ser leído.

ICFULL (input channel full)
=0, el canal de entrada no está lleno
=1, el canal de entrada está lleno

OCRDY (output channel ready)
=0, el canal de salida está lleno y no está listo para escribir en él
=1, el canal de salida no está lleno y está listo para escribir en él

OCEMPTY (output channel empty)
=0, el canal de entrada no está vacío
=1, el canal de entrada está vacío. [19]

3.2.4.2. LOS PUERTOS DE COMUNICACIONES DEL TMS320C40 EN EL SISTEMA DE PROCESAMIENTO PARALELO (PPDS)

Un sistema de procesamiento paralelo apoya un óptimo desempeño del sistema mediante la distribución de tareas entre dos o más procesadores. Cada TMS320C40 contiene seis idénticos puertos de comunicaciones bidireccionales de alta velocidad (Com0 - Com5)[20].

Los seis puertos, de 8 bits, proporcionan una rápida comunicación entre los procesadores a través de las interfases de comunicaciones de cada puerto. Acoplados con las dos interfases de memoria del C4x (global y local), es posible construir un sistema de procesadores paralelos que logra un óptimo desempeño del sistema mediante la distribución de tareas entre varios procesadores. Cada C4x puede pasar los resultados de su trabajo a otro, quedando en la capacidad de continuar trabajando.

Los puertos de comunicaciones Com0, Com1, Com3, Com4 conectan cada TMS320C40 directamente con otro, facilitando la comunicación entre procesadores.

Los puertos de comunicaciones Com2 y Com5 de cada TMS320C40 están dirigidos a los conectores externos P5 - P12 del TMS320C4x PPDS (Parallel Procesing Development System). Esos conectores permiten que la potencia de los TMS320C40s esté disponible para los dispositivos externos al TMS320C40 como convertidores A/D y D/A, controladores SCSI (Small Controller System Interface), tomadores de cuadros (frame grabbers) y otros periféricos.

Los puertos de comunicaciones pueden ser usados para traer (download) códigos y datos al TMS320C4x PPDS o enviarlos (upload) a un sistema anfitrión (host).

---

[19] UG, pág. 2-30, 12-125, 12-126

[20] El diagrama a bloques mostrando las interconexiones puede verse en el apéndice B

A continuación se muestra la disposición de los conectores externos para cada procesador en el PPDS:

CPU A: com2 --> P6
com5 --> P5

CPU B: com2 --> P8
com5 --> P7

CPU C: com2 --> P10
com5 --> P9

CPU D: com2 --> P12
com5 --> P11[21]

3.2.4.3. EL ACCESO DIRECTO A MEMORIA (DMA)

Los seis canales del coprocesador integrado[22] de acceso directo a memoria (DMA) pueden leer o escribir en cualquier localidad del mapa de memoria sin interferir con la operación del CPU. Esto permite la interfaz con memorias externas lentas y periféricos sin reducir el desempeño, o capacidad de transferencia[23], del CPU.

El coprocesador DMA contiene su propio generador de direcciones, registros de fuente y destino, y contador de transferencias, cuyos diagramas son mostrados en el apéndice B. Las direcciones dedicadas del DMA y los buses de datos permiten la minimización de conflictos entre el CPU y el coprocesador DMA. Una operación DMA consiste en una transferencia de un bloque o una palabra simple desde o hacia la memoria. Una característica clave del coprocesador DMA es su habilidad para reinicializar automáticamente cada canal a continuación de una transferencia de datos. Este coprocesador tiene dos modos básicos de operación:

- *El modo unificado*. Se emplea para transferencias entre áreas de memoria. Pueden transferirse palabras o bloques.
- *El modo dividido* (split). Se usa para transferencias bidireccionales entre áreas de memoria y puertos de comunicaciones.

---

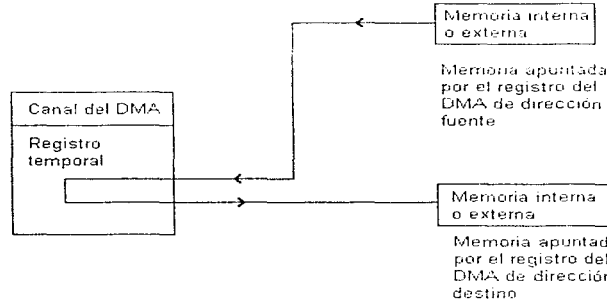
[21] PPDSTR, fig. 2-7; puede verse la figura del mapa de memoria de los periféricos en el apéndice B

[22] On-chip; incluido, contenido en el circuito integrado (chip)

[23] Throughput; se entenderá como la cantidad de información manejada por unidad de tiempo

Una transferencia de palabras unificada del DMA consiste en dos pasos:

- El canal del DMA lee el valor de los datos fuente de la dirección apuntada por el registro de dirección fuente y lo guarda en un registro temporal.
- El canal del DMA lee el valor del registro temporal y lo escribe en la dirección apuntada por el registro de dirección destino.



Puede emplearse el modo unificado para realizar transferencias entre puertos de comunicaciones, especialmente transferencias unidireccionales. El empleo del modo dividido presenta más ventajas en transferencias bidireccionales.

El modo dividido transforma un canal del DMA en dos canales del DMA:

- Canal primario. Se dedica a leer datos de una localidad en el mapa de memoria (interna o externa) y a escribirlos en una salida FIFO de un puerto de comunicaciones.
- Canal auxiliar. Se dedica a recibir datos de una entrada FIFO de un puerto de comunicaciones y a escribirlos en una localidad del mapa de memoria. [24]

Si la señal a transformar llega como una cadena continua de datos, el DMA podría ser usado para recibir el dato nuevo mientras el dato ya recibido es procesado. En este caso, la dirección de la fuente de datos del DMA apunta a la localidad de memoria correspondiente a un puerto serie o a otro puerto asociado con un dispositivo externo. El destino es un espacio de memoria destinado al almacenamiento.

---

[24] UG, pág. 2-30 y cap. 9

Hay dos maneras de usar tales buffers. Una posibilidad es designar un buffer como almacenador temporal y el otro buffer como el área de trabajo. Cuando el buffer de almacenamiento recibe la cantidad necesaria de datos, los datos son transferidos al área de trabajo y el DMA comienza a llenar nuevamente el buffer de almacenamiento.

Alternativamente, los dos buffers pueden ser considerados equivalentes: cuando el procesador termina de procesar y sacar los datos de uno y el DMA ha llenado el otro, los dos buffers intercambian funciones; i.e., el DMA comienza a llenar el primer buffer mientras el CPU está procesando los datos en el buffer recién llenado. [25]

3.2.4.4. TIMERS

Los dos módulos de timer son contadores de eventos o timers de propósito general de 16 bits con dos modos de señalización y reloj interno o externo. Pueden enviar su señal internamente al C4x o externamente al mundo exterior a intervalos especificados, o pueden contar eventos externos. Cada timer tiene un pin o terminal de I/O que puede ser usado como un reloj de entrada al timer, como una señal de salida manejada por el mismo o como un pin de I/O de propósito general. [26]

3.3. DATOS TÉCNICOS DEL PPDS

El Sistema de Desarrollo de Procesamiento Paralelo (PPDS) del TMS320C40 es el primer sistema de desarrollo diseñado exclusivamente para evaluar y desarrollar aplicaciones de procesamiento paralelo con punto flotante. Es posible desarrollar, comparar o referir (benchmark) y evaluar códigos (en tiempo real) en un generoso ambiente de desarrollo.

Las características de las secciones principales del TMS320C40 PPDS incluyen:

- Cuatro procesadores digitales de señales TMS320C40 en una tableta. Cada TMS320C40 es apoyado por un bus local que contiene:
  - i 64K de palabras de 32 bits de SRAM de "estado de espera cero" (zero wait-state).
  - ii 8K bytes de EPROM.
- 128K de palabras de 32 bits de SRAM de "estado de espera uno" (one wait-state) en un bus global compartido.
- Un conector de bus de expansión (P3) que provee una interfaz externa al bus de memoria global compartida.

---

[25] Papamichalis, DSP Applicationes with the TMS320 family, vol. 3. Pág. 76, 77

[26] UG, pág. 2-30

- Ocho conectores de comunicación externa (P5 - P12) que proveen una interfaz para conectar TMS320C40s fuera de la tableta y periféricos externos a los TMS320C40s del PPDS.
- Un conector de prueba JTAG (P4) que provee una interfaz para conectar el emulador XDS510 (contenido en el circuito) al TMS320C40 PPDS.
- Soporte para TMS320C40s con velocidades menores o iguales a 32 MHz para acceso de memoria compartida.
- Soporte para TMS320C40s con velocidades menores o iguales a 50 MHz cuando la memoria compartida no es usada. [27]

3.4. TIPOS DE DIRECCIONAMIENTO

3.4.1. DIRECCIONAMIENTO INDIRECTO

El direccionamiento indirecto es usado para especificar la dirección de un operando en memoria a través de los contenidos de un registro auxiliar, desplazamientos opcionales y registros índices. Esta aritmética es llevada a cabo por las ARAUS y es *no signado* (sin signo). (Todos los 32 bits de los registros auxiliares e índices son usados en direccionamiento indirecto.)

La flexibilidad del direccionamiento indirecto es posible porque las ARAUs en el C4x son usadas para modificar los registros auxiliares en paralelo con operaciones dentro del CPU principal. El direccionamiento indirecto es especificado por un campo de 5 bits en la palabra de instrucción, a la que se refiere como el *mod field*. Un *desplazamiento* puede ser tanto un entero explícito de 5 bits sin signo o de 8 bits contenido en la palabra de instrucción ó un desplazamiento implícito de uno.

Los dos registros de índice, IR0 e IR1, pueden también ser usados en direccionamiento indirecto, habilitando el uso de desplazamientos indirectos de 32 bits (IR0 e IR1 son tratados como enteros con signo). En algunos casos, es opcional un esquema de direccionamiento usando direccionamiento circular o de bit revertido. [28]

---

[27] PPDSTR, pág. 1-1, 1-2

[28] UG, pág. 5.5

3.4.2. DIRECCIONAMIENTO CIRCULAR

Muchos algoritmos de DSP requieren un buffer circular en memoria. En convolución y correlación, un buffer circular actúa como una ventana corrediza que contiene los datos más recientes para ser procesados. Cuando es traído un dato nuevo, éste es escrito sobre el dato más viejo. La clave para usar un buffer circular es la implementación de un modo de direccionamiento circular.

El registro de tamaño de bloque (block-size register) BK especifica el tamaño del buffer circular. Si el bit más significativo es igual a 1 en el registro BK, se le denomina bit N, con N ≤ 15, la dirección que sigue inmediatamente al final (bottom) del buffer circular puede ser encontrada mediante el concatenamiento de los bits que van del 31 hasta N +1 de un registro seleccionado por el usuario (ARn) con los bits N hasta 0 del registro BK la dirección del inicio (top) del buffer es llamada la base efectiva (EB) y puede ser encontrada mediante el concatenamiento de los bits 31 hasta N + 1 de ARn. Los bits de N a 0 de EB son cero. [29]

3.4.3. DIRECCIONAMIENTO DE BIT REVERTIDO

El C4x puede implementar transformadas rápidas de Fourier (FFT) con el direccionamiento de bit revertido. Si el dato a ser transformado está en el orden correcto, el resultado final de la FFT está en el orden de bit revertido. Para obtener el dato del dominio de la frecuencia en el orden correcto, ciertas localidades de memoria deben ser intercambiadas. El modo de direccionamiento de bit revertido hace innecesario este intercambio. La próxima vez que un dato deba ser accesado, lo hace en una forma de bit revertido en vez de secuencialmente.

Para la correcta operación de bit revertido en CPU o en DMA, la dirección base del direccionamiento de bit revertido debe ser localizado en un límite del tamaño de la tabla. Para aclarar este punto, supóngase una FFT de tamaño $N = 2^n$. Cuando los datos reales e imaginarios deben ser guardados en arreglos separados, los n LSBs de la dirección base deben ser cero (0) e IR0 deben ser inicializados en 2^{n-1} (la mitad del tamaño de la FFT). Cuando los datos reales e imaginarios son guardados en localidades consecutivas de memoria (Re-Im-Re-Im) los n + 1 LSBs de la dirección base deben ser cero (0) e IR0 debe ser igual a $IR0 = 2^n = N$ (tamaño de la FFT). [30]

---

[29] UG, pág. 5-25

[30] UG, pág. 5-30

3.4.3.1. DIRECCIONAMIENTO DE BIT REVERTIDO EN CPU

Un registro auxiliar (AR0 en este caso) apunta a la localidad física de un valor dato. Cuando se añade IR0 al registro auxiliar mediante el uso de direccionamiento de bit revertido, las direcciones son generadas en un modo de bit revertido (propagación revertida del acarreo). El índice más grande (IR0 en este caso) para reversión de bit es 00FF FFFFh.

3.4.3.2. DIRECCIONAMIENTO DE BIT REVERTIDO EN DMA

En el direccionamiento de bit revertido en DMA, dos bits en el registro de control del DMA habilitan el direc cionamiento de bit revertido en las lecturas de DMA (READ BIT REV) y las escrituras de DMA (WRITE BIT REV). El registro índice de la dirección fuente y el registro índice de la dirección destino definen el tamaño de del direccionamiento de bit revertido, Su función es similar a la del registro índice del CPU, IR0, descrita anteriormente. Dos transferencias de bloques de DMA son requeridas cuando el DMA es usado para transferencia de bit revertido de números complejos: una para transferir los puertos reales y otra para transferir los puertos complejos. [31]

---

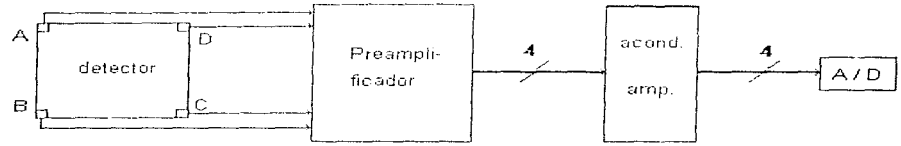
[31] UG; pág. 12-30, 12-31

CAPÍTULO CUARTO

DESARROLLO DE ALGORITMOS

4.1. DESCRIPCIÓN GENERAL

La descripción (que en este caso pretende ser más general, considerando la previa presentación de especificaciones hecha en los capítulos anteriores) se apoya del siguiente diagrama de bloques de la parte inicial del sistema:



Al detector llega una nube electrónica que puede representarse por una campana de Gauss; el centroide de esta nube se considerará como el fotón incidente en el fotocátodo.

La operación del fotomultiplicador provoca una cierta cantidad de carga que es registrada en las cuatro esquinas del detector. Estos niveles de carga detectados pasan a la etapa de preamplificación, donde son convertidos a niveles de voltaje.

La etapa siguiente es de acondicionamiento, donde se prepara a la señal para llevarla finalmente a un conjunto de convertidores analógico-digitales (ADC), cuyas salidas son cuatro palabras de doce bits.

Estas palabras (que se manejarán como A, B, C y D) llegan al PPDS a través de los puertos de comunicaciones. En este sistema se tiene el algoritmo que determina la posición del detector en que incidió el fotón de interés, mediante coordenadas (x, y).

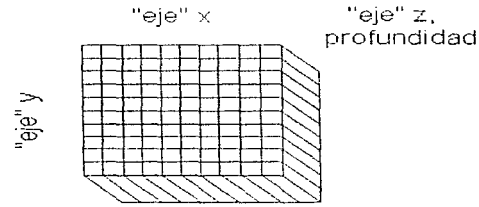
Aunque una parte del conjunto de algoritmos elaborados se halla en lenguaje ensamblador, el conjunto de programas proporcionado por Texas Instruments para empleo del sistema de DSPs incluye un compilador de C, que traduce los programas escritos en ANSI C a programas ejecutables de tipo COFF, aunque existe la alternativa de escribir los programas en código COFF para después hacerlos ejecutables e instalarlos en los DSPs.

Más adelante se presenta una exposición de las características de estas dos formas de programación (COFF y C), incluyéndose al final de las mismas un diagrama de flujo que muestra las diversas opciones que permite el paquete para programar los DSPs.

Conviene no olvidar el objeto principal del proyecto: un visualizador para un detector astronómico. Dicho visualizador es en realidad la interfaz con el usuario final, lo que lo hace acreedor a diversas características que también se detallarán en su oportunidad.

4.1.1. CONFIGURACIÓN DE LA MEMORIA

Una sección de la memoria del sistema total se configurará, para almacenamiento, de la siguiente manera:



Esta distribución de celdas, o sistema coordenado, corresponde a la distribución de pixeles de la zona del fotocátodo en la que se registrará la llegada de fotones.

Cada celda representa un bit de la palabra de ***x*** y de ***y***, teniéndose así para cada una de estas variables una palabra de 10 bits, que define un rango de [0, 2^{10}-1=1023], y pudiendo entonces numerarse los pixeles desde (0, 0) hasta (1023, 1023).

La razón por la cual se hace una disminución en el número de bits (12 en A, B, C, D y 10 en ***x*** y ***y***) es no buscar una precisión excesiva (no apreciable), que se tendría con 12 bits. Bastan 10 bits, que implican un "mapa" de 1023x1023 pixeles, para "cubrir" el área del detector. (Se hace pertinente aclarar que, aunque en el capítulo segundo la figura del detector mostraba un cuadrado, en realidad es un círculo de diámetro igual a 25 mm.)

Para una mayor claridad del algoritmo a elaborar, puede considerarse que en cada uno de estos pares coordenados se tiene un contador, que se activará cada vez que la posición determinada del fotón incidente corresponda a ese pixel. Este contador,

"definido" en la variable ***z***, será una palabra de 16 bits (2 bytes[1]), lo que permitirá contar hasta 2^{16}-1 = 65,535 eventos en cada punto (pixel). Este aspecto es comentado más adelante, en el inciso sobre el lenguaje C.

De este modo puede verse que los valores ***x*** y ***y*** indicarán en realidad la dirección del contador respectivo. Asimismo se observa que la capacidad de almacenamiento de información será: (1024)(1024)(2 bytes)=2'097,152 bytes, es decir, 2 MB; además podría contarse un total de (2097152)(65535)=137x10^9 eventos.

4.1.2. COORDENADAS DEL PUNTO DE INCIDENCIA

Las coordenadas (***x***, ***y***) del punto de incidencia quedan determinadas, en función de las palabras A, B, C y D, por las expresiones mostradas a continuación, como explica Hernández Alva[2]:

$$x = \frac{(A+B)-(C+D)}{A+B+C+D}$$

$$y = \frac{(B+C)-(A+D)}{A+B+C+D}$$

Puede verse que la definición de las coordenadas (***x***, ***y***) les permite tener signo negativo, lo que implicaría tener un sistema coordenado con rango [-512, 511] en vez de [0, 1023]. No obstante, para cumplir con requerimientos de llenado de la matriz, se les acondiciona sumándole a cada variable el valor 512, lo que provoca un desplazamiento del origen. Un segundo acondicionamiento, una resta de dicho valor, es hecho antes del despliegue final.

Los datos A, B, C y D son recibidos en un puerto del PPDS, para su posterior conversión de acuerdo con las expresiones indicadas. Esta etapa constituye la primera parte del conjunto de algoritmos elaborados para los DSPs, mostrado en el apéndice C.

Al final de esta sección de programa, se tienen los valores de *x* y ***y*** contenidos en las variables X y Y, debe ahora activarse el contador ya descrito y guardar la información. Antes de indicar estos procedimientos, se expondrán las características del COFF y el C.

---

[1] Aunque para el DSP 1 byte = 32 bits, se usará la convención de 8 bits si no se especifica lo contrario

[2] Hernández Alva. Diseño de la unidad aritmética de la electrónica de un detector tipo Mepsicrón

4.2. EL LENGUAJE ENSAMBLADOR DEL TMS320

Una de las formas de elaborar programas para los DSPs es hacerlo en código ensamblador, el cual se analizará en este punto.

4.2.1. MODELO DE ENUNCIADOS FUENTE

Los programas fuente en lenguaje ensamblador TMS320 consisten en enunciados fuente que contienen directivas de ensamblador, instrucciones en lenguaje ensamblador, directivas macro y comentarios. Las líneas de enunciados fuente pueden ser tan largas como el formato del archivo fuente lo permita, pero el ensamblador lee hasta 200 caracteres por línea. Si un enunciado contiene más de 200 caracteres, el ensamblador truncará la línea y hará una llamada de atención (warning).

Las siguientes líneas muestran ejemplos de enunciados fuente:

```
SIMB1     .set    0A5h
Inicio:   ADDI    SIMB1+5,R1      ;suma (SIMB1+5) al contenido de R1
          LDI     R1,R2           ;mueve el contenido de R1 a R2
```

Un enunciado fuente puede contener cuatro campos ordenados. La sintaxis general para los enunciados fuente es como se indica:

[etiqueta] [:] mnemónico [lista de operandos] [;comentario]

Deben tenerse en cuenta estas consideraciones:

- Todos los enunciados deben comenzar con una etiqueta, un espacio en blanco, un asterisco o un punto y coma(;).
- Las etiquetas son opcionales; si se usan deben comenzar en la columna 1.
- Uno o más espacios en blanco deben separar cada campo. Los espacios de tabulación son equivalentes a los espacios en blanco.
- Los comentarios son opcionales; los que comiencen en la columna 1 pueden comenzar con un asterisco o un punto y coma, pero losque comiencen en cualquier otra columna *deben* comenzar con un punto y coma.

Cada uno de estos campos es extensamente explicado en el manual correspondiente[3].

---

[3]Texas Instruments. TMS320 Floating-Point DSP Assembly Language Tools User's Guide; abreviado como ALT.

4.2.2. INTRODUCCIÓN AL COFF

La siguiente información está basada en uno de los manuales de Texas Instruments [4], en el que pueden encontrarse los detalles y ejemplos respectivos.

El ensamblador y el enlazador (linker) crean archivos objeto que pueden ser ejecutados por un dispositivo TMS320C3x/C4x. El formato en que esos archivos objeto se encuentran es llamado "formato de archivos objeto comunes" o *COFF*.

COFF hace más fácil la programación modular porque induce a pensar en términos de bloques de códigos y datos cuando el programador escribe un programa en lenguaje ensamblador. Estos bloques son conocidos como *secciones*. Tanto el ensamblador como el enlazador proporcionan directivas que permiten crear y manipular secciones.

4.2.2.1. SECCIONES

La unidad más pequeña de un archivo objeto se denomina una sección, y es un bloque de código o datos. Los bloques originados por una misma directiva finalmente ocuparán un espacio contiguo en el mapa de memoria del TMS320. Cada sección de un archivo objeto está separado de las demás secciones y es diferente de ellas. Los archivos objeto COFF siempre contendrán tres secciones por omisión:

.text	usualmente contiene código ejecutable
.data	usualmente contiene datos inicializados
.bss	usualmente reserva espacio para variables no inicializadas

Además, el ensamblador y el enlazador permiten crear, nombrar y enlazar secciones *nombradas* que son usadas en forma similar a las secciones anteriores.

Es importante hacer notar que hay dos tipos básicos de secciones:

I *Secciones inicializadas*. Contienen datos o código; de ellas se tienen:
- .text
- .data
- nombradas, creadas con las directivas .sect y .asect del ensamblador

II *Secciones no inicializadas*. Reservan espacio en memoria para datos no inicializados; de ellas se tienen:
- .bss
- nombradas, creadas con la directiva .usect del ensamblador

---

[4] ALT.

Debido a que la mayoría de los sistemas contiene varios tipos diferentes de mapas de memoria, el uso de secciones puede ayudar a usar la memoria destino más eficientemente. Todas las secciones pueden ser dispuestas independientemente y pueden colocarse diferentes secciones en varios bloques de la memoria destino.

4.2.2.2. DESCRIPCIÓN DEL ENSAMBLADOR

El ensamblador traslada archivos fuente en lenguaje ensamblador a archivos objeto, que están en COFF. Aunque se analizarán con mayor detalle en incisos posteriores, a continuación se presenta un resumen sobre las funciones del ensamblador:

- Procesa los enunciados[5] fuente en un archivo de texto para producir un archivo objeto reubicable.
- Produce un listado fuente (si se solicita) y proporciona al usuario control en dicho listado.
- Permite al usuario segmentar su código en secciones y mantener un SPC[6] para cada sección del código objeto.
- Define y referencia símbolos globales y añade un listado de referencia cruzada al listado fuente (si se solicita).
- Ensambla bloques condicionales.
- Soporta macros, permitiendo al usuario definir macros en línea o en una biblioteca.
- Permite al usuario ensamblar códigos de TMS320C3x y TMS320C4x.

4.2.3. HERRAMIENTAS PARA PROGRAMACIÓN DE LOS DSPs

El C4x está bien apoyado por un juego completo de hardware y herramientas de desarrollo de software, incluyendo un compilador de C, ensamblador y enlazador. A continuación se enlistan las diferentes herramientas proporcionadas para manejar y desarrollar programas para los DSPs, con una breve descripción de ellas:

- *El compilador de C* acepta código fuente en C y produce código fuente en lenguaje ensamblador del TMS320C3x/C4x.

- *El ensamblador* traduce archivos fuente en lenguaje ensamblador a archivos objeto en lenguaje de máquina COFF.

- *El archivador* permite reunir un grupo de archivos en un archivo-conjunto, también llamado biblioteca. El archivador puede además modificar una biblioteca borrando, remplazando, extrayendo o añadiendo miembros.

---

[5]Statements: declaraciones, enunciados

[6]Contador de programa para secciones

- Las dos bibliotecas objeto, *rts30.lib* y *rts40.lib*, incluidas en el compilador de C, contienen funciones estándar para soporte al momento de ejecución (runtime-support), funciones de utilería del compilador y funciones matemáticas que pueden ser llamadas desde programas en C que hayan sido compilados para el C3x ó C4x. Estas bibliotecas pueden ser incluidas en cualquier programa compilado para el C3x ó C4x, con el modelo de memoria pequeña y el modelo de tiempo de ejecución estándar. Las bibliotecas de este tipo para otras configuraciones deben ser creadas desde rts.src usando la utilería de instalación de soporte de tiempo de ejecución, *mk30*. Estas bibliotecas contendrán las mismas funciones que las rts30.lib y rts40.lib.

- *El enlazador* combina archivos objeto en un módulo objeto ejecutable simple. Al crear este módulo, realiza la reubicación y resuelve las referencias externas. El enlazador acepta archivos reubicables COFF y bibliotecas objeto como entradas.

El resultado de este proceso de desarrollo es producir un módulo que puede ser ejecutado en un sistema destino TMS320C3x/C4x. Pueden emplearse diversas herramientas de depuración para refinar y corregir el código antes de llevarlo al sistema destino:

- *El simulador* es un programa de software que simula las funciones del C3x/C4x. Puede ejecutar módulos objeto COFF enlazados.

- *El emulador XDS* es un emulador en un circuito, residente en PC, de tiempo real, con la misma interfaz que el simulador.[7]

Cada una de las herramientas posee un conjunto de opciones que pueden ser empleadas para obtener el tipo de salida requerido en particular. Dichas opciones son tratadas ampliamente en la bibliografía señalada.

Estas herramientas utilizan el COFF, que favorece la programación modular. COFF permite dividir el código en bloques lógicos, definir el mapa de memoria del sistema y entonces enlazar el código en áreas específicas de memoria. COFF también provee amplio soporte para depuración a nivel de fuente.

4.2.3.1. CARACTERÍSTICAS DEL COMPILADOR DE C

El compilador de ANSI C para optimización del C4x es un compilador de optimización de abundantes características que traduce programas en el ANSI C estándar a lenguaje ensamblador fuente de C4x.

---

[7]Texas Instruments. TMS320 Floating Point DSP Optimizing C Compiler User's Guide; abreviado como OCC pág. 1-3

La lista siguiente enumera la amplia gama de funciones proporcionada por el compilador de C. Estas funciones son descritas en detalle en la bibliografía correspondiente[8].

i Estándar ANSI C
ii Optimización
iii Soporte en tiempo de ejecución de estándar ANSI
iv Salida en fuente de ensamblador
v Modelos de memoria grande y pequeño
vi Programa shell del compilador
vii Interfaz flexible de lenguaje ensamblador
viii Preprocesador (C) integrado
ix Archivos objeto COFF
x Código para ROM
xi Utilería para "entrelistar" (interlist) fuentes
xii Tamaños de datos de 32 bits
xiii Utilería para construir bibliotecas

4.2.2.3. COMANDOS PARA LAS HERRAMIENTAS

Las herramientas de programación y depuración mencionadas anteriormente son llamadas individualmente mediante los comandos listados en seguida, aunque pueden llamarse también en conjunto, como lo muestra el diagrama de flujo de desarrollo de código posteriormente. En dicho diagrama de flujo también puede observarse el tipo de archivo que produce como salida cada uno de los comandos; la descripción específica de estos archivos puede verse en los manuales propios[9].

ac30	Descriptor gramatical (parser) de C
cg30	Generador de código
asm30	Ensamblador
clist	Utilería de entrelistado
lnk30	Enlazador
opt30	Optimizador
emu4xo	Emulador para OS/2
sim4x	Simulador

[8]OCC pág. 1-5

[9]OCC y Texas Instruments TMS320C4x C Source Debugger User's Guide, abreviado como CSD

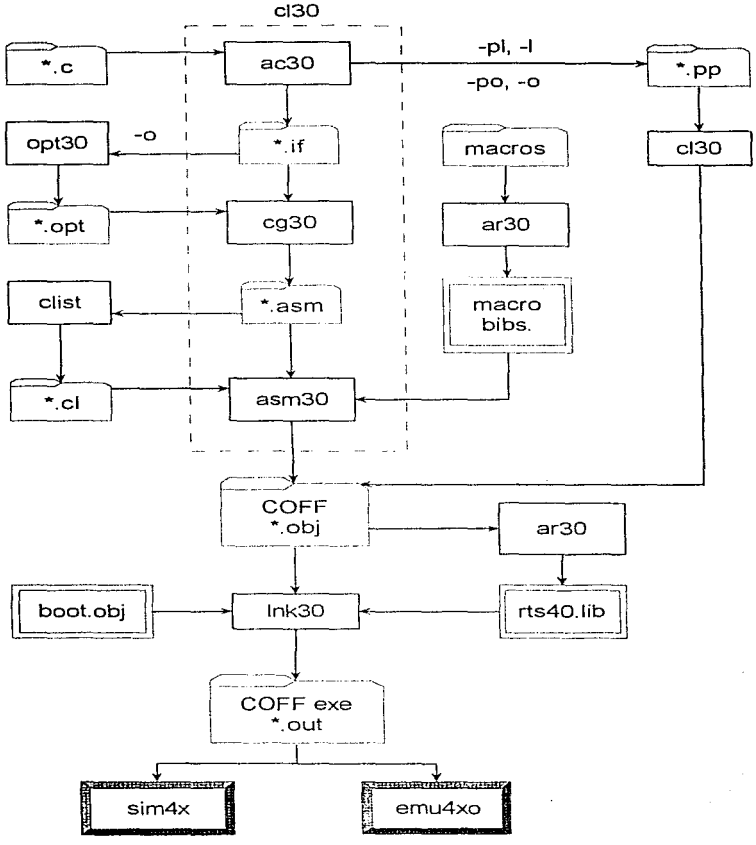


Diagrama de flujo de desarrollo de código

4.3. EL LENGUAJE DE PROGRAMACIÓN C

La siguiente descripción de características es hecha en forma general, para la obtención de más detalles o especificaciones, puede acudirse a los textos presentados en la bibliografía.

C es un lenguaje de programación de propósito general que ha sido estrechamente asociado con el sistema UNIX, en donde fue desarrollado, puesto que tanto el sistema como los programas que corren en él están escritos en lenguaje C. Sin embargo, este lenguaje no está ligado a máquina o sistema operativo alguno, siendo sumamente portable.

Aunque se le llama "lenguaje de programación de sistemas" debido a su utilidad para escribir compiladores y sistemas operativos, se utiliza con igual eficacia para escribir importantes programas en diversas disciplinas.

Al lenguaje C se le ha identificado como un lenguaje relativo de "bajo nivel", o bien, de "nivel medio", lo cual no pretende hacer una clasificación despectiva, sino mostrar que C maneja el mismo tipo de objetos que la mayoría de las computadoras, llámense caracteres, números y direcciones. Éstos pueden ser combinados y cambiados de sitio con los operadores aritméticos y lógicos implantados por máquinas reales. Como lenguaje de nivel medio, C combina elementos de lenguaje de alto nivel con el funcionalismo del lenguaje ensamblador.

El lenguaje C proporciona lo que el programador requiere: pocas restricciones, estructuras de bloques, funciones independientes (sin necesidad de compartir variables o valores) y un compacto conjunto de palabras clave.

El componente estructural principal de C es la función o subrutina independiente. En C las funciones son los bloques constitutivos en los que se desarrolla toda la actividad de los programas. Son las que permiten definir las tareas de un programa y codificarlas por separado, permitiendo así que los programas sean modulares. Una vez que se ha creado una función, se le puede aprovechar, desempeñándose perfectamente en distintas situaciones, sin crear efectos secundarios en otras partes del programa.

Todas estas características, sumadas a las que se presentan en el siguiente punto, pueden observarse en el código correspondiente en el apéndice C.

4.3.1. TURBO C

El compilador para lenguaje C, llamado Turbo C, que la casa Borland presentó en junio de 1987, revolucionó el mercado de compiladores C para computadoras personales, casi exclusivamente acaparado, hasta ese momento, por los compiladores de Microsoft, Lattice y Aztec.

Turbo C ha sido uno de los compiladores más rápidos del mercado, contiene la definición completa del lenguaje, más algunas innovaciones; combina la potencia, flexibilidad y portabilidad del lenguaje C con un entorno integrado de desarrollo: editor, compilador y enlazador, estos dos últimos con la consecuente propiedad de depuración de módulos.

Otras ventajas que presenta son: facilidad de instalación, velocidad de compilación, una ampila biblioteca de funciones, código eficiente y compacto, utilidades para desarrollar aplicaciones grandes, acceso a los registros del procesador, gestión de las interrupciones y un largo etc.

La declaración del contador (descrito anteriormente como variable de profundidad z) como *unsigned int* en la rutina en Turbo C cumple con las condiciones buscadas en la configuración de la memoria, pues el espacio destinado a este tipo de variables en el estándar ANSI es de 16 bits en el rango descrito: [0, 65,535].

El compilador permite definir seis modelos de memoria distintos para aprovechar la arquitectura del microprocesador 8086 y se muestran en la siguiente tabla:

Modelos de memoria del compilador de Turbo C			
modelo	código, datos y pila del programa	tipo manejado por omisión	
		funciones	apuntadores a datos
Tiny	64 K para todo el conjunto	near	near
Small	64 K para el código y 64 K para los datos y pila	near	near
Medium	64 K para los datos y pila y 1 M para el código	far	near
Compact	64 K para el código, 64 K para la pila y datos de tipo "static" y 1 M para el heap	near	far
Large	64 K para la pila y datos de tipo "static", 1 M para el código y 1 M para el heap	far	far
Huge	64 K para cada uno de los segmentos múltiples de datos, 64 K para la pila y 1 M para el código	far	far

4.4. MANEJO DE LA INFORMACIÓN

Es necesario reservar un espacio de memoria para guardar los datos y realizar su posterior manipulación y análisis. Aunque el PPDS maneja 4 Gwords ó 16 Gbytes en memoria, la mayor parte de ella está destinada a buses, como puede verse en los mapas correspondientes, tratados en el capítulo anterior, teniéndose solo 2 bloques de RAM de 4 kbytes cada uno, insuficientes para manejar los datos en cuestión: se requieren 2 Mbytes.

Por las razones anteriores, se decidió enviar la información a la computadora para seguidamente almacenarla en un archivo y realizar su despliegue. No obstante, para no reducir la velocidad de transferencia de datos, dicho archivo estará en una sección de memoria RAM empleada como unidad de disco (RAMdrive[10]).

Los datos son almacenados en forma de matriz en un archivo que se halla en la unidad de disco virtual (unidad de RAM). Una vez que la matriz producida por la rutina en C se ha llenado, una subrutina se encarga de grabarla en un archivo del disco duro, antes de volver a comenzar nuevamente el proceso de carga de datos en dicha matriz.

A continuación se enumeran los pasos del proceso, en resumen:

1 En el archivo CONFIG.SYS se declara una unidad de RAM (disco virtual), para almacenar los datos.
2 Un archivo de proceso por lotes se encarga de copiar del disco duro a la unidad virtual una rutina en C y sus archivos necesarios.
3 Los puertos del TMS320C40 PPDS reciben la información del detector y el programa en ensamblador la procesa.
4 Este programa se encarga de enviar la información del PPDS a la computadora.
5 La rutina en C crea un archivo en la unidad virtual, almacena los datos en ella en forma de matriz y los despliega en la pantalla.
6 Al llenar alguna celda de la matriz en toda su capacidad (cuenta=65535), se hace una actualización de la memoria para poder seguir recibiendo datos.

4.5. ESPECIFICACIONES DEL VISUALIZADOR

El programa encontrado en la computadora implica ya la estructuración de la interfaz ante el usuario final, está elaborada en Turbo C. El bloque básico está constituido de los siguientes módulos, que deben ser copiados a toda computadora en la que desee emplearse el visualizador:

---

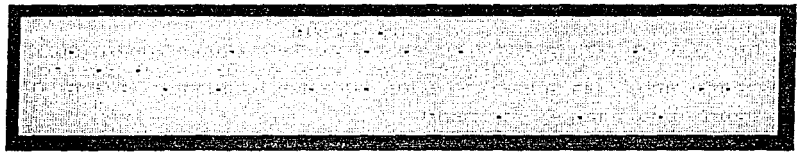
[10]Véase el apéndice A.

- El archivo de proceso por lotes visual.bat, que realiza la copia al disco virtual del resto de los archivos e inicia su ejecución.
- El programa G10-150.EXE, que contiene los algoritmos para manejar la matriz de datos y realizar su despliegue en pantalla.
- El archivo K, que contiene la información mostrada en la pantalla de ayuda
- El archivo A, un archivo vacío que permite la continuación del proceso al tratar equivocadamente de abrir un archivo.

La siguiente descripción puede emplearse también como guía del usuario.

4.5.1. PANTALLA DE INICIO

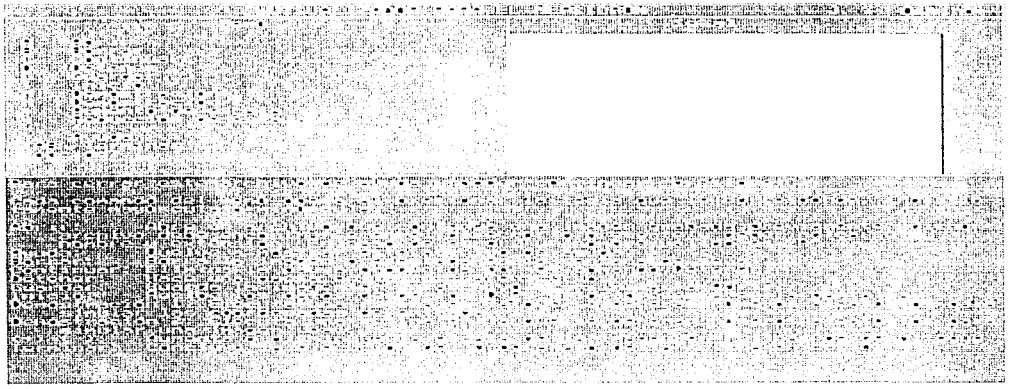
Al iniciar el programa, se presenta al usuario una pantalla similar a la siguiente. Esta característica tuvo su origen en una petición de los diseñadores de las primeras etapas del detector: los dispositivos electrónicos empleados alrededor del detector, y el detector mismo, no resistirían la aplicación de determinados valores de voltaje, lo que causaría su destrucción.



En esta pantalla no se dan más detalles debido a que no se trata de informar al usuario no experto de lo que debe hacer en cada situación, antes bien, se busca que no maneje dispositivos o condiciones que no conozca.

4.5.2. PANTALLA DE VENTANAS

Después de la presentación de las precauciones pertinentes, ya explicadas, se despliega una pantalla, similar a la mostrada a continuación, formada por diversas ventanas, cada una con funciones diferentes y cuya explicación procede ahora:



4.5.2.1. VENTANA DE AYUDA

Esta ventana (mostrada en la parte inferior de la pantalla, como se ve arriba) presenta una breve descripción de los comandos del menú que el usuario puede emplear. Aparece por omisión en la pantalla de ventanas al principio (el usuario no necesita llamarla) y, para continuar, es necesario cerrarla presionando la tecla C un par de veces.

4.5.2.2. VENTANA DE MENÚ

Presenta las diferentes opciones que el usuario puede emplear:

- *Abrir* (tecla A). Abre un archivo que contenga una imagen previamente grabada con este mismo sistema y despliega dicha imagen. El archivo debe encontrarse en la unidad y directorio actuales; por esta última razón, es conveniente "cargar" o copiar en el disco virtual aquellas imágenes que quieran ser desplegadas o, en su defecto, correr el programa directamente en disco duro, sin emplear el archivo visual.bat. Si se ejecuta esta instrucción por error bastará dar, como nombre de archivo, A, que es un archivo vacío.

- *Ayuda* (tecla Y). Despliega en la parte inferior de la pantalla el archivo de ayuda, que describe brevemente los comandos o instrucciones que el usuario puede emplear. Una vez abierto este archivo, debe cerrarse presionando la tecla C, para poder continuar.

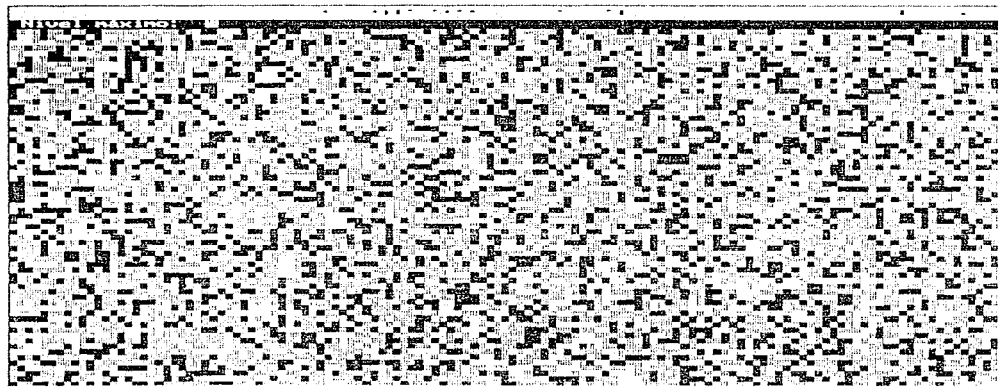
- *Borrar* (tecla B). Limpia la pantalla de señal, sin eliminar el archivo que haya sido abierto ni la información contenida en la matriz en ese momento.

- *Continuar* (tecla C). Cierra el archivo de ayuda si se encuentra abierto o elimina la pausa producida por la instrucción Detener.

- *Detener* (tecla D). Hace una pausa en el despliegue en curso. Puede ser utilizado para la rutina de demostración 1 o para la rutina de adquisición real.

- *Guardar* (tecla G). Almacena la matriz de datos en un archivo en el directorio en curso. Si el programa se está corriendo desde disco duro, el archivo se almacenará en esta unidad, es análogo con una unidad virtual.

- *Inicio* (tecla I). Ejecuta la rutina de adquisición de datos desde los DSPs y los maneja en el modo descrito: los guarda en una matriz y los presenta en pantalla completa. Además, indica cuál es el nivel máximo alcanzado; esto significa un nivel por cada cuenta de 10,000 fotones; es decir, el primer nivel es de 1 a 9,999, el segundo de 10,000 a 19,999, etc., hasta 65,535.



- *Demostración 1* (tecla M). Ejecuta una rutina en la que se generan números aleatorios simulando coordenadas de entradas y despliega la imagen resultante en la ventana de señal.

- *Demostración 2* (tecla N). Ejecuta una rutina en la que se generan números aleatorios para simular una imagen resultante real, incluyendo la consideración de la capacidad de actualización de celdas en la matriz de datos.

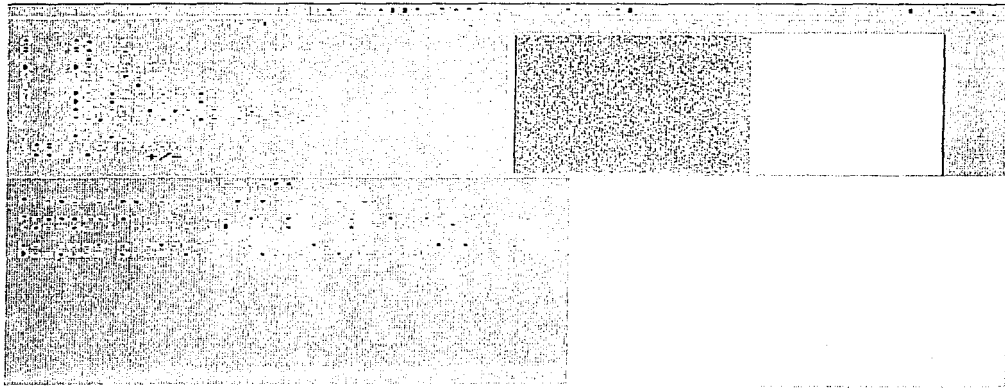
- *Escala* (teclas + y -). Cambia la escala de la imagen siempre en números enteros.

i Cuando se está haciendo el despliegue de demostración 1, aumenta o disminuye la escala de la imagen en la ventana de señal.

ii Cuando se está ejecutando la rutina de despliegue real, hace lo mismo pero en pantalla completa.

iii Cuando se ha desplegado en la ventana de señal la imagen correspondiente a un archivo previo, muestra el mensaje "Preparando aumento de escala" y pide al usuario definir un valor de escala en el rango de 1 a 10 y vuelve a desplegar la imagen, esta vez en pantalla completa y con valor de escala aportado (un valor menor que 1 ó mayor que 10 producirá una escala de 1 ó 10, respectivamente).

iv Si ya se ha llegado al valor mínimo (1) o al valor máximo (10), no tendrá efecto presionar estas teclas para pasar de estos límites.



- *Terminar* (tecla de tabulación). Concluye la rutina de demostración 1 o la de despliegue real. Presionarla cuando la ventana de estado indica "Esperando instrucción", provocará que el programa termine y se regrese a la línea de comandos del sistema operativo.

A continuación se presentan puntos de interés para el usuario con respecto a estas instrucciones:

- En caso de presionar una tecla que no corresponda a las que se han presentado, simplemente aparecerá un par de signos de interrogación en la pequeña ventana de opción, preguntando por una opción válida.

- Es permitido emplear la tecla BackSpace (retroceder) al abrir un archivo o guardar una imagen.

- Al iniciar el programa es necesario presionar un par de veces la tecla C para cerrar el archivo de ayuda y poder continuar con otra instrucción.

- Si quiere grabar la imagen actual:
i presione TAB para detener la adquisición,
ii presione G para guardar dicha imagen,
iii proporcione el nombre del nuevo archivo;
iv para volver a la adquisición, presione I (i).

4.5.2.3. VENTANA DE SEÑAL

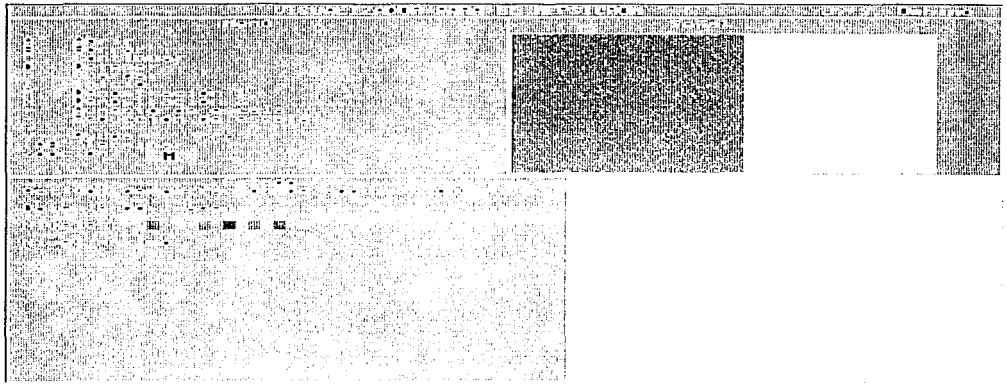
En ella se hará el despliegue de las imágenes generadas por las rutinas de demostración correspondientes; mostrará también el archivo que haya sido abierto, en escala de 1 pixel (mínima).

4.5.2.4. VENTANA DE ESTADO

Presenta el estado actual de la rutina o el proceso de la instrucción solicitada:

- Para la demostración 1, presenta los mensajes "Activo (demostración de adquisición)" y "Obteniendo imagen" y presenta la lista de niveles por cuenta de fotones, que es la siguiente, además del nivel máximo alcanzado hasta el momento:

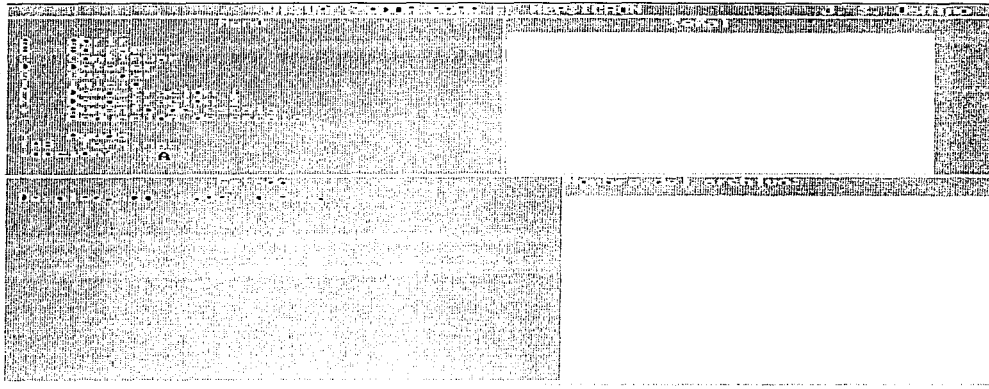
i	negro	cero (no indicada en realidad)
ii	café	1 - 9 999
iii	gris claro	10 000 - 19 999
iv	azul claro	20 000 - 29 999
v	rojo	30 000 - 39 999
vi	verde	40 000 - 49 999
vii	magenta	50 000 - 59 999
viii	amarillo	60 000 - 65 535



- Para la demostración 2, solo indica que se trata de esta demostración con el mensaje "Activo (demostración de imagen final)" mientras la rutina está en curso.

- Al estar guardando una imagen en un archivo, solo indica que se realiza esta operación con el mensaje "Guardando...". Junto a la ventana de estado aparecerá una segunda ventana pidiendo el nombre del archivo correspondiente.

- Al estar abriendo un archivo con una imagen previamente grabada, solo indica que se realiza esta operación con el mensaje "Desplegando imagen previa". Al igual que al solicitar Guardar, aparece una ventana pidiendo un nombre de archivo válido, como puede verse.



- Al solicitar un cambio de escala, pide el nuevo valor del factor de escala y anuncia que la imagen será desplegada en pantalla completa. La pantalla correspondiente ya ha sido mostrada.

- Mientras no se realice instrucción alguna ni se esté ejecutando algún proceso o rutina, esta ventana solo mostrará el mensaje "Esperando instrucción".

CAPÍTULO QUINTO

CONCLUSIONES

El proyecto Mepsicrón del Instituto de Astronomía no había tenido modificaciones importantes (podría decirse que ninguna) desde 1985, presentando las condiciones que ya han sido expuestas en el capítulo segundo. Las diferentes etapas están siendo modificadas actualmente y ha tocado al presente trabajo el mejoramiento de una de ellas.

Se buscó en este trabajo de tesis un mejor aprovechamiento de la memoria de la computadora empleada para el despliegue de imágenes. El conjunto de programas desarrollado aprovecha dicha memoria para conseguir procesos de transferencia y almacenamiento de datos en forma mucho más práctica y rápida que lo que se tenía originalmente.

Estos procesos son más prácticos porque no se está requiriendo el banco de memorias que se había construido inicialmente para este propósito, sino solo la memoria de tipo RAM que se tiene en la computadora. Además, se ha dado al usuario la capacidad de disponer convenientemente de estos datos en forma de archivos, que él puede almacenar o consultar con tan solo hacer uso de una instrucción en cada caso.

Del mismo modo, el proceso es más rápido por el empleo de DSPs para realizar los procesos de cálculo de coordenadas; esto significa que se han diseñado programas para ser ejecutados en dispositivos fabricados explícitamente para el manejo de señales, en vez de recurrir a la implementación de sistemas que realicen dichos procesos.

El equipo basado en DSPs fue adquirido para el mejoramiento del detector astronómico, empero no había sido puesto en funcionamiento, esto es, no se había realizado su instalación, por consiguiente, no habían sido observados los procedimientos propios para su manejo ni, por supuesto, habían sido descubiertos los diferentes conflictos con el resto del sistema vía la realización de pruebas.

Todos los aspectos del párrafo anterior han sido tratados en las diferentes secciones de este trabajo. Con ello se busca no simplemente la exposición de los resultados obtenidos, sino el aprovechamiento de los mismos en los procesos que sé consideren para este equipo en el futuro.

No se hace aquí la pretensión de un funcionamiento óptimo de dichos procesadores, más aún, con base en las pruebas realizadas con el simulador y emulador de los mismos, se llega a la conclusión de que, con las condiciones buscadas para el Mepsicrón en la actualidad en el Instituto, el sistema de procesamiento paralelo será subutilizado hasta las etapas del Mepsicrón sean operadas conjuntamente.

Esta conclusión tiene también como razón el hecho de que será hasta ese momento en que sean conocidos los nuevos problemas y, en cuya solución, sea entonces conocida la capacidad real de las características tecnológicas de los procesadores digitales de señales.

En lo referente a la situación de no tenerse aún la conexión de ambas etapas cabe recordar que el problema que se tuvo con los primeros Mepsicrones se debió a características de construcción de los fotocátodos, que se tradujeron en la pérdida total o parcial de algunos de estos componentes; en el caso de la pérdida parcial se habla de "ceguera del detector", lo que significa que no responde apropiadamente a los estímulos que recibe.

Como lo menciona el título mismo, el objetivo de la tesis es la implementación de un visualizador para el sistema Mepsicrón, esto es, una interfaz que proporcione a un usuario final una imagen o que despliegue un conjunto de imágenes que el usuario sea capaz de interpretar, meta que ha sido conseguida.

El programa final presentado está sujeto a todo tipo de posibles modificaciones debido a que aún no ha sido probado con la etapa del detector. Esto significa que, aunque el programa esté ejecutando actualmente rutinas de demostración o simulación (que permiten analizar cuáles serían las modificaciones pertinentes para su mejoramiento), todavía no han sido proporcionadas las conexiones físicas formales con el detector para así ejecutar un proceso real.

No obstante, las ventajas proporcionadas por el lenguaje de programación Turbo C, particularmente las funciones independientes, permiten un mantenimiento eficiente y cómodo de los programas; así, bastarán cambios, que pueden esperarse sencillos, en el código fuente para tener rutinas que manejen datos reales como entrada, en vez de datos generados para simulación. No se detallan aquí las posibles modificaciones debido a que es claro que éstas resultarán de observar el desempeño conjunto de las dos etapas del sistema.

No es necesaria la elaboración de todos los programas que pudieran ser requeridos para esta conexión, ya que en el paquete de programas de los DSPs se incluyen algunos que, si bien podrían no ser suficientes, pueden emplearse como base para conseguir las características deseadas en el sistema funcionando en su totalidad.

En lo que respecta al sistema operativo, en uno de sus manuales, el fabricante de los DSPs dilucida: "Usted no puede usar el sistema de desarrollo de procesamiento paralelo (PPDS) 'C4x como un sistema destino cuando emplea el depurador en un ambiente de DOS porque el PPDS tiene múltiples 'C4xs en la ruta de búsqueda del emulador".

Aunado con lo anterior, en el manual de instalación del sistema de los DSPs, en donde se muestra la lista de requerimientos tanto de hardware como de software, se especifica, como primer punto dentro del concepto de software, el sistema operativo OS/2 versión 1.1 ó posterior. Las consultas realizadas al hotline de Texas Instruments pusieron en claro la necesidad del sistema operativo empleado finalmente (OS/2 versión 2.1).

Se ha hecho hincapié en detallar características del sistema operativo de IBM OS/2 en los apéndices dado que no había sido utilizado en el Instituto de Astronomía y presenta ciertas diferencias y semejanzas con respecto a los que han sido manejados. Fue necesario salvar diversos problemas presentados por no conocer en un principio estas características, pretendiéndose en esta ocasión que, al efectuarse modificaciones y mejoras al sistema, el proceso se vea agilizado con la información que en el presente trabajo se proporciona.

El material proporcionado por el Instituto de Astronomía como apoyo para la implementación de mejoras al sistema Mepsicrón fue el que en un principio se consideró adecuado, lo cual consiste en una computadora 486 DLC de 40 MHz con 4 Mbytes en memoria de tipo RAM; no teniendo este sistema conflictos con lo que el sistema de DSPs mostraba como necesario para su correcta instalación.

Sin embargo, al comenzar a trabajar con dicho equipo, se hizo presente el requerimiento de otras características, situación que fue enfrentada como se menciona: no se tenía el sistema operativo necesario y fue instalado sobre el ya existente en el disco duro (el equipo proporcionado no tuvo exclusividad para el sistema Mepsicrón); aunque fue suficiente la capacidad del disco duro para dicha instalación, no ocurrió lo mismo con la RAM, puesto que actualmente el desempeño en velocidad es muy bajo.

Se intentó el cambio de disco duro a una computadora con mayor capacidad, como se buscaba, sin embargo, el modelo de dicho disco no permitía un traslado sencillo (no fue reconocido por las distintas máquinas en que se hizo el intento). Actualmente continúa con la capacidad descrita de RAM, a la que debería restarse un mínimo de 2 MB por las características expuestas en el capítulo cuarto sobre la unidad de disco virtual; empero el sistema se vuelve inadmisiblemente lento cuando es dispuesta una unidad de este tipo de tan solo 100 kB.

Del modo descrito, la interfaz proporcionada finalmente para el usuario es ejecutada en forma adecuada en las condiciones actuales si se elimina la disposición de memoria RAM. El programa de la interfaz no presenta conflicto alguno con el sistema operativo ni con el estado actual del hardware, ya descrito.

Para un aprovechamiento real del visualizador del sistema Mepsicrón será necesario que el Instituto de Astronomía disponga un equipo con las características que ahora se conocen indispensables: una computadora 486 de 50 MHz con 8 MB de RAM teniendo OS/2 versión 2.1 como sistema operativo único.

Por supuesto, la configuración enunciada en el párrafo anterior es la mínima, todas ellas pueden ser incrementadas, particularmente, por la posibilidad de futuros cambios, sería más adecuado tener 16 MB en memoria RAM; asimismo, podría utilizarse un disco duro pequeño si el equipo va a tener exclusividad para Mepsicrón. Esta última consideración es hecha porque en el sistema actual en un principio no fue instalada la ayuda en línea del sistema operativo por no hallarse la capacidad precisada.

Actualmente este sistema operativo aún no es manejado eficazmente por las demás personas responsables del proyecto Mepsicrón o cercanas al mismo, lo cual hace también indispensable contar con dicha ayuda (ésta no se halla en los manuales).

Entre otras características que pueden buscarse es posible incluir un lenguaje visual apropiado para OS/2, el cual podría ofrecer más ventajas al implementar el visualizador que lo conseguido hasta el presente.

La importancia que este proyecto reviste se basa en la intención de conseguir un sistema para observación astronómica que cumpla con requerimientos establecidos por científicos de todo el orbe, y que no esté apoyado solamente por el aprovechamiento de tecnología extranjera sino que también haya sido producto del desarrollo científico nacional.

APÉNDICE A

CONCEPTOS DE COMPUTACIÓN

En el presente apéndice se exponen diversos aspectos de computación, dando atención especial a las cuestiones de la memoria. Inicialmente se habla de la memoria RAM y en seguida se explican las diferentes divisiones que se han hecho de la misma; la forma de manejar estas clases de memoria es el tema que aparece en tercer término. Conociendo las diferentes características de ellas (como ventajas y aplicaciones específicas), puede entonces analizarse las diferencias que se presentan al trabajar con distintos sistemas operativos, abordando así los aspectos que hubo que salvar en este proyecto.

A.1. MEMORIA RAM

Una memoria del tipo RAM (Random Access Memory) almacena datos mientras el circuito a que pertenece esté siendo alimentado. En las memorias hechas a partir de semiconductores se emplea este nombre (RAM) para indicar aquellas memorias en las que toma el mismo tiempo realizar una operación de escritura que una de lectura, cuando en realidad quiere decir que es tan fácil tener acceso a una dirección de memoria como a otra, de aquí el término "acceso aleatorio".

Cabe recordar que la información contenida en la RAM cambiará frecuentemente entre muchas localidades de ésta última a medida que algún programa sea ejecutado. Esto hace preciso que los tiempos de los ciclos de lectura y escritura de la memoria sean lo suficientemente cortos (rápidos) para que no disminuya el tiempo de operación de la computadora.

La primera presentación a analizar de la RAM es la SRAM o RAM estática: Las celdas de memoria de la RAM estática están construidas a partir de multivibradores biestables (flip-flops) y solo almacenan datos, indefinidamente, mientras el circuito esté siendo alimentado.

Por el otro lado se encuentran las DRAM o memorias RAM dinámicas. La diferencia más importante entre la SRAM y la DRAM es que las celdas que constituyen a ésta última retienen los datos solo por un tiempo limitado (por lo general, 2 ms), pasado el cual se pierde dicha información. Esta característica requiere una constante actualización[1] de los datos a intervalos regulares.

---

[1]Refresh: refresco, actualización

Lo anterior permite ver una desventaja de la DRAM, en tanto que las ventajas que pueden mencionarse son bajo consumo de energía y bajo costo, debido esto último a la simplicidad de construcción de sus celdas.

(Pueden hallarse más datos sobre el funcionamiento, especificaciones y estructura de estas memorias en la bibliografía de Tocci[2].)

A.2. TIPOS DE MEMORIA

A.2.1. MEMORIA CONVENCIONAL DEL DOS (0 k a 640 kB)

Ésta es la memoria que el DOS usa para

- "abrir" y "correr" aplicaciones
- manipular datos
- mover archivos
- copiar
- imprimir
- proveer la base "real" en la parte inferior de Windows
- servir a los requerimientos de memoria de Windows y DOS
- manejar operaciones generales

A.2.2. MEMORIA RESERVADA (640 kB a 1024 kB)

Esta área de memoria fue reservada por IBM para expansiones de hardware en curso y futuras:

- adaptadores de video
- adaptadores de red
- ROM BIOS
- otro hardware "mapeado" en memoria
- cargar programas sobre 640 kB

A.2.3. MEMORIA EXPANDIDA O PAGINADA

La memoria básica expandida o paginada (Paged Memory) emplea cuatro páginas contiguas de 16 kB de memoria para crear una ventana de 64 kB llamada un "marco de página" (page frame).

---

[2]Tocci, Ronald J. Sistemas Digitales, cap. 11

Una aplicación debe ser específicamente escrita para intercambiar (intercambio o conmutación entre bancos, bank-swicht) datos dentro y fuera de esta ventana. El programa por sí mismo emplea memoria convencional del DOS para funcionar y solamente accesa la memoria expandida para datos.

A.2.4. MEMORIA GENÉRICA EXTENDIDA (1024 kB a 16 MB)

Esta memoria no puede ser accesada por el DOS para uso estándar y es por tanto totalmente inútil para aplicaciones del DOS. Solamente puede ser usada como un espacio útil no protegido para programas de tipo RAM DISK y RAM CACHE y aplicaciones que "acaparen" la máquina, como Autocad2.

La memoria extendida es también usada por otros sistemas operativos tales como OS/2 y UNIX.

A.2.5. ESPECIFICACIÓN DE MEMORIA EXTENDIDA (XMS)

XMS provee tres tipos de memoria protegida virtual no convencional:

- UMB, bloques de memoria superior
- HMA, área de memoria alta
- EMB, bloques de memoria extendida

El DOS usa dos controladores para proveer el soporte de XMS:

- HIMEM.SYS, para HMA y EMB
- EMM386.EXE, para UMB y EMS

A.2.5.1. ÁREA DE MEMORIA SUPERIOR (640 kB a 1024 kB)

Pertenecientes al UMA (Upper Memory Area), los bloques de memoria superior, UMB, están en el área de memoria reservada no usada entre 640 kB y 1024 kB. La memoria es programada dentro de este espacio vacante por el manejador de memoria. Del área de 384 kB, 192 kB son "típicamente" no usados.

Una vez inicializada, es usada para cargar programas residentes de más de 640 kB, tablas del DOS, extensiones del DOS, manejadores de dispositivos, TSRs y software de redes.

Cargando programas en esta región, puede disponerse de más memoria convencional para DOS, WIndows y todas las aplicaciones.

A.2.5.2. ÁREA DE MEMORIA ALTA (1024 kB a 1088 kB)

El área de memoria alta HMA (High Memory Area), un "bug" (obstáculo) en el CPU 286 tan popular, continúa en 386, 486 y Pentium. HMA son los primeros 64 kB menos 16 B de memoria extendida a 1024 kB). Esta área puede ser usada por sólo un programa específicamente escrito para cargar parte de sí mismo dentro de esta región.

El uso más común actualmente es para el núcleo (kernel) del DOS (DOS=high). Otros programas que podrían usar este espacio alternativamente son DesqView, Novell NetWare, Tiara 10Net y Banyan VINES.

A.2.5.3. ÁREA DE MEMORIA EXTENDIDA (1088 kB y superior)

Los bloques de memoria extendida (EMB) son el remanente de la memoria extendida sobre el área de datos de HMA, a partir de 1088 kB.

Esta memoria es usada por programas de DOS de 16 MB en modo protegido como SMARTDRV y ambientes gráficos como Windows en los modos estándar o mejorado.

A.2.6. MEMORIA VIRTUAL (1 MB a 16 MB)

La memoria virtual es creada por Windows con el fin de usarla para intercambiar la entrada y salida de grandes bloques de datos usados por requerimientos de aplicaciones individuales.

Mediante la utilización de memoria extendida para accesar al espacio de disco duro como memoria física, Windows puede correr más programas simultáneamente.

A.3. SOFTWARE DEL DOS PARA MEMORIA

A.3.1. ACTIVACIÓN DE CONTROLADORES DE DISPOSITIVOS

Los controladores de dispositivos son rutinas de programas que contienen todo el código e instrucciones requeridas para que el sistema operativo controle un dispositivo periférico y éste opere correctamente. El DOS tiene un par de comandos para activar los controladores:

- DEVICE Carga en memoria el controlador de dispositivos especificado. Este comando sólo puede utilizarse en el archivo CONFIG.SYS.

- DEVICEHIGH Carga el controlador de dispositivos en el área de memoria superior. Al cargar un controlador de dispositivos en el área de memoria superior, queda libre más memoria convencional para otros programas. Si la memoria superior no está disponible, el comando DEVICEHIGH funcionará de la misma manera que el comando DEVICE. Este comando sólo puede utilizarse en el archivo CONFIG.SYS.

A.3.2. ADMINISTRADORES DE MEMORIA

A.3.2.1. HIMEM.SYS

HIMEM es un administrador de área de memoria extendida: un programa que coordina el uso de la memoria extendida de la PC, incluyendo el área de memoria alta (HMA) a fin de que no sea posible que dos aplicaciones o dos controladores de dispositivos utilicen la misma memoria simultáneamente.

HIMEM se instala agregando un comando <DEVICE> para HIMEM.SYS al archivo CONFIG.SYS. La línea del comando HIMEM deberá aparecer antes que cualquier comando que inicie aplicaciones o controladores de dispositivos que utilicen memoria extendida. Por ejemplo, HIMEM.SYS deberá aparecer antes que la línea de comando EMM386.

A.3.2.2. EMS

Las iniciales EMS se refieren a la especificación de memoria expandida, una técnica para expandir la memoria más allá de un megabyte en PC corriendo bajo DOS. EMS fue introducida en 1984 por Lotus, Intel y Microsoft, con memoria incrementada a 8 MB y en 1987 se presentó la versión 4.0, la cual incrementa el total de memoria con la que las aplicaciones DOS pueden trabajar desde uno hasta 32 MB, suministrando una capacidad de conmutación de bloques que permite a segmentos de la memoria convencional a la memoria EMS.

En máquinas 8086 y 286, el EMS se instala conectando una placa de memoria EMS y agregando un controlador EMS al DOS. En máquinas 386 y superiores, el EMS es activado agregando un controlador de software llamado EMM.

EMS (memoria expandida) y la memoria extendida (o ampliada) no son lo mismo. EMS puede ser instalado en todas las PCs incluyendo las 8086, mientras que la memoria extendida es memoria normal más allá de un megabyte en máquinas 286 y superiores. Este aspecto se detalla a continuación:

La memoria expandida EMS se diseñó cuando se vendieron muchísimas máquinas tipo XT (sin memoria ampliada) entre 1984 y 1987. Cuando las 286 se convirtieron en máquinas de límite inferior, la memoria ampliada se comenzó a explotar enormemente.

Aunque las 286 se pueden configurar tanto para memoria expandida como para memoria ampliada, hay que asignar la cantidad que se necesite de cada una de ellas. De esta forma, el remedio de la mayoría de las memorias flexibles es una 386 (SX o DX) o una 486, que convierte memoria ampliada en memoria expandida mediante programas. La que se posee depende de la aplicación. Algunas aplicaciones grandes requerían memoria expandida, mientras que otras la utilizaban si disponían de ellas.

A.3.2.3. EMM386.EXE

Es un controlador de dispositivos que proporciona acceso al área de memoria superior y utiliza la memoria extendida para simular la memoria expandida. Este controlador de dispositivos deberá ser cargado por un comando <DEVICE> en el archivo CONFIG.SYS y se podrá utilizar únicamente en PCs que tengan un procesador 386 o superior.

EMM386.EXE usa memoria extendida para simular memoria expandida en programas que puedan utilizar memoria expandida. EMM386.EXE también permite que se carguen programas y controladores de dispositivos en bloques de memoria superior (UMB).

A.3.2.4. EMM386

El EMM (Expanded Memory Manager) es un controlador de software para PC que administra la memoria expandida. En máquinas 8086 y 286, deben instalarse placas de memoria expandida. En máquinas 386 y 486, el hardware tiene capacidad de memoria expandida incorporada y sólo requiere el controlador o el administrador de memoria para activarla.

Habilita o inhabilita el acceso a la memoria expandida EMS en PCs con un procesador 386 o superior. No debe utilizarse este comando cuando se esté ejecutando Windows.

Activa el controlador de dispositivos EMM386.EXE si se especifica ON, lo desactiva si se especifica OFF o lo coloca en modo automático si se especifica AUTO. El modo automático habilita el acceso a la memoria expandida sólo cuando el programa la solicite. El valor predeterminado es ON.

A.3.2.5. RAMDRIVE.SYS

RAMDRIVE.SYS es un controlador de dispositivo del sistema operativo que usa parte de la memoria RAM (memoria de acceso aleatorio) de la PC como si fuera una unidad de disco duro. Este controlador de dispositivos deberá ser cargado por un comando <DEVICE> o <DEVICEHIGH> en el archivo CONFIG.SYS.

Para las unidades de RAM puede emplearse memoria convencional, extendida o expandida. Estas unidades son mucho más rápidas que las unidades del disco duro, ya que la PC puede leer información de esta unidad de memoria más rápidamente dado que no hay dispositivos mecánicos que hagan lento el acceso.

La unidad de RAM parece ser una unidad normal de disco duro y puede usarse de la misma manera. La principal diferencia entre una verdadera unidad de disco y una unidad de RAM es que la información se perderá al apagarse o reiniciarse la computadora, ya que dicha información sólo existe en la memoria RAM. Pueden establecerse tantas unidades de RAM como se deseen, hasta ocupar toda la memoria de la computadora. Para ello se agrega al archivo CONFIG.SYS una línea RAMDRIVE.SYS por cada unidad de RAM adicional que se desee.

A.4. SOFTWARE DEL OS/2 PARA MEMORIA

(Al igual que en el caso del DOS, solo se mencionan los aspectos de más interés.)

A.4.1. CONMUTACIÓN ENTRE SISTEMAS OPERATIVOS

A.4.1.1. PROTECTONLY

Selecciona uno o dos entornos operativos. Este mandato debe incluirse en el archivo CONFIG.SYS; no debe ejecutarse en la línea de mandatos del OS/2.

El sistema operativo OS/2 requiere esta sentencia en el archivo CONFIG.SYS. La sentencia PROTECTONLY=YES permite que la memoria por debajo de los 640 KB, utilizada generalmente para programas de DOS, esté disponible para programas de OS/2. Cuando PROTECTONLY=YES, no se pueden ejecutar programas en sesiones de DOS. Si más adelante se decide ejecutar programas de DOS en los 640 KB inferiores de la memoria, debe especificarse PROTECTONLY=NO. Esto permite utilizar tanto programas de DOS como de OS/2.

A.4.2. CONTROLADORES DE DISPOSITIVOS (PARA MEMORIA)

Los comandos DEVICE del OS/2 y del DOS operan en la misma forma, pero no se tiene la variante DEVICEHIGH para el OS/2.

Siempre que se añadan o realicen cambios en el archivo CONFIG.SYS, debe volverse a iniciar el sistema para que el nuevo dispositivo o modificación entre en vigor. El archivo CONFIG.SYS sólo se lee durante el arranque del sistema. Cualquier modificación a este archivo después de la inicialización del sistema no entrará en vigor hasta que vuelva a iniciarse el sistema.

A.4.2.1. VEMM.SYS

Proporciona el Gestor de Memoria Expandida (EMM) para sesiones del DOS. Es un controlador de dispositivo virtual que proporciona la emulación de la Especificación de Memoria Expandida (EMS) para las sesiones del DOS y opera como se indicó anteriormente para el EMM.

A.4.2.2. VXMS.SYS

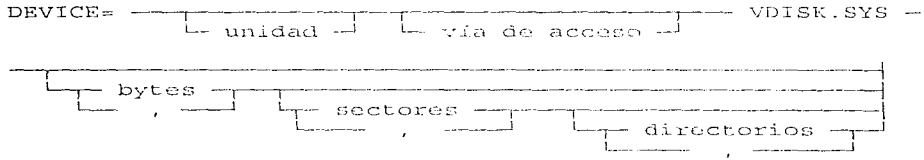
Proporciona la Especificación de Memoria Ampliada (XMS) para sesiones del DOS. Es un controlador de dispositivo virtual que proporciona la emulación de la Especificación de Memoria Ampliada (XMS) para sesiones del DOS. XMS permite que las aplicaciones del DOS accedan a más de 1 MB de memoria, bajo el control de la XMS.

Las sentencias DEVICE=C:\OS2\MDOS\VXMS.SYS deben colocarse después de las sentencias DEVICE=C:\OS2\MDOS\VEMM.SYS en el archivo CONFIG.SYS ya que VXMS reserva todas las direcciones disponibles entre 640 kB y 1 MB para utilizarlas como bloques de memoria superior (UMB). VXMS no se instalará si algún otro controlador de dispositivo ha reservado la región de 1 MB a 1 MB + 64 kB.

A.4.2.3. DISK.SYS

Este comando es similar al RAMDISK del DOS e instala un disco virtual. En este caso se presentan más detalles del mismo dado que es la opción que fue empleada en el proyecto.

Cualquier sentencia DEVICE=C:\OS2\VDISK.SYS debe listarse después de cualquier sentencia DEVICE=C:\OS2\EXTDSKDD.SYS en el archivo CONFIG.SYS para evitar que afecte a las asignaciones de letras de unidad lógica.



Los parámetros bytes, sectores y directorios son parámetros posicionales. Basta una coma si no desea alterar el valor por omisión del parámetro para el cual se utiliza la coma. Esto indica al sistema que pase por alto el valor que está buscando en esta posición, que inserte el valor por omisión y que siga con el siguiente valor. Las siguientes tablas presentan los valores por omisión y ejemplos:

VALORES POR OMISIÓN DE LOS PARÁMETROS	
Bytes	64
Sectores	128
Directorios	64

EJEMPLOS			
Bytes [kB]	Sectores [bytes]	Directorios [entradas]	Enunciado
160	128	64	DEVICE=C:\OS2\VDISK.SYS 160 128 64
160	--	--	DEVICE=C:\OS2\VDISK.SYS 160,,
--	--	32	DEVICE=C:\OS2\VDISK.SYS ,,32
160	--	64	DEVICE=C:\OS2\VDISK.SYS 160,64

VDISK muestra un mensaje para indicar que está instalado y da la letra de la unidad que está asignada al disco virtual.

Si el tamaño del disco virtual especificado es demasiado grande para que quepa en el almacenamiento, VDISK intentará hacer un disco virtual de 16 KB. Esto puede producir un disco virtual con un número diferente de entradas de directorio del que se especificó para directorios.

A.5. LA MEMORIA EN DIFERENTES SISTEMAS OPERATIVOS

En Windows 3.xx, interfaz gráfica de 16 bits que complementa a DOS, cuando una aplicación requería memoria, el sistema le permitía tomar la memoria disponible, siguiendo este proceso con cualquier otra aplicación que así lo requiriera; sin embargo, al cerrar una aplicación, no se cerraban las bibliotecas (libraries) de la misma, por lo que continuaban ocupando el espacio de memoria y no permitían entonces poder "correr" otra aplicación.

Debido a que Windows 3.xx no es realmente un sistema operativo, no es capaz de administrar por sí mismo la memoria, y lo hace a través de algunos programas como HIMEM.SYS y EMM386.EXE, los cuales son parte del sistema operativo DOS.

Windows NT y OS/2, como sistemas protegidos de 32 bits, toman el control de la memoria y "protegen" la empleada por cada aplicación: Cuando una aplicación va a ser ejecutada, el sistema averigua cuánta memoria va a requerirse y la proporciona si se

encuentra disponible, haciendo lo mismo con otras aplicaciones en esa circunstancia; no obstante, al "cerrar" la aplicación, el sistema cierra también las correspondientes bibliotecas y toma nuevamente el control de esa memoria para posteriores requerimientos. Por otra parte, cuando una aplicación está empleando un área de memoria, el sistema se encarga de que ninguna otra la ocupe, i.e., la "protege", consiguiendo con esto evitar los errores de "crash" comunes en Windows 3.xx.

El sistema operativo Windows NT, que maneja un caché de 16 MB de RAM, toma todo el control sobre la memoria existente, administrándola entre las aplicaciones que lo requieran protegiéndola para evitar conflictos de memoria entre ellas. No permite al usuario la creación de unidades de disco de RAM ("discos virtuales") debido a que internamente el sistema operativo no cuenta con esa opción.

Windows NT y OS/2 son sistemas operativos cuyos códigos son en su totalidad de 32 bits, a diferencia de Windows 95 que, para efectos de compatibilidad, aún conserva códigos de 16 bits. En Windows 95 pueden emplearse de esta forma archivos como AUTOEXEC.BAT y CONFIG.SYS, que en principio no son necesarios para el sistema. De tenerse, estos archivos harían lo que les correspondiera al iniciar el sistema y no se regresaría a ellos a menos que el usuario así lo determinara. Al correr sesiones de DOS en ventana, por ejemplo, se vería la influencia de esos archivos: el caso es a la inversa que con Windows 3.xx, es decir, Windows 95 es aquí el sistema operativo y las ventanas de DOS son una emulación de esa interfaz y sistema.

Del modo indicado en el párrafo anterior, sería téoricamente posible el manejo de los denominados "discos virtuales" en Windows 95, constituyendo así espacios de memoria que el sistema respetaría, aunque las especificaciones del fabricante indican que la opción conocida en DOS como RAMDRIVE (o RAMDISK en otros casos) ha sido abandonada en Windows 95, probablemente por no ser un recurso muy socorrido.

Cabe indicar que, por ser un sistema operativo multitareas, Windows 95 puede correr diversas aplicaciones de DOS en sus respectivas "ventanas" y cada una de ellas "reconocería" emplear un área de memoria convencional de 640 kBytes, que en realidad ha sido asignada a cada una por el sistema.

Por su parte, OS/2 no puede correr aplicaciones de 16 bits y, en su lugar, realiza emulaciones de DOS y Windows. Windows 95 lo hace también de alguna forma, pues aunque cuenta con un DOS nativo, lo "corre" en una ventana sin dejarle tener el control total sobre el sistema, sino más bien administrando a la interfaz.

Windows NT no reconoce las denominaciones habituales de memoria: convencional (640 kB desde 0000h hasta FFFFh), superior (desde el límite superior de los anteriores 640 kB hasta 1 MB) y extendida (desde el límite superior de 1 MB hasta 16 MB en aplicaciones de 16 bits), sino que agrupa a todas ellas en el término de "memoria plana", sin definir divisiones en todo el rango de memoria disponible y "remapea" los segmentos según lo solicite la aplicación.

APÉNDICE B

DATOS TÉCNICOS

Físicamente, la interfaz de cada puerto tiene asociados 12 pines del tipo I/O[1], enunciados en la tabla siguiente, en donde n es el número del puerto:

SEÑAL	PINES	DESCRIPCION
CREQn	1	señal de petición de token
CACKn	1	señal de reconocimiento de petición de token[2] [3]
CSTRBn	1	señal de destello de datos[4]
CRDYn	1	señal de disponibilidad (ready signal)
CnD(7-0)	8	bus de datos

[5]

El puerto de comunicaciones transmite cada una de las palabras de 32 bits guardadas en su FIFO de salida con base en un módulo de "byte a byte". Debido a que las líneas de control y de datos son bidireccionales, cada C4x debe tener la pertenencia del bus de datos del puerto de comunicaciones antes de comenzar una transferencia de palabras. El término *token* designa la pertenencia del bus: al puerto de comunicaciones que tiene el token le pertenece el bus de datos y puede así transmitir datos. [6]

CREQn Un C4x activa esta señal para solicitar el uso del bus de datos del puerto de comunicaciones.
CACKn Un C4x activa esta señal al dejar de tener la pertenencia del bus de datos del puerto de comunicaciones, habiendo recibido un CREQn de otro C4x.

---

[1]En la interfaz de los puertos externos pueden contarse 16 pines físicos, lo cual no contradice la afirmación sobre los del tipo I/O. El diagrama de los puertos externos se muestra en este mismo apéndice.

[2]Token: señal, ficha, vale; se manejará el término original

[3]token request aknowledge signal

[4]data strobe signal

[5]UG, cáp. 14, pág. 3-23

[6]UG pág. 8-5

CSTRBn Un C4x transmisor activa esta señal para indicar que ha colocado un byte de datos válido en el bus de datos del puerto de comunicaciones.
CRDYn Un C4x receptor activa esta señal para indicar que ha recibido un byte de datos vía el bus de datos del puerto de comunicaciones.
CnD(7-0) Este bus transporta datos bidireccionalmente (en una base de bytes) entre dos C4xs ó entre un C4x y otro dispositivo.

Direcciones de los puertos

PUERTO	CPCR, registros de control del puerto de comunicaciones	Puerto de entrada, posición FIFO0	Puerto de salida, posición FIFO7
0	0010 0040h	0010 0041h	0010 0042h
1	0010 0050h	0010 0051h	0010 0052h
2	0010 0060h	0010 0061h	0010 0062h
3	0010 0070h	0010 0071h	0010 0072h
4	0010 0080h	0010 0081h	0010 0082h
5	0010 0090h	0010 0091h	0010 0092h

Termina en 0010 009Fh. [7]

En la figura B.6. se muestra el mapa de memoria de los periféricos; con respecto a dicha figura, es conveniente indicar que en el sistema de los DSPs se maneja que una palabra es equivalente a 32 bits[8] (1 word = 32 bits), por lo que cada una de las direcciones, e.g. 0010 0050h y 0010 0051h, es una dirección de 32 bits. Por consiguiente, las 16 palabras de cada puerto implican un total de (16 palabras) x (32 bits) = 512 bits.

[7] UG pág. 3-23

[8] UG pág. 3-18

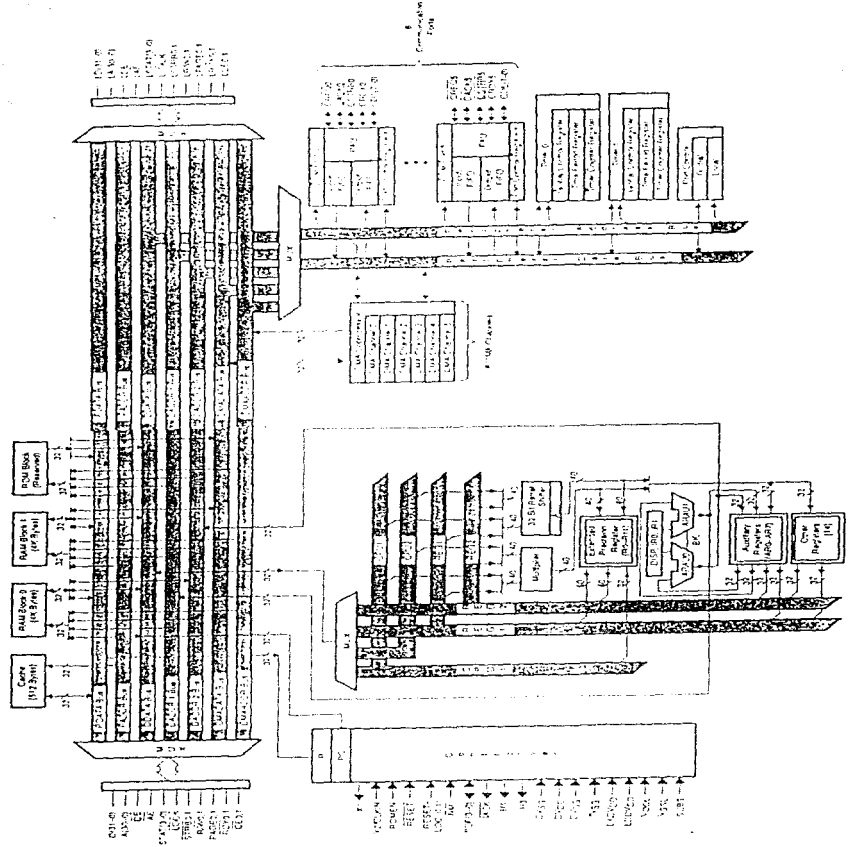


Figura B.1. Diagrama a bloques del TMS320C40

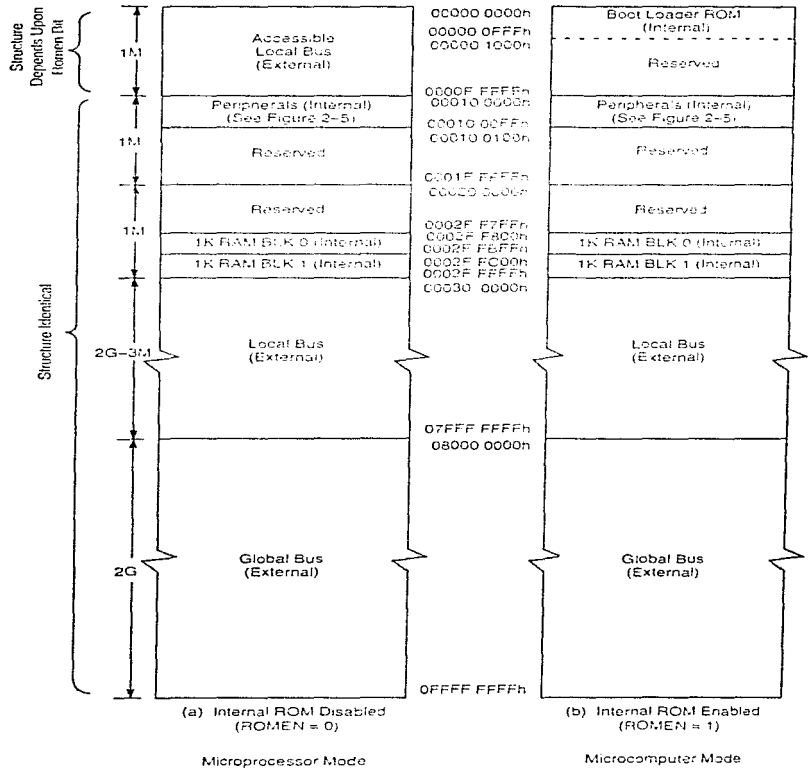


Figura B.2. Mapas de memoria

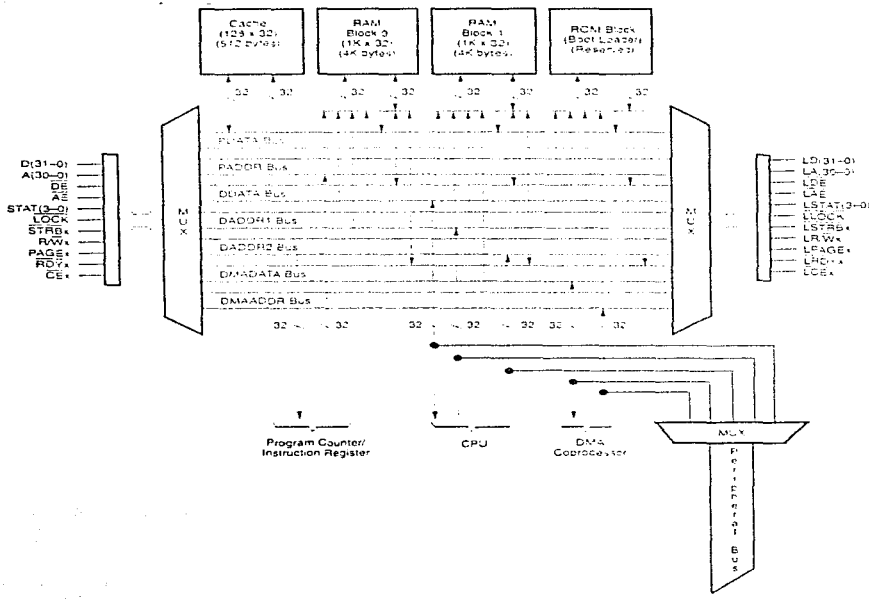


Figura B.3. Organización de la memoria

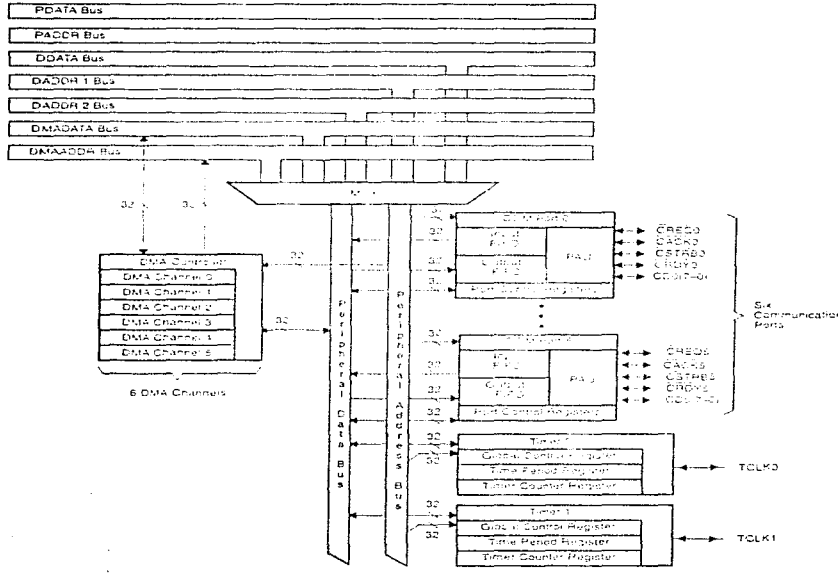


Figura B.4. Módulos periféricos

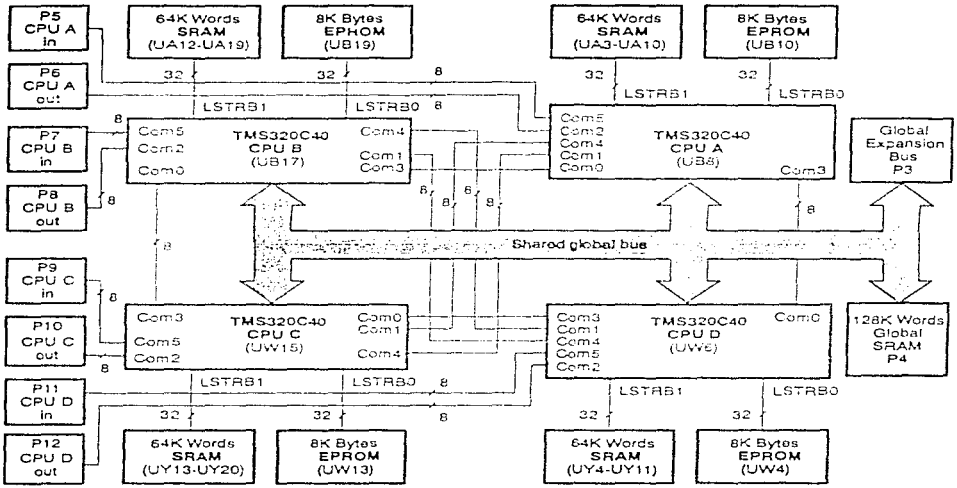


Figura B.5. Diagrama a bloques del TMS320C4x PPDS

Address	Peripheral
0010 0000h 0010 000Fh	Local and Global Port Control (16 words)
0010 0010h 0010 001Fh	Analysis Block Registers (16 words)
0010 0020h 0010 002Fh	Timer 0 Registers (16 words)
0010 0030h 0010 003Fh	Timer 1 Registers (16 words)
0010 0040h 0010 004Fh	Communication Port 0 (16 words)
0010 0050h 0010 005Fh	Communication Port 1 (16 words)
0010 0060h 0010 006Fh	Communication Port 2 (16 words)
0010 0070h 0010 007Fh	Communication Port 3 (16 words)
0010 0080h 0010 008Fh	Communication Port 4 (16 words)
0010 0090h 0010 009Fh	Communication Port 5 (16 words)
0010 00A0h 0010 00AFh	DMA Coprocessor Channel 0 (16 words)
0010 00B0h 0010 00BFh	DMA Coprocessor Channel 1 (16 words)
0010 00C0h 0010 00CFh	DMA Coprocessor Channel 2 (16 words)
0010 00D0h 0010 00DFh	DMA Coprocessor Channel 3 (16 words)
0010 00E0h 0010 00EFh	DMA Coprocessor Channel 4 (16 words)
0010 00F0h 0010 00FFh	DMA Coprocessor Channel 5 (16 words)

Figura B.6. Mapas de memoria de los periféricos

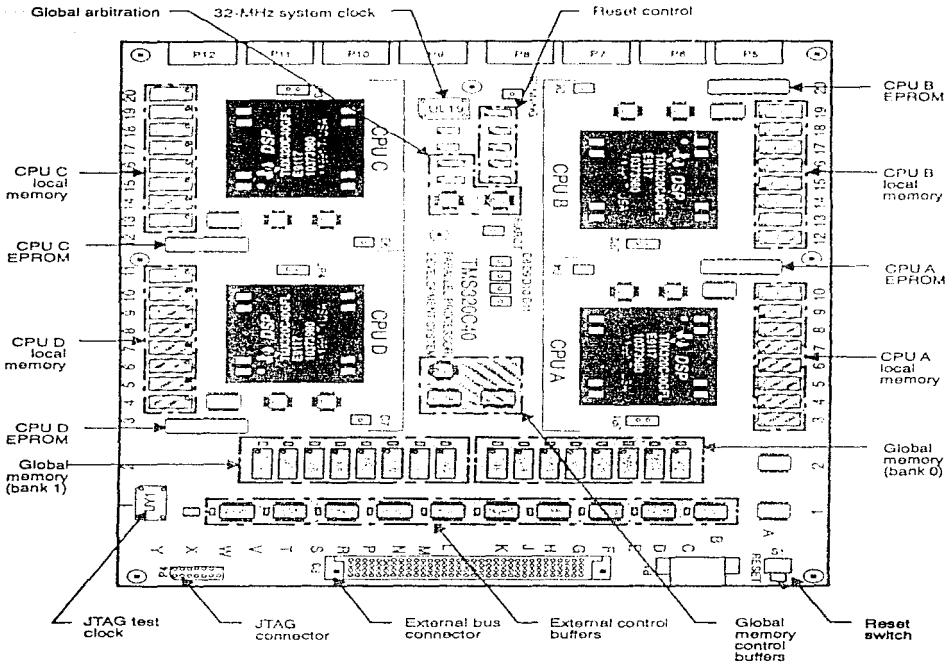


Figura B.7. Diagrama de la tarjeta del PPDS

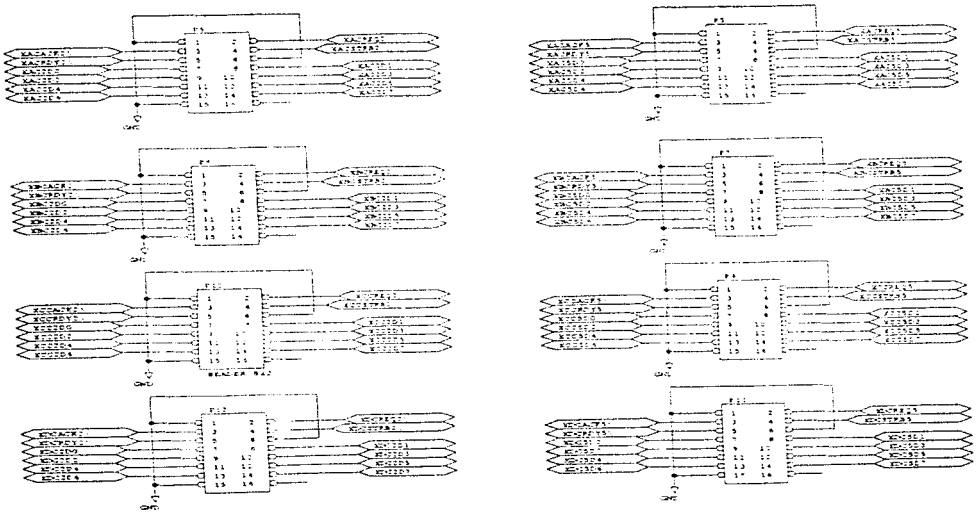


Figura B.8. Puertos externos de comunicaciones del PPDS

APÉNDICE C

CÓDIGO FUENTE DE PROGRAMAS

PRIMERA ETAPA EN LOS DSPs (VISUAL1.ASM)

```
*************************************************************************
*       Inicialización de la memoria de datos interna                   *
*************************************************************************
BLK0    .word   02FF800H        ;dirección de inicio del bloque 0 de RAM
BLK1    .word   02FFC00H        ;dirección de inicio del bloque 1 de RAM
        LDI     @BLK0,AR0       ;AR0 apunta al bloque 0
        LDI     @BLK1,AR1       ;AR1 apunta al bloque 1
        LDF     0.0,R0          ;0->R0
        RPTS    1023            ;ejecuta 1024 veces las próx. inst.
                                ;o el próx. bloque de instrucciones
        STF     R0,*AR0++(1)    ;0->B0
        STF     R0,*AR1++(1)    ;0->B1
*************************************************************************
*       Declaración de variables                                        *
*************************************************************************
        .global X,Y             ;
        .hword  A,B,C,D         ;las declara de 16 bits
        .hword  DEN             ;(") DEN = A+B+C+D
        .hword  NUMX, NUMY      ;(")
*************************************************************************
*       Inicia el programa                                              *
*************************************************************************
ciclo   LDF     @100041, A
||      LDF     @100051, B
        LDF     @100061, C
||      LDF     @100071, D
        ADDF3   A,B,R1          ;A+B --> R1
        ADDF3   C,D,R2          ;C+D --> R2
        ADDF3   R1,R2,DEN       ;A+B+C+D --> DEN
        ADDF3   B,C,R3          ;B+C --> R3
        ADDF3   A,D,R4          ;A+D --> R4
        SUBF3   R2,R1,NUMX      ;R1-R2 --> NUMX, A+B-(C+D) --> NUMX
        SUBF3   R4,R3,NUMY      ;R3-R4 --> NUMY, B+C-(A+D) --> NUMY
*       RCPF    DEN,R0          ;1/DEN --> R0  Se sustituye por lo sig.
        LAJU    INVF            ;salto incond. a subrutina para exactitud
        LDF     DEN,R0
        NOP                     ;pueden ser otras instrucciones
        NOP                     ;non-pipeline-break
                                ;termina subrutina, los NOPs se requieren
                                ;por características de LAJcond
        MPYF3   R1,NUMX,X       ;NUMX/DEN --> X
        MPYF3   R1,NUMY,Y       ;NUMY/DEN --> Y
        ADDF    200H,X          ;X+512 --> X
        ADDF    200H,Y          ;Y+512 --> Y
*
        BR      ciclo
```

```
*************************************************************************
*        Subrutina para exactitud                                       *
*************************************************************************
* INVERSO DE UN NÚMERO FLOTANTE
* CON EXACTITUD DE 32 BITS EN LA MANTISA
*
* SUBRUTINA INVF
*
* Asignación de argumentos :
*
*    Argumento          Función
*  ------------------------------------------------------
*    R0               v = número flotante al que se le hallará
*                         el recíproco (recibido en la llamada)
*    R1               1/v  resultado (enviado)
*
* Registro usado como entrada :          R0
* Registros modificados :                R1, R2
* Registro conteniendo el resultado :    R1
* Registro para llamado de subrutina :   R11
*
* Ciclos:   8            Palabras:   8
*
        .global    INVF
*
INVF:   RCPF       R0, R1      ;toma x(0) = la estimación de 1/v, R0 = v
*
        MPYF3      F1,R0,R2
        SUBRF      2.0,R2
        MPYF       R2,R1       ;fin de la primera iteración
                               ;(exactitud de 16 bits)
*
        BUD        R11         ;retorno atrasado al llamador
        MPYF3      R1,R0,R2
        SUBRF      2.0,R2
        MPYF       R2,R1       ;fin de la segunda iteración
                               ;(exactitud de 32 bits)
*
*       R1 = 1/v, regresa al llamador
*
        .end
```

ARCHIVO DE PROCESO POR LOTES (VISUAL.CMD)
(Equivalente en OS/2 a visual.bat)

```
cd\C4XHLL
copy G10-150.EXE d:
copy K d:
copy A d:
d:
g10-150.exe
```

VISUALIZADOR (GRAF10.C)

```
#include "stdio.h"
#include "stdlib.h"
#include "ctype.h"
#include "time.h"
#include "graphics.h"
#include "conio.h"
#include "bios.h"
#include "global.h" /* Macros, prototipos, estructuras, variables globales */


/*=============================================================================*/
/* MAIN(): PROGRAMA PRINCIPAL                                                  */
/*=============================================================================*/
void main(void)
{
     int opcion=1;
     int fin=0;

     init_graphics();      /* prepara el modo gráfico                          */
     atencion();           /* despliega un mensaje de atención (alerta)        */
     frames();             /* presentación de las partes permanentes           */
     openhelp();           /* abre y despliega el archivo de ayuda             */

     for(;fin!=1;)
     {
        if (kbhit())
        {
         opcion=getch();
         command(&opcion,&fin);       /* proceso de comandos del usuario      */
        }
     }
     setviewport(0,0,639,349,ON);     /* limpia la ventana al salir           */
     clearviewport();
}


/*=============================================================================*/
/* INIT_GRAPHICS():                                                            */
/* Inicializa los gráficos y muestra las partes permanentes de la pantalla     */
/*=============================================================================*/
void init_graphics(void)
{
     char ch;
     clrscr();

/*-----------------------------------------------------------------------------*/
/* En la siguiente etapa, la función detectgraph proporciona los valores      */
/* para el controlador y modo gráfico.                                         */
/*-----------------------------------------------------------------------------*/
     GraphDriver = GraphMode = DETECT;
     detectgraph(&GraphDriver, &GraphMode);      /* ¿qué controlador se usa? */
     if ((GraphDriver == EGA) || (GraphDriver == VGA))
     {
       GraphMode = EGAHI;
       registerbgidriver(EGAVGA_driver);
     }
```

```
    else if (GraphDriver == CGA)
      {
       printf("¿Se tiene un monitor Compaq de Plasma? (S o N):");
          ch = getche();
          if ( (ch == 's') || (ch == 'S') )
          {
              GraphDriver = ATT400; GraphMode = ATT400HI;
              registerbgidriver(ATT_driver);
          }
          else
          {
              display_error();
          }
      }
      else
          display_error();

    initgraph(&GraphDriver, &GraphMode, "");
    if ( graphresult() != grOk )   display_error();
/*-------------------------------------------------------------------------*/
/* Se asegura que el monitor tenga la resolución adecuada                  */
/*-------------------------------------------------------------------------*/
    if (getmaxx() < 639 || getmaxy() < 349)  display_error();
}


/*=========================================================================*/
/* DISPLAY_ERROR(): Presenta un mensaje de error si no inicializa gráficos */
/*=========================================================================*/
void display_error(void)
{
      printf("\nRequiere un monitor VGA, EGA, or AT&T 640x400 \n");
      exit(1);
}


/*=========================================================================*/
/* FRAMES(): Presenta todas las ventanas permanentes                       */
/*=========================================================================*/
void frames(void)
{
    char **entry = main_menu;              /* presenta el menú de opciones */
    setcolor(WHITE);                     /* color blanco para texto y marco */
/* Si se usa un monitor Plasma 2 a color, el color de relleno debe ser     */
/* negro, en otro caso, basta con azul                                     */
    if (GraphDriver != ATT400)
     setfillstyle(SOLID_FILL,BLACK);
    else
     setfillstyle(SOLID_FILL,BLUE);
/*-------------------------------------------------------------------------*/
/* ventana de despliegue del visualizador                                  */
/*-------------------------------------------------------------------------*/
    setviewport(321,12,639,158,ON);            /* dimensiones de la ventana */
    bar(0,0,318,146);                          /* rellena de azul           */
    rectangle(0,0,318,146);                    /* bordes                    */
    moveto(2,2);                               /* ubica el cursor           */
    gputs("              Señal");              /* título                    */
    setcolor(WHITE);
```

```
/*-------------------------------------------------------------------*/
/* sección para la señal, en la ventana del visualizador             */
/*-------------------------------------------------------------------*/
    setviewport(323, 27, 598, 156, ON);        /* dimensiones de la ventana */
    clearviewport();                           /* sección de la ventana     */
    rectangle(0,0,274,129);                    /* bordes                    */
/*-------------------------------------------------------------------*/
/* ventana de titulo                                                 */
/*-------------------------------------------------------------------*/
    setviewport(0,0,639,11,ON);                /* dimensiones de la ventana */
    bar(0,0,639,11);                           /* rellena de azul           */
    rectangle(0,0,639,11);                     /* bordes                    */
    moveto(2,2);                               /* ubica el cursor           */
    outtextxy(525,2,"J. S. Oshino");
    gputs("                        VISUALIZADOR PARA EL MEPSICRON");
/*-------------------------------------------------------------------*/
/* ventana de menú                                                   */
/*-------------------------------------------------------------------*/
    setviewport(0,13,320,158,ON);              /* dimensiones de la ventana */
    bar(0,0,320,146);                          /* rellena de azul           */
    rectangle(0,0,320,146);                    /* bordes                    */
    moveto(2,2);                               /* ubica el cursor           */
    gputs("                    Menú");         /* titulo de la ventana      */
    setcolor(WHITE);                           /* color para el menú        */
    while(*entry) gputs(*entry++);             /* opciones del menú         */
}


/*===================================================================*/
/* ATENCION(): Mensaje de alerta para el usuario no experto          */
/*===================================================================*/
void atencion(void)
{
    int orden=0;
    setfillstyle(SOLID_FILL,RED);
    setcolor(WHITE);
    setviewport(80,100,530,200,ON);        /* define la ventana         */
    clearviewport();                       /* limpia la sección         */
    bar(0,0,450,100);                      /* rellena de rojo           */
    rectangle(0,0,450,100);                /* bordes                    */
    moveto(2,2);                           /* ubica el cursor           */
    gputs("");
    gputs("                        Atención:");
    gputs("");
    gputs(" ¡No intente subir el valor de voltaje a más de 4.7 kV!");
    gputs("");
    gputs(" ¡Puede dañar el sistema seriamente!");
    gputs("");
    gputs(" Llame al personal experto en el sistema si tiene dudas.");
    gputs("");
    gputs("");
    gputs("                                  Presione C para continuar");
    if (kbhit()) orden=getch();
    while(orden!=67&&orden!=99) orden=getch();   /* espera por Continuar */
    clrscr();
}
```

```
/*=============================================================================*/
/* GPUTS(): Equivalente gráfico de PUTS()                                      */
/*=============================================================================*/
void gputs(char *s)
{
     outtext((char far *) s);                    /* imprime el texto           */
     moveto(2, gety() + 8);                      /* "retorno de carro"         */
}


/*=============================================================================*/
/* STATUS(): Inicializa la ventana de estado del sistema y la señal            */
/*=============================================================================*/
void status(void)
{
     setfillstyle(SOLID_FILL,BLUE);
     setcolor(WHITE);
     setviewport(0,159,639,349,ON);          /* mitad inferior de la pantalla  */
     clearviewport();                        /* limpia la sección              */
     setviewport(0, 159, 360, 349,ON);       /* define la ventana              */
     bar(0,0,360,190);                       /* rellena de azul                */
     rectangle(0,0,360,190);                 /* bordes                         */
     moveto(2,2);                            /* ubica el cursor                */
     gputs("                    Estado");
}


/*=============================================================================*/
/* ESTADO(): Actualiza y despliega el estado del sistema y la señal            */
/*=============================================================================*/
void estado(char *fuente)
{

     switch(*fuente)
     {
/*-----------------------------------------------------------------------------*/
/* Estado active en demostración de adquisición                                */
/*-----------------------------------------------------------------------------*/
     case '1':
        setviewport(0, 159, 360, 349,ON);            /* define la ventana      */
        moveto(2,10);
        setcolor(WHITE);
        sprintf(string, " Activo (demostración de adquisición)");
        gputs(string);
        gputs(" ");
        setcolor(LIGHTGREEN);
        gputs(" Obteniendo imagen");
        gputs(" ");
        setcolor(WHITE);
        sprintf(string," Niveles: ");
        outtext(string);
        setcolor(BROWN);
        outtext(" 8");
        setcolor(LIGHTGRAY);
        outtext(" 9");
        setcolor(LIGHTBLUE);
        outtext(" A");
        setcolor(RED);
```

```
        outtext(" ");
        setcolor(GREEN);
        outtext(" ");
        setcolor(MAGENTA);
        outtext(" ");
        setcolor(YELLOW);
        gputs(" ");
        gputs(" ");
        setcolor(WHITE);
        sprintf(string," Nivel máximo: ");
        outtext(string);
        setcolor(BROWN);
        outtextxy(120,58," ");
        break;
/*-------------------------------------------------------------------------*/
/* Estado activo en demostración de imagen final                           */
/*-------------------------------------------------------------------------*/
    case '2':
        setviewport(0, 159, 360, 349,ON);          /* define la ventana    */
        moveto(2,10);
        setcolor(WHITE);
        sprintf(string, " Activo (demostración de imagen final)");
        outtext(string);
        break;
/*-------------------------------------------------------------------------*/
/* Estado activo real                                                      */
/*-------------------------------------------------------------------------*/
    case 'i':
        setviewport(0,12,639,349,ON);      /* pantalla total para despliegue */
        clearviewport();
        setcolor(WHITE);
        outtextxy(0,0," Nivel máximo: ");
        setcolor(BROWN);
        outtextxy(120, 0, " ");
        break;
/*-------------------------------------------------------------------------*/
/* Estado de grabación o de reproducción                                   */
/*-------------------------------------------------------------------------*/
    case 'g':
        setviewport(0, 159, 360, 349,ON);          /* define la ventana    */
        setcolor(LIGHTGREEN);
        gputs("");
        gputs("");
        gputs(" Guardando...");
        break;
    case 'r':
        setviewport(0, 159, 360, 339,ON);          /* define la ventana    */
        setcolor(LIGHTGREEN);
        gputs("");
        gputs("");
        gputs(" Desplegando imagen previa");
        break;
/*-------------------------------------------------------------------------*/
/* Estado de cambio de escala de imagen desplegada                         */
/*-------------------------------------------------------------------------*/
    case '/':
        setviewport(0, 159, 360, 349,ON);          /* define la ventana    */
        moveto(2,10);
```

```
        setcolor(LIGHTGREEN);
        gputs("");
        gputs(" Preparando aumento de escala");
        setcolor(WHITE);
        gputs("");
        gputs(" Proporcione el valor del factor de escala");
        gputs(" (desde 1 hasta 10) y presione ENTER");
        gputs("");
        gputs(" La imagen aparecerá en pantalla completa");
        gputs(" Después presione C para continuar");
        break;
/*-------------------------------------------------------------------------*/
/* Estado de espera                                                        */
/*-------------------------------------------------------------------------*/
    default:
        setviewport(0, 155, 360, 349,ON);              /* define la ventana */
        moverel(0,8);
        gputs(" Esperando instrucción");
    }
    setviewport(325,28,596,155,ON);      /* sección superior de la pantalla */
}


/*=========================================================================*/
/* COMMAND(): proceso de la opción del usuario                             */
/*=========================================================================*/
void command(int *haz, int *acaba)
{
    int c;

    setviewport(90,135,120,145,ON);                    /* prepara la ventana */
    setfillstyle(SOLID_FILL,LIGHTGRAY);
    bar(0,0,30,10);                                    /* rellena de gris    */
    moveto(2,2);
    setcolor(RED);                                     /* texto rojo         */

    switch(*haz)
    {
    case 65:
    case 97:
      outtext(" A");                        /* lee imagen de archivo binario   */
      status();
      estado("r");
      open_file("rb");                      /* abre y lee archivo binario      */
      viewimg();
      break;
    case 66:
    case 98:
      outtext(" B");                        /* borra la ventana de señal, i.e. */
      setviewport(325,28,596,155,ON);       /* sección superior de la pantalla */
      clearviewport();
      break;
    case 67:
    case 99:
      outtext(" C");                        /* indica continuación del proceso */
      break;
    case 68:
    case 100:
```

```
      outtext(" D");                    /* indica detención del proceso     */
      break;
    case 71:
    case 103:
      outtext(" G");                    /* guarda imagen en archivo binario */
      status();
      estado("g");
      open_file("wb");                  /* nuevo archivo para escritura     */
      saveimg();
      break;
    case 73:
    case 105:
      outtext(" I");                    /* adquisición real desde los DSPs  */
      estado("i");
      realcase();
      frames();                         /* presenta las partes permanentes  */
      break;
    case 77:
    case 109:
      outtext(" M");                    /* demostración de adquisición      */
      status();
      estado("1");                      /* pantalla de estado de demo.      */
      demo1();
      break;
    case 78:
    case 110:
      outtext(" N");                    /* demostración de imagen final     */
      status();
      estado("2");                      /* pantalla de estado de demo.      */
      demo2();
      break;
    case 43:                            /* tecla +                          */
    case 45:                            /* tecla -                          */
      outtext("+/-");
      status();
      estado("/");                      /* pantalla de cambio de escala     */
      zoom2();
      c=getch();
      frames();                         /* presenta las partes permanentes  */
      break;
    case 89:
    case 121:
      outtext(" Y");                    /* abre archivo de ayuda            */
      openhelp();
      break;
    case 9:
      *acaba=1;                         /* con TAB termina                  */
      break;
    default:
      outtext(" ¿?");                   /* para cualquier otra tecla        */
      break;
    }
    status();                           /* limpia la pantalla de estado     */
    estado("e");                        /* indica estado de espera          */
}
```

```
/*=========================================================================*/
/* OPENHELP(): Abre y despliega el archivo de ayuda                        */
/*=========================================================================*/
void openhelp(void)
{
     int orden;
     FILE *ayuda;
     /*-----------------------------------------------------------------*/
     /* abre el archivo en el directorio en uso                         */
     /*-----------------------------------------------------------------*/
     if((ayuda=fopen("k","r"))==NULL)
     {
       printf("Nó se puede abrir el archivo de ayuda \n");
       exit(0);
     }
     setviewport(0,159,639,349,ON);
     clearviewport();
     setcolor(LIGHTGRAY);
     moveto(2,2);
     while(!feof(ayuda)) /* */
     {
       if(fgets(string,2000,ayuda)) gputs(string);
     }
     do
     {
        orden=getch();
     }while(orden!=67&&orden!=99);        /* espera la orden Continuar      */
     fclose(ayuda);
}


/*=========================================================================*/
/* DEFINECOLOR(): Selecciona el color según el valor de conteo             */
/*=========================================================================*/
void definecolor(unsigned int *valor)
{
      if (*valor==0                    ) seudocolor=BLACK;
      if (*valor>=1      &&*valor<10000) seudocolor=BROWN;
      if (*valor>=10000 &&*valor<20000) seudocolor=LIGHTGRAY;
      if (*valor>=20000 &&*valor<30000) seudocolor=LIGHTBLUE;
      if (*valor>=30000 &&*valor<40000) seudocolor=RED;
      if (*valor>=40000 &&*valor<50000) seudocolor=GREEN;
      if (*valor>=50000 &&*valor<60000) seudocolor=MAGENTA;
      if (*valor>=60000                ) seudocolor=YELLOW;
}
```

```
/*=============================================================================*/
/* DEFINECOLOR2(): Como definecolor() pero para la rutina demo1()              */
/*=============================================================================*/
void definecolor2(unsigned int *valor)
{
     if (*valor==0                          ) seudocolor=BLACK;
     if (*valor>=1      &&*valor<15         ) seudocolor=BROWN;
     if (*valor>=15     &&*valor<30         ) seudocolor=LIGHTGRAY;
     if (*valor>=30     &&*valor<45         ) seudocolor=LIGHTBLUE;
     if (*valor>=45     &&*valor<60         ) seudocolor=RED;
     if (*valor>=60     &&*valor<75         ) seudocolor=GREEN;
     if (*valor>=75     &&*valor<90         ) seudocolor=MAGENTA;
     if (*valor>=90                         ) seudocolor=YELLOW;
}


/*=============================================================================*/
/* ZOOM(): Despliega la imagen en diferentes escalas (+/-)                     */
/*=============================================================================*/
void zoom(int *factor)
{
    int i, j;
    X*=(*factor);                          /* empleo del factor de escala      */
    Y*=(*factor);                          /* con las coordenadas iniciales    */
    for(i=0; i!=(*factor); i++)
    {
       for(j=0; j!=(*factor); j++) putpixel(X+i, Y+j, seudocolor);
    }
}


/*=============================================================================*/
/* ZOOM2(): Despliega en la pantalla una imagen con diferente escala           */
/*=============================================================================*/
void zoom2(void)
{
    int factesc=1;
    int x=0, y=0, i, j, i1, j1;

    for (i=0, j=0; j!=13; i++)             /* 13 equivale a ENTER              */
    {
      string[i]=getch();                   /* se recibe la escala deseada      */
      j=string[i];
      if (string[i]==8) i-=2;              /* caso de BACK SPACE               */
      if (i<0) i=-1;
    }
    factesc=atoi(string);                  /* conversión de cadena a entero    */
    if (factesc>10) factesc=10;            /* valor máximo del factor de esc.  */
    if (factesc<1) factesc=1;              /* valor mínimo del factor de esc.  */

    setviewport(0, 12, 639, 349, ON);      /* pantalla completa                */
    clearviewport();

    for(j1=0; j1!=MAXY; j1++)
       for(i1=0; i1!=MAXX; i1++)
       {
         cont=coord[i1][j1];
         definecolor2(&cont);
```

```
            x=i1*factesc;                          /* empleo del factor de escala    */
            y=j1*factesc;                          /* con las coordenadas iniciales */
            for(i=0; i!=factesc; i++)
            {
                for(j=0; j!=factesc; j++) putpixel(x+i, y+j, seudocolor);
            }
        }
}

/*=============================================================================*/
/* DEMO1(): Simula adquirir datos y despliega una imagen en la pantalla       */
/*=============================================================================*/
void demo1(void)
{
    int factor=1;
    int orden=0;
    int i, j;

    cont=0;
    maxcont=0;
    for(j=0; j!=MAXY; j++)              /* inicialización de la matriz          */
    {
      for(i=0; i!=MAXX; i++)
      {
        coord[i][j]=0;
      }
    }
    setviewport(325,28,596,155,ON);        /* sección superior de la pantalla */
    clearviewport();
    for(;cont!=MAXM;)
    {
      orden=0;
      if (kbhit()) orden=getch();
      switch(orden)
      {
     case 68:                               /* si la orden es Detener          */
     case 100:                              /* esperará la orden Continuar     */
         while(orden!=67 && orden!=99)
           orden=getch();
         break;
     case 43:                               /* tecla +                         */
         factor++;                          /* incrementa el factor de escala  */
         if (factor>10) factor=10;
         clearviewport();                   /* limpia los pixeles "sobrantes"  */
         break;
     case 45:                               /* tecla -                         */
         factor--;                          /* disminuye el factor de escala   */
         if (factor<1) factor=1;            /* no usa menores que el inicial   */
         clearviewport();                   /* limpia los pixeles "sobrantes"  */
         break;
     case 9:
         return;                            /* TAB ordena Terminar             */
     default:
         break;
      }
```

```
      X=random(MAXX);
      Y=random(MAXY);
      cont=coord[X][Y]++;
      definecolor2(&cont);
      /*-------------------------------------------------------------------------*/
      /* capacidad de ampliar o reducir la imagen                                */
      /*-------------------------------------------------------------------------*/
      if (factor!=1) zoom(&factor);
      else putpixel(X, Y, seudocolor);
      /*-------------------------------------------------------------------------*/
      /* datos de la adquisición de imagen                                       */
      /*-------------------------------------------------------------------------*/
      if (cont>maxcont+14)   /* directamente relacionado con definecolor2()   */
      {                      /* para optimizar la velocidad de despliegue     */
         maxcont=cont;
         setviewport(0, 159, 300, 339,ON);
         setcolor(seudocolor);
         outtextxy(120,58," ▒ ");
         setviewport(325,28,596,155,ON); /* sección superior de la pantalla */
      }
   }
}

/*=========================================================================*/
/* DEMO2(): Despliega en la pantalla una imagen similar a una real         */
/*=========================================================================*/
void demo2(void)
{
   setviewport(325,28,596,155,ON);       /* sección superior de la pantalla */
   clearviewport();
   for(cont=0;cont!=65535;cont++)
   {
     X=random(272);
     Y=random(128);
     switch(cont)
     {
     case 1:
        seudocolor=BROWN;
        break;
     case 10000:
        seudocolor=RED;
        break;
     case 20000:
        seudocolor=LIGHTGRAY;
        break;
     case 30000:
        seudocolor=YELLOW;
        break;
     case 40000:
        seudocolor=LIGHTGREEN;
        break;
     case 50000:
        seudocolor=BLUE;
        break;
     case 60000:
        seudocolor=MAGENTA;
        break;
```

```
      default:
        break;
       }
      putpixel(X,Y,seudocolor);
    }
}

/*=============================================================================*/
/* REALCASE(): Recibe los datos desde los DSPs y los despliega                 */
/*=============================================================================*/
void realcase(void)
{
    int factor=1;
    int orden=0;
    int i, j;

    cont=0;
    maxcont=0;
    for(j=0; j!=MAXY; j++)                    /* inicialización de la matriz      */
    {
       for(i=0; i!=MAXX; i++)
       {
        coord[i][j]=0;
       }
    }
    setviewport(0, 21, 639, 349, ON);         /* pantalla completa                */
    clearviewport();
    for(;;)
    {
      orden=0;
      if (kbhit()) orden=getch();
      switch(orden)
      {
      case 68:                                /* si la orden es Detener           */
      case 100:                               /* esperará la orden Continuar      */
         while(orden!=67 && orden!=99)
           orden=getch();
         break;
      case 43:                                /* tecla +                          */
         factor++;                            /* incrementa el factor de escala   */
         if (factor>10) factor=10;
         clearviewport();                     /* limpia los pixeles "sobrantes"   */
         break;
      case 45:                                /* tecla -                          */
         factor--;                            /* disminuye el factor de escala    */
         if (factor<1) factor=1;              /* no usa menores que el inicial    */
         clearviewport();                     /* limpia los pixeles "sobrantes"   */
         break;
      case 9:
        return;                               /* TAB ordena Terminar              */
      default:
        break;
       }

       X=random(MAXX);
       Y=random(MAXY);
```

```
      if (cont==MAXM)                          /* si se llega al valor máximo      */
      {                                       /* vuelve a iniciar                 */
       coord[X][Y]=0;                         /* el despliegue                    */
      }
      cont=coord[X][Y]++;
      definecolor2(&cont);
      /*-------------------------------------------------------------------------*/
      /* capacidad de ampliar o reducir la imagen                                */
      /*-------------------------------------------------------------------------*/
      if (factor!=1) zoom(&factor);
      else putpixel(X, Y, seudocolor);
      /*-------------------------------------------------------------------------*/
      /* datos de la adquisición de imagen                                       */
      /*-------------------------------------------------------------------------*/
      if (cont>maxcont+14)   /* directamente relacionado con definecolor2()   */
      {                      /* para optimizar la velocidad de despliegue     */
       maxcont=cont;
       setviewport(0, 12, 150, 30, ON);
       setcolor(seudocolor);
       outtextxy(120, 0, " ■ ");
       setviewport(0, 21, 639, 349, ON);      /* pantalla completa             */
      }
    }
}


/*=========================================================================*/
/* OPEN_FILE(): Abre un archivo en una unidad de disco                     */
/*=========================================================================*/
void open_file(char *mode)
{
    clock_t i;                              /* contador de reloj            */
    char c = NULL;                          /* variable auxiliar            */
    int  j = 0;                             /* apuntador a cadena           */
    setcolor(WHITE);
    setfillstyle(SOLID_FILL,BLUE);
    setviewport(360,150,639,179,ON);        /* ventana de acceso            */
    rectangle(0,0,279,19);                  /* bordes                       */

    string[0] = NULL;                       /* inicializa cadena            */
    do
    {
      bar(1,1,278,18);                      /* rellena de azul              */
      moveto(2,2);                          /* ubica el cursor              */
      outtext("Nombre del archivo: ");      /* pide el nombre del archivo   */
       do
         {
         if (kbhit())                    /* espera por la señal del teclado */
         {
            c = bioskey(0);                 /* recibe dicha señal           */
            if (c)
            {
             /*-------------------------------------------------------------*/
             /* si se recibe BACK SPACE borra un carácter                   */
             /*-------------------------------------------------------------*/
                 if (c == '\b' && j) string[--j] = NULL;
```

```
            /*-------------------------------------------------------------*/
            /* si es un carácter imprimible, lo añade a la cadena          */
            /*-------------------------------------------------------------*/
                else if (isprint(c))
                {
                  string[j] = c;              /* añade nuevo carácter      */
                  string[++j] = NULL;         /* añade nuevo fin NULL      */
                }
                bar(155,2,224,9);             /* borra la salida anterior  */
                moveto(155,2);                /* nuevo nombre de archivo   */
                outtext(string);
                }
            }
        }
      /*-------------------------------------------------------------------*/
      /* continúa hasta recibir ENTER o hasta tener 12  caracteres         */
      /*-------------------------------------------------------------------*/
        while(c != '\r'&& j < 12);
      file = fopen(string, mode);                 /* intenta abrir archivo */
      if (file == NULL)                           /* si no puede abrirlo   */
        {
          moveto(2,10);                           /* mensaje de error      */
          outtext("Nombre incorrecto");
          i = clock();                            /* por un segundo        */
            while((clock() - i)/CLK_TCK < 1);
            j = 0;
          string[j] = NULL;                       /* prepara nueva recepción */
            c = NULL;
        }
    }
    while(file == NULL);      /* continúa hasta recibir un nombre correcto */
    clearviewport();          /* borra la ventana de acceso                */
    status();                 /* actualiza la ventana de estado            */
}

/*===========================================================================*/
/* SAVEIMG(): Guarda una imagen desplegada en la pantalla                    */
/*===========================================================================*/
void saveimg(void)
{
    int x, y;

    for(y=0; y!=MAXY; y++)
    {
       for(x=0; x!=MAXX; x++)
       {
        fwrite(&coord[x][y], sizeof(int), 1, file); /* escribe los datos */
       }
    }
    fclose(file);
}
```

```
/*=============================================================================*/
/* VIEWIMG(): Despliega en la pantalla una imagen guardada en un archivo      */
/*=============================================================================*/
void viewimg(void)
{
     int x, y;
     setviewport(325,28,596,155,ON);      /* sección superior de la pantalla */
     clearviewport();
     for(y=0; y!=MAXY; y++)
        for(x=0; x!=MAXX; x++)
        {
         fread(&coord[x][y], sizeof(int), 1, file);
         cont=coord[x][y];
         definecolor2(&cont);
         putpixel(x,y,newidcolor);
         if (feof(file)) return;
        }
     fclose(file);
}
```

BIBLIOTECA O CABECERA DEL VISUALIZADOR (GLOBAL.H)

```
/*****************************************************************************/
/*    GLOBAL.H                                                               */
/*                                                                           */
/*    VISUALIZADOR PARA EL MEPSICRON                                         */
/*                 :PROTOTIPOS, MACROS Y ESTRUCTURAS                         */
/*                                                                           */
/*****************************************************************************/
/* Prototipos de funciones                                                   */
/*===========================================================================*/
void main(void);
void init_graphics(void);
void display_error(void);
void frames(void);
void atencion(void);
void gputs(char *s);
void status(void);
void estado(char *fuente);
void command(int *haz, int *acaba);
void openhelp(void);
void definecolor(unsigned int *valor);
void definecolor2(unsigned int *valor);
void zoom(int *factor);
void demo1(void);
void demo2(void);
void realcase(void);
void open_file(char *mode);
void viewimg(void);
void saveimg(void);
/*===========================================================================*/
/* MACROS                                                                    */
/*===========================================================================*/
#define MAXX        150          /* valor máximo de X en la matriz de datos  */
#define MAXY        150          /* valor máximo de Y en la matriz de datos  */
#define MAXM        100          /* valor máximo del tamaño de la matriz     */
```

```
/*=============================================================================*/
/* VARIABLES GLOBALES                                                          */
/*=============================================================================*/
FILE *file;                     /* apuntador a archivo actual                  */
unsigned int coord[MAXX][MAXY]={0}; /* arreglo matricial para datos */
unsigned int cont, maxcont;  /* contadores de datos recibidos                  */
int X, Y;                       /* variables para la matriz de datos           */
int seudocolor;                 /* color determinado por la cuenta de datos    */
char string[80];                /* arreglo temporal para cadenas de caracteres */

int GraphDriver = EGA;          /* controlador de video EGA                    */
int GraphMode   = EGAHI;        /* controlador en modo EGA alta resolución     */
/*-----------------------------------------------------------------------------*/
/* menú principal                                                              */
/*-----------------------------------------------------------------------------*/
char *main_menu[] =
{
    " ",
    " A    Abrir",
    " B    Borrar ",
    " C    Continuar",
    " D    Detener",
    " G    Guardar",
    " I    Inicio",
    " M    Demostración 1",
    " N    Demostración 2",
    " +    Aumento de escala",
    " -    Reducción de escala",
    " ",
    " Y    Ayuda",
    " TAB  Terminar",
    " ¿opción?",
     NULL
};
```

ARCHIVO DE AYUDA (K)

```
Para cerrar este archivo de ayuda y poder continuar, presione C.
.                                                                       .
Cada comando del menu opera al presionar la tecla que le corresponde, indistin-.
tamente mayúscula o minúscula.                                          .
.                                                                       .
Abrir:      Abre un archivo que contenga una imagen grabada previamente; dicho  .
     archivo debe encontrarse en el directorio presente y debe indicarse su     .
     nombre y extensión. En caso de error, un nombre valido es "A".             .
Ayuda:      Vuelve a mostrar este archivo de ayuda.                             .
Borrar:     Borra la imagen presentada en la ventana de señal.                  .
Continuar:  Elimina la pausa producida por el comando Detener.                  .
Detener:    Hace una pausa en la rutina en curso.                               .
Guardar:    Guarda la imagen en un archivo. Es similar a Abrir.                 .
Inicio:     Recibe los datos desde los DSPs y los presenta en la pantalla.      .
Demostración 1: Ejecuta una rutina en la que se generan números aleatorios para.
     simular las coordenadas de entrada.                                        .
Demostración 2: Ejecuta una rutina en la que se generan números aleatorios para.
     simular una imagen real.                                                   .
(+) (-):    Aumenta o disminuye la escala de la imagen en rutinas de despliegue..
Terminar:   Concluye la ejecución del programa o de la rutina en curso.         .
```

APÉNDICE D

DESCRIPCIÓN DE PRUEBAS

Puede verse que la capacidad de manejo de archivos (como creación, modificación, sustitución y transferencia) es básica en la consumación de este proyecto, por ello, se pensó compilar un programa para los DSPs para que, al ejecutarlo desde el emulador de éstos, abriera un nuevo archivo binario para el almacenamiento de la matriz de datos.

De este modo, se hizo necesaria la realización de diversas pruebas tanto en el conjunto de los DSPs como en la computadora, pruebas de las que se desprendieron los siguientes resultados y conclusiones, además de la decisión de dejar al usuario la alternativa de hacerlo desde la interfaz misma, es decir, mediante el visualizador compilado en Turbo C, desde donde pueden realizarse programas que manejen archivos adecuadamente.

D.1. VARIABLES DE AMBIENTE

Son símbolos del sistema que son definidos y a los cuales se asigna una cadena

- A_DIR Directorios alternos que contienen archivos copy/include o bibliotecas de "macros" y, si no existe la variable C_DIR, bibliotecas objeto[1].
- C_DIR Directorios alternos que contienen bibliotecas objeto.
- C_DIR Directorios alternos que contienen archivos #include[2].
- D_DIR Directorios alternos que contienen comandos para el depurador (debugger).
- D_SRC Directorios adicionales que contienen archivos *.out para el depurador (debugger).
- C_OPTIONS Opciones para el compilador[3].
- D_OPTIONS Opciones para el depurador[4]

---

[1]ALT pág. 3-7, 8-10

[2]OCC pág. 2-16

[3]OCC, pág. 2-13

[4]TMS320C4x Emulator Installation Guide

D.2. COMPILACIÓN EN C PARA EL TMS320

El compilador de C acepta código fuente en C y genera código fuente en lenguaje ensamblador para el TMS320C3x/C4x; en otras palabras, la salida es un archivo objeto COFF. El comando cl30 no solo puede invocar al compilador, sino que maneja los siguentes constructores de software:

ac30	Descriptor gramatical (parser) de C
cg30	Generador de código
asm30	Ensamblador

[Nota: parece que el compilador debe estar en el directorio c:\dsptools necesariamente]

La siguiente lista define a los constructores restantes:

ar30	Archivador
clist	Utilería de "entrelistado"
lnk30	Enlazador
mk30	Utilería para construcción de bibliotecas
opt30	Optimizador

En el diagrama de flujo de desarrollo de código mostrado en el capítulo cuarto se presenta este conjunto de constructores, además de los diferentes momenteos en que es posible activar también los constructores opcionales que son ofrecidos en el conjunto de software.

(Puede consultarse además el capítulo 2 de OCC para más datos sobre ac30, cg30, opt30 y asm30, así como el capítulo 6 para mk30, en tanto que ALT es útil para ar30 (capítulo 7) y para lnk30 (capítulo 8).)

El comando cl30 activa a la terna descrita, pero el usuario puede llamar a cada uno de ellos individualmente, para producir los archivos que también son indicados por el diagrama.

Cada uno de estos constructores puede ser ejecutado con opciones específicas que determinarán así un archivo de salida con características determinadas por el usuario o programador. Estas opciones son ampliamente detalladas en el manual correspondiente[5].

A continuación son descritas las características de dos de los constructores de la terna mencionada, utilizados individualmente para elaborar de archivos específicos (no se hace necesaria la descripción de archivos en esta bitácora). No se considera el generador de código por no haberse probado el funcionamiento de sus opciones correspondientes.

---

[5]OCC pág. 2-4 y subsecuentes

Posteriormente se hace mención de diferentes opciones permitidas por el comando que agrupa al conjunto (cl30), así como de las que corresponden a otro par de constructores empleados en las pruebas presentadas en esta bitácora.

El enlazador fue también ampliamente probado y se le ha asignado una sección fuera de la que utilizada para describir la compilación.

D.2.1. EL DESCRIPTOR GRAMATICAL

La ejecución del descriptor gramatical es el primer paso en la compilación: tiene como entrada el archivo en código fuente de C, al cual analiza buscando errores de sintaxis y semánticos, generando una representación interna del programa llamado y los archivos intermedios, mediante la ejecución de las funciones de preprocesamiento.

D.2.1.1. OPCIONES DEL DESCRIPTOR GRAMATICAL

Controlan el preproceso, la revisión de sintaxis y el comportamiento de manejo de errores del compilador.

-pe trata a los errores de código E como avisos (warnings)
-pf genera prototipos para funciones
-pk activa la compatiblidad con K&R (Kernighan y Ritchie)
-pl genera un listado de preproceso en un archivo .pp
-pn elimina las directivas #line en el archivo .pp
-po solo preprocesa
-pw elimina los avisos
-p? activa la expansión trigráfica

D.2.2. OPCIONES DEL ENSAMBLADOR

Controlan el comportamiento del ensamblador:

-al crea un archivo de listado en ensamblador
-as conserva las etiquetas como símbolos
-ax crea un archivo de referencia cruzada

D.2.3. OPCIONES GENERALES DEL COMPILADOR

Las opciones por omisión para cl30 pueden ser dispuestas en la variable de ambiente C_OPTIONS. A continuación se muestran las opciones generales para el compilador, que controlan la operación en conjunto del cl30 (las restantes, mucho más específicas, pueden ser consultadas en el manual):

-c	desactiva el enlazador (niega -z)
-dnombre[=def]	predefine nombre
-g	activa la depuración simbólica
-i<dir>	añade dir a la ruta de búsqueda de #include
-k	conserva el archivo en lenguaje ensamblador (*.asm) producido por el compilador (es borrado por omisión)
-l	(L minúscula) indica que el archivo de entrada es una biblioteca objeto
-n	solo compila
-o	nombra al archivo de salida (el nombre es a.out por omisión)
-q	elimina los mensajes de avance (quiet)
-qq	elimina todos los mensajes (super quiet)
-s	entrelista los enunciados C y asm
-unombre	elimina la definición de nombre
-vxx	identifica el procesador destino TMS320Cxx
-z	activa el enlazador y produce un módulo objeto ejecutable en vez de un archivo COFF

D.2.4. EL OPTIMIZADOR

El optimizador incluido en el compilador es ejecutado entre el parser y el generador de código, transformando el código para un mejor desempeño. Puede ser llamado individualmente con el comando opt30, o bien, ser llamado desde la misma línea de comando del cl30 con la opción -o (la primera "o" indicada):

```
cl30 func1 -o -z -o funcsal.out -l rts40.lib
```

D.2.4.1. OPCIONES DEL OPTIMIZADOR

Activan y controlan los pasos de optimización

-o0	nivel 0, optimización de registros
-o1	nivel 1, optimización de registros y local
-o2 (-o)	nivel 2, optimización de registros, local y global

D.2.5. LA UTILERÍA DE ENTRELISTADO

El paquete del compilador también incluye una "utilería de entrelistado" (interlist utility), que agrega la lista de las declaraciones en C como comentarios en la salida del compilador en lenguaje ensamblador. Puede ser llamada individualmente con el comando clist, o bien, ser llamada desde la misma línea de comando del cl30 con la opción -s (lleva implícita la opción -k):

cl30 func1 -s -z -o funcsal.out -l rts40.lib

D.2.5.1. OPCIONES PARA ENTRELISTADO

-b elimina espacios en blanco y líneas inútiles (comentarios y líneas con solo { ó })
-q eliminar el banner (letrero) y la información de estado
-r elimina directivas simbólicas de depuración

D.3. PRUEBAS DE COMPILACIÓN CON EL COMANDO asm30

D.3.1. PRUEBA 1

Se intentó ensamblar un archivo en lenguaje ensamblador (*.asm) con la siguiente línea de comando desde OS/2:

asm30 -v40 suma64 -l

no siendo realizable, pues despliega el siguiente mensaje:

SYS0191: No puede ejecutarse c:\...\ASM30.EXE en una sesión de OS/2

SYS0191 indica un error en el sistema.

D.3.2. PRUEBA 2

Al ejecutar la misma línea desde DOS el comando produjo sin problemas los archivos *.lst y *.obj esperados.

D.4. PRUEBAS DE COMPILACIÓN CON EL COMANDO cl30

D.4.1. PRUEBA 1

El compilador cl30 de los DSPs (con opciones -v40 -g) no puede crear el *.obj al no poder emplear la directiva #include y, de este modo, incluir directamente las bibliotecas necesarias (stdio.h, stdarg.h y graphics.h, básicamente)

Este problema logró ser resuelto en las siguientes formas:

1 Sustituyendo las directivas #include por los archivos *.h correspondientes
2 Empleando la opción -iruta, para indicar la ruta del directorio donde se hallan los *.h

Teóricamente, también debió haberse superado esta eventualidad con la especificación de la ruta en la variable de ambiente C_DIR (ó A_DIR) en el archivo config.sys, como lo especifica el manual[6], pero no ha sido así.

En todos los casos anteriores, es decir, tanto en los que se obtuvo el resultado esperado como aquellos en los que no, las rutas especificadas son DSPTOOLS ó C4XHLL que tienen, por cierto, muchos archivos comunes. Los archivos *.h, que son llamados con la directiva #include, fueron colocados también en estos subdirectorios.

D.4.2. PRUEBA 2

Se comprobó que el comando cl30 no puede compilar los archivos *.h independientemente, i.e., archivos de cabecera (*.h) no incluidos directamente en los programas fuente de C (*.c). Esto puede entenderse al recordar que cl30 es un compilador de código fuente y, por tanto, busca ciertas características propias de la estructura de programación en C antes de realizar la compilación.

D.4.3. PRUEBA 3

Tomando en cuenta las pruebas anteriores, se compiló el archivo en C uaotro.c (archivo de prueba que copia un archivo en otro), que emplea los archivos de cabecera stdio.h y stdlib.h (encontrados en el subdirectorio c:\dsptools), con las opciones -v40 y -g (empleadas con la configuración -v40g). La línea de comando fue entonces la siguiente:

```
cl30 -v40g -ic:\dsptools uaotro
```

Esta línea de comando mostró, mediante mensajes, que ejecutó los siguientes constructores:

```
ANSI C compiler
C code generator
COFF assembler
```

creando así el archivo .obj sin mayor problema

---

[6] OCC pág. 2-16

D.4.4. PRUEBA 4

La opción -pl genera una versión modificada del archivo fuente, con extensión *.pp, que contiene toda la fuente de los archivos manejados con las directivas #include, además de las macros expandidas, sin contener comentarios. Se empleó esta opción en la línea de comando del cl30, para probar su manejo de los archivos *.h.

El comando cl30 generó así efectivamente un archivo con la extensión esperada, pero éste no contenía los archivos *.h enunciados tras las directivas en el archivo fuente. Incluso, se tomó en cuenta la consideración de encerrar los archivos llamados con directivas #include entre comillas (" "), pues hacerlo con los signos < y > causará que no se busque en el directorio en curso

D.5. EL ENLAZADOR

El enlazador combina archivos objeto en un módulo objeto ejecutable simple. Al crear este módulo, realiza la reubicación y resuelve las referencias externas. El enlazador acepta archivos reubicables COFF y bibliotecas objeto como entradas.

El enlazador es llamado con el comando lnk30 y produce como salida un archivo *.out, que es ejecutable. Puede ser llamdado también desde el compilador con la opción -z:

```
cl30 func1 -z -o funcsal.out -l rts40.lib
```

D.5.1. OPCIONES DEL ENLAZADOR

Controlan el comportamiento del enlazador

-a	genera una salida absoluta
-ar	genera una salida reubicable
-c	enlaza código en C; inicialización de ROM
-cr	enlaza código en C; inicialización de RAM (el valor por omisión es -c)
-e simb	define el "entry point"
-f val	define el valor de llenado
-h	hace estáticos los símbolos globales
-heapxxxx	dispone el tamaño del heap[7] en xxxx palabras
-idir	define la ruta de búsqueda (un directorio) de las bibliotecas
-llib	proporciona el nombre de la biblioteca

[7]Heap, montículo, montón. Es la memoria libre disponible en el momento para cargar y ejecutar programas

-m archivo	nombra al archivo mapa
-o archivo	nombra al archivo de salida
-r	genera una salida reubicable
-s	strip symbol table
-stackxxxx	dispone el tamaño de la pila (stack) en xxxx palabras
-u simb	elimina la definición de simb
-x	búsqueda exhaustiva de bibliotecas

D.5.1.1. NOTAS SOBRE LAS OPCIONES

1 opción -l
- nombra una biblioteca de archivos como entrada al enlazador
- el nombre de la biblioteca debe seguir las convenciones del sistema operativo

2 opción -x
- lee bibliotecas exhaustivamente
- resuelve las dependencias mutuas entre bibliotecas[8]

D.5.2. EL ARCHIVO DE COMANDOS DEL ENLAZADOR

Existe un archivo de comandos del enlazador, llamado c40.cmd, que contiene un conjunto de opciones para el comando del enlazador, lnk30, que el usuario puede modificar si así lo desea. El listado original de este archivo se muestra a continuación[9]:

```
/*************************************************************************/
/* C40.CMD  - v4.30  ARCHIVO DE COMANDOS PARA ENLAZAR PROGRAMAS EN C PARA C40 */
/*************************************************************************/
-c                                      /* enlaza usando convenciones de C      */
-stack 0x400                            /* pila de 1K                           */
-heap  0x400                            /* heap de 1K                           */
-lrts40.lib                             /* añade soporte en tiempo de ejecución */

/* ESPECIFICACIÓN DEL MAPA DE MEMORIA */

MEMORY
{
    ROM:    org = 0x000000   len = 0x1000       /* ROM interna              */
    RAM0:   org = 0x2FF800   len = 0x400        /* bloque 0 de RAM          */
    RAM1:   org = 0x2FFC00   len = 0x400        /* bloque 1 de RAM          */
    LOCAL:  org = 0x300000   len = 0x7D00000    /* bus local                */
    GLOBAL: org = 0x8000000  len = 0x8000000    /* bus global               */
}

/* ESPECIFICACION DE LA UBICACION DE LAS SECCIONES EN LA MEMORIA */

SECTIONS
```

---

[8]ALT pág. 8-12

[9]OCC pág. 2-47

```
{
    .text:    > LOCAL                    /* código                        */
    .cinit:   > LOCAL                    /* tablas de inicialización      */
    .const:   > LOCAL                    /* constantes                    */
    .stack:   > RAM0                     /* pila del sistema              */
    .sysmem:  > RAM1                     /* memoria dinámica (heap)       */
    .bss:     > LOCAL, block 0x10000     /* variables                     */
}
```

Con estas opciones, en lo referente a programas en C, se incluye rts40.lib, se especifica el modelo de inicialización (en ROM) y se aloja el programa en memoria, por otra parte, enlaza con boot.obj (lo hace automáticamente al definir el modelo de inicialización)[10], define las seis secciones (con las directivas MEMORY y SECTIONS)[11].

D.6. PRUEBAS CON EL ENLAZADOR Y SU ARCHIVO DE COMANDOS

D.6.1. PRUEBA 1

Se empleó el siguiente modelo para la línea de comando:

lnk30 <obj files> -o <out file> -m <map file> C40.cmd

Al ser ejecutada en OS/2, se desplegó el mensaje que se muestra:

SYS0191: No puede ejecutarse c:\...\LNK30.EXE en una sesión de OS/2

D.6.2. PRUEBA 2

La línea de comando de la prueba anteror, no obstante, fue ejecutada en DOS adecuadamente; esto con la consideración de que sólo fue activado el comando lnk30, es decir, aún no se probaban las opciones de la línea de comando.

D.6.3. PRUEBA 3

Se presentaron problemas al intentar incluir la biblioteca rts40.lib en el enlazado. Este problema fue resuelto de las siguientes formas, que indican un aspecto a tomar en cuenta al enlazar:

---

[10] OCC pág. 2-44

[11] OCC pág. 4-2

1 Debe verificarse que rts40.lib esté en el directorio en curso
2 Se puede personalizar c40.cmd para usar la variable C_DIR ó la opción -i

D.6.4. PRUEBA 4

El archivo comando c40.cmd, entre otras cosas, "llama" a la biblioteca rts40.lib, de la cual se considera que ha de tener como base prts.src. Esta biblioteca debe tener la definición de las funciones en ANSI C necesarias para construir el módulo ejecutable *.out a partir del archivo compilado *.obj, sin embargo, no aparece la definición de stdio.h, que es la que contiene propiamente las funciones para manejo de flujo de archivos.

Aunque parece haber forma de que stdio.h pueda agregarse a prts.src, se tiene aún la pregunta sobre cómo podría agregarse a rts40.lib; o bien, la forma en que deberían ser manejadas las opciones de lnk30 para que sea incluido stdio.h.

En lo que se estudió sobre el comando lnk30 no se encontró forma de salvar este problema.

D.6.5. PRUEBA 5

Inicialmente, se empleó la siguiente línea de comando, basada en el modelo descrito al principio de las pruebas:

```
lnk30 uaotro.obj -o uaotro.out -m uatro.map c40.cmd
```

Sin embargo, el archivo de comandos c40.cmd fue modificado como se indica:

contenido inicial:

```
-c
-stack 0x400
-heap 0x400
(*)
-lrts40.lib
y especificaciones de memoria y secciones
```

se le agregó, en la parte indicada con (*):

```
uaotro.obj
-o uaotro.out
-m uatro.map
-i c:\c4xhll
-i c:\dsptools
```

-x

y se le llamó uaotro.cmd, entonces solo fue necesario usar el conjunto:

lnk30 uaotro.cmd

D.7. BIBLIOTECAS CON SOPORTE EN TIEMPO DE EJECUCIÓN

Las bibliotecas con soporte en tiempo de ejecución rts40.lib y rts30.lib (que serán llamadas "bibliotecas" simplemente, salvo indicación de lo contrario), parte del conjunto de programas para los DSPs, contienen un número de funciones en lenguaje ensamblador que proveen rutinas aritméticas para operadores matemáticos de C que el conjunto de instrucciones del TMS320C3x/C4x no proporciona, como la división[12].

El código fuente de estas bibliotecas es el archivo rts.src, el cual, al ser construido con las opciones -v40, -o2, -x y -ml produce la biblioteca rts40.lib para el TMS320C4x. El caso para el C3x es similar y solo cambia la opción -v40 por -v30 simplemente[13].

Puede observarse que la importancia de estas bibliotecas reside en que todos los programas en C deben ser enlazados con una de ellas. Las funciones de esta biblioteca de archivos son usadas por el compilador para administrar el ambiente en C adecuadamente[14].

D.7.1. UTILERÍA PARA CONSTRUCCIÓN DE BIBLIOTECAS

Esta utilería es invocada por el comando mk30 y permite emplear, además de las opciones propias del compilador y el ensamblador, las siguientes, descritas brevemente:

--v imprime en la pantalla la información progresiva sobre la ejecución de la utilería
--u evita el uso de los archivos de cabecera (headers) contenidos en el archivo fuente.
--h obliga a emplear los archivos de cabecera contenidos en el archivo fuente.
--c los archivos fuente en C son extraídos de la biblioteca al terminar de ejecutar mk30.
--k sobreescribe archivos.

El comando mk30 permite la obtención de una biblioteca con soporte en tiempo de ejecución a partir del archivo fuente rts.src, con las opciones que el usuario seleccione, sin alterar el código mismo de rts.src propiamente.

---

[12] OCC pág. 4-39

[13] OCC Reference card

[14] OCC pág. 2-41, 2-43

A continuación se muestran los archivos que constituyen esta biblioteca[15]:

assert.h
ctype.h
errno.h
float.h
limits.h
math.h
stdarg.h
setjump.h
stddef.h
stdlib.h
string.h
time.h

Debido a que el código de rts.src no contiene la biblioteca de C stdio.h como ha podido verse, los archivos que resultan al compilar rts.src no permiten el manejo de archivos por parte los programas escritos para los DSPs.

Existe efectivamente una cierta capacidad de manejar archivos, como se verá posteriormente, pero no es lo suficientemente adecuada para incluirla en la interfaz final; esto significa que no tiene la flexibilidad que se pudiera esperar como en el caso de programas en C en los que se incluyan las bibliotecas *.h mencionadas.

En virtud de lo anterior, se decidió hacer pruebas con la utilería invocada por el comando mk30, buscando tener programas, escritos para los DSPs, capaces de realizar dicho manejo. A continuación se describen las pruebas efectuadas y los resultados obtenidos.

D.8. PRUEBAS CON LA UTILERÍA

D.8.1. PRUEBA 1

Se agregó directamente (sin que los manuales aportaran un método para hacerlo) a rts.src las bibliotecas de C stdio.h y graphics.h y a este nuevo archivo se le llamó rtsjs.src. Se ejecutó el comando desde DOS con las opciones de descripción de la ejecución (--v), omisión de archivos de cabecera (--u), nivel de optimización 0 (-o), empleando como archivo fuente rtsjs.src y dando como nombre de archivo en la salida rts40a.lib (-l):

---

[15]La descripción de estos archivos puede verse en OCC pág. 5-4

```
mk30 --v --u -o rtsjs.src -l rts40a.lib
```

Al ejecutar el comando en la forma descrita, solo aparece el siguiente mensaje y se detiene toda posterior ejecución:

```
>>File "rtsjs.src" not an archive
```

Se probó también llamar a esta utilería desde OS/2 con la misma línea de opciones; el sistema "llamó" a DOS y se obtuvo idéntico resultado.

D.8.2. PRUEBA 2

Se ejecutó el comando desde DOS con las opciones de construcción para el procesador C40 (-v40), nivel de optimización 2 (-o2), empleando la opción de "inlinning" (-x), deshabilitando la instrucción RPTS (-mi), empleando como archivo fuente rts2js.src, que es simplemente una copia de rts.src y dando como nombre de archivo en la salida rts402js.lib (-l):

```
mk30 -v40 -o2 -x -mi rts2js.src -l rts402js.lib
```

se obtiene, al igual que en el caso anterior, el mismo mensaje:

```
>>File "rts2js.src" not an archive
```

D.8.3. PRUEBA 3

Usando el nombre original, rts.src, con el comando mk30 y con las opciones mencionadas en la prueba pasada, comienza a ser ejecutado apropiadamente, indicando los siguientes tres mensajes:

```
Building archive "rts.src"--> "rts402js.lib"
```

(el mensaje que se muestra a continuación se abrevió con *.h, pues en realidad apareció con el nombre de cada uno de los archivos de cabecera, es decir, de extensión "h")

```
>>File "*.h" already exists

>>Directory contains files which would be overwritten if the archive is
built.  Move/delete the files, or use the --k option which will overwrite
them
```

("El directorio contiene archivos que serían sobreescritos si el archivo final es construido. Mueva o borre los archivos o utilice la opción --k que sobreescribirá en ellos")

D.8.4. PRUEBA 4

Empleando la opción --k (sobreescribir) es presentada una serie de nombres de funciones de C que parecen estar siendo compiladas: boot, divf, assert, atan, atof, atoi, etc., pero marca varias veces el siguiente mensaje (entre la serie de dichos nombres):

>>> ERROR - OUT TO MEMORY - OPTIMIZER ABORTED

>>> optimizer aborted abnormally

y se genera, en efecto, un archivo en la salida, pero de una dimensión menor que la esperada: la intención fue generar una biblioteca ya existente (rts40.lib) a partir de los módulos que lo conformaron de acuerdo con la información de los manuales, pero el resultado fue el indicado.

En números:

rts402js.lib = 67, 626 bytes
rts40.lib = 69, 020 bytes

D.8.5. PRUEBA 5

Empleando el comando mk30 con las opciones indicadas a continuación y ya descritas anteriormente; tomando como archivo fuente el modificado conteniendo a stdio.h y los demás, rtsjs.src, renombrado a rts.src:

rts.src ---> rts.tmp
rtsjs.src ---> rts.src

mk30 --k --u --v -v40 -o2 -x -mi rts.src -l masdec.lib

(el nombre "masdec.lib" significa "más cosas de C")

se obtiene el siguiente mensaje (ya mostrado)

>> File "rts.src" not an archive

D.8.6. PRUEBA 6

Bajo las mismas especificaciones de la prueba anterior, simplemente eliminando la opción --u, se obtiene el mismo mensaje, al igual que al eliminar la opción --v .

D.9. OTRAS OPCIONES DE PROGRAMACIÓN

Buscando las características óptimas para la realización de los programas para el manejo de la señal en el conjunto de los DSPs y para el visualizador que sería la interfaz con el usuario final, se pensó en emplear, para este último caso, la denominada programación orientada a objetos (OOP, por sus iniciales en inglés), que presenta ciertas ventajas[16] sobre el lenguaje empleado finalmente, pero no se abundó demasiado en este aspecto por la experiencia que se expone a continuación:

Se hizo ciertamente un pequeño programa de prueba en el lenguaje de programación Delphi, que es el que puede conseguirse en el Instituto de Astronomía, pero al intentar ejecutarlo en el ambiente de OS/2, se presentaron conflictos de memoria que no permitieron su funcionamiento.

Puede verse un problema de portabilidad en el hecho de que Delphi es un lenguaje de programación destinado a trabajar en los ambientes Windows de Microsoft, sin mostrar capacidad o compatibilidad con el sistema operativo OS/2 de IBM. Fue esta misma situación lo que decidió no contemplar al lenguaje Visual Basic como opción viable[17].

Existe la posibilidad de implementar en el futuro el visualizador con este tipo de programación, que puede ser más accesible para el usuario; para ello, es claro el requerimiento de un lenguaje de programación que pueda ser ejecutado en el ambiente de OS/2.

---

[16]Puede consultarse Object-Oriented Programming: An Introduction, de Greg Voss

[17]Se detalla esta situación en Personal Computing México, mayo 1995, pág.40, 45, 46

BIBLIOGRAFÍA

1 BOHIGAS J. MEPSI para principiantes. Manual de usuarios del sistema Mepsicrón - Tlachialoni. Reporte técnico No. 19. Instituto de Astronomía, UNAM,1985

2 ENRÍQUEZ CALDERA, Rogerio (sic) et al. Tlachialolni I. Sistema para la adquisición, almacenamiento y despliegue de imágenes bidimensionales. Reporte técnico No. 20. Instituto de Astronomía, UNAM, 1985

3 FREEDMAN, Alan. Diccionario de Computación. (Trad. de Morales Jareño, Isabel.) Quinta edición. España: McGraw Hill,1993. 934 pp.

4 HECHT, Eugene; ZAJAC, Alfred. Óptica. México: Addison-Wesley Iberoamericana, 1990. 586 pp

5 HERNÁNDEZ ALVA, Adrián. Tesis: "Diseño de la unidad aritmética de la electrónica de un detector tipo Mepsicrón"
Asesor: M. C. José Luis Pérez Silva
Facultad de Ingeniería, UNAM, 1994

6 JAMSA, Kris. DOS Manual de Referencia. (Trad. de García-Bermejo G., Rafael.) México: Osborne/Mc Graw Hill. 1991. 1046 pp.

7 KARTTUNEN, Hannu et al. Fundamental Astronomy. Second Enlarged Edition. Alemania: Springer-Verlag, 1994. 511 pp.

8 LÉNA, Pierre. Observational Astrophysics. Alemania: Springer-Verlag, 1988. 328 pp.

9 LYNXX/LYNXX PLUS CCD Digital Imaging System User's Manual. Spectra Source Instruments.

10 MATA ESTARELLAS, Antonio. Turbo C/C++, Iniciación y Programación Avanzada. 4a. edición. España: Paraninfo, 1993.

11 McLAN, Ian S. Electronic and Computer-Aided Astronomy, from eyes to electronic sensors. New York: John Wiley & Sons, 1989.

12 MILLMAN, Jacob; GRABEL, Arvin. Microelectronics. New York: McGraw Hill International Editions, Electronic Engineering Series, 1987.

13 OS/2 2.1 Utilización del Sistema Operativo. IBM. Dinamarca, 1993.

14 PAPAMICHALIS. DSP Applications with the TMS320 family, vol. 3.

15 Parallel Debug Manager, Addendum to the TMS320C4x and TMS320C5x C Source Debugger User's Guides. Texas Instruments. USA, 1995. 58 pp.

16 PLANTZ, Alan C. C, Manual de bolsillo. (Trad. de Flores Suárez, Pedro.) México: Addison-Wesley Iberoamericana, 1989. 191 pp.

17 RUIZ, E. Martín; IRIARTE, Arturo. Preamplificadores para el Sistema Mepsicrón. Reporte técnico No. 51. Instituto de Astronomía, UNAM, 1988

18 SCHILDT, Herbert. C, Manual de Referencia. (Trad. de Hdez. Yáñez, Luis.) España: Osborne / McGraw-Hill, 1990. 695 pp.

19 SEARS, Francis W. Fundamentos de Física III, Óptica. 3a. edición. Madrid: Aguilar, 1958. 385 pp.

20 SMITH, Robert C. Observational Astrophysics. Great Britain: Ed. Cambridge University Press, 1995.

21 TMS320C4x Emulator Installation Guide, Texas Instruments, USA. 1993. 58 pp.

22 TMS320C4x Parallel Processing Development System, Technical Reference, Texas Instruments, USA. 1993. 138 pp.

23 TMS320C4x User's Guide, Texas Instruments, USA. 1993. 944 pp.

24 TMS320C4x C Source Debugger User's Guide, Texas Instruments, USA. 1992. 366 pp.

25 TMS320 Floating-Point DSP Optimizing C Compiler User's Guide, Texas Instruments. USA. 1991. 242 pp.

26 TMS320 Floating-Point DSP Assembly Language Tools User's Guide, Texas Instruments, USA. 1991. 332 pp.

27 TOCCI, Ronald J. Sistemas Digitales. (Trad. de Vega F., Juan Carlos) 3a. Edición. México: Prentice Hall, 1987. 624 pp.

28 VOSS, Greg. Object-Oriented Programming: An Introduction. USA: Osborne MacGraw-Hill, 1991. 584 pp.

29 Diccionarios Rioduero. Física
Ediplesa, 1976. 253 pp.

Otras fuentes de información:

30 Artículo:

"Las Nuevas Herramientas Visuales"
Personal Computing México
Mayo 1995, pág. 36 a 46
Ed. Sayrols

31 Texas Instruments Hot Line:

713 274 2320
214 644 5580

32 Conferencia:

Fernando Garfias
Pablo Sotelo
Sistema de control para óptica adaptativa
Instituto de Astronomía, UNAM
11 de abril de 1996

33 Conferencia:

Silvayn Martini
Microcontroladores y procesadores digitales de señales de Texas Instruments
Facultad de Ingeniería, UNAM
30 de mayo de 1996

34 Internet:

Understanding Memory Types
http://www.northnet.com/training/sld012.htm
30 de septiembre de 1996

35 Internet:

Cyberware... el BBS!
http://www.cware.com.mx/
20 de octubre de 1996
Tel: 645 9000
SysOp: Miguel Priego
Gte. Soporte Técnico: Jinkichi Oshino

36 Ayudas en línea:

MS-DOS
IBM OS/2

Otra bibliografía relacionada (no empleada directamente):

37 FIRMANI C.; GUTIÉRREZ L., et. al.
Espectrofotometría de Alta Dispersión con el contador de fotones Mepsicrón
Instituto de Astronomía, UNAM
Segundo simposio de instrumentación
Centro de Instrumentos, UNAM
1982

38 FIRMANI C.
El Mepsicrón y el avance en la detección de imágenes con alta resolución
Revista Mexicana de Física 31 No. 3 (503-519)
1985

39 PÉREZ S., FUENTES G., NOGUEIRA J.
Electrónica para detector Mepsicrón
Memorias del VIII Congreso Nacional de Instrumentación
1993