34

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE INGENIERÍA

DISEÑO DE UNA INTERFAZ ELECTRÓNICA COMO PARTE DEL CONTROL DE UN TELESCOPIO CON MONTURA ECUATORIAL

TESIS

QUE PARA OBTENER EL TÍTULO DE

INGENIERO ELÉCTRICO ELECTRÓNICO

PRESENTA:

PALENCIA CASTRO FRANCISCO

DIRECTOR DE TESIS:

M.I. LAURO SANTIAGO CRUZ

MÉXICO, D.F. SEPTIEMBRE 2002

A Dios,
que es Amor.

A Papá y a Mamá,
regios faros de su marinero.

A Magalena y a Ramón,
dones y compañeros fraternales

A Magali,
la epopéyica Kriemhilde

Gracias por todo.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El origen de este trabajo surge directamente del Programa de Sistema de Interfaz y Control de Telescopio, llevado a cabo en el Laboratorio de Electrónica del Instituto de Astronomía de la UNAM. Este programa nació cuando se vió la necesidad de cambiar la computadora del sistema de control electrónico del telescopio de 1 metro del Observatorio Astronómico Nacional en Tonantzintla, Puebla. Este sistema utilizaba un procesador 80286 y, cuando se intentó cambiar el CPU (*Central Processing Unit*, Unidad de Proceso Central) del sistema por un 80486, el sistema presentó fallas de funcionamiento en su control manual. Fue entonces cuando se vió la necesidad de construir una nueva interfaz, que pudiera funcionar en una microcomputadora tipo PC de mayor velocidad de operación. Como parte del proceso de desarrollo del *hardware* y del *software* de este programa, se desarrolló una tarjeta construida mediante la técnica de *wire-wrap* que se encuentra actualmente funcionando con el telescopio.

Esta tarjeta de interfaz puede encontrar uso en otros Observatorios o en otra clase de aplicaciones que requieran muestreo de trenes de pulsos o señales digitales de entrada y salida, además de que también es posible adaptarle un ADC (*Analog Digital Converter*, Convertidor Analógico Digital) para realizar la interfaz de la microcomputadora con otros dispositivos. Sin embargo, debido al considerable número de circuitos integrados que componen a esta tarjeta y, debido a la dificultad que representa reproducirla, es deseable integrar la electrónica de esta tarjeta al máximo para, posteriormente, poder lograr su implementación de una manera rápida y fácil, de acuerdo con la demanda y con las diversas aplicaciones que se presenten.

Así, el objetivo principal del presente trabajo de tesis es diseñar una interfaz que integre al máximo el circuito de la tarjeta en un CPLD (*Complex Programmable Logic Device*, Dispositivo Lógico Programable Complejo), para conseguir la máxima reducción del número de componentes y facilitar la implementación de otras tarjetas de interfaz.

Básicamente, el CPLD programado debe responder a las siguientes especificaciones:

- Presentar cuatro entradas que permitan procesar las señales que proceden del controlador manual del telescopio, o *handset*, para cada uno de los cuatro movimientos posibles de la montura del telescopio, y con el cual, por medio de sus botones de dirección, se dirige el telescopio. Las señales provenientes del controlador manual del telescopio son un tren de pulsos, independientes para cada eje de movimiento, que el circuito en la tarjeta cuenta para que, posteriormente, el programa principal, haciendo lecturas a los puertos, indique los movimientos al controlador de motores.
- Indicador de inicio de cada movimiento, su velocidad y su finalización
- Una entrada por la que llega una frecuencia de 90 Hertz siderales. Esta frecuencia va de acuerdo a la frecuencia de guiado sideral necesaria para operar el telescopio.
- El control del envío de datos estará habilitado por un *flipflop* en sincronía con el controlador de motores, denominado *Galil*, que se encuentra alojado en la misma computadora.
- De igual manera, el CPLD debe interactuar con un convertidor Analógico/Digital contenido en la misma tarjeta, para interpretar las señales procedentes de distintos sensores, y enviarlos al programa de la consola de control.

La programación del CPLD se llevará a cabo con la ayuda del *software* Max+Plus II de Altera. Max+Plus II es en este momento la tecnología de punta dentro de la programación de los CPLDs. Su *software* compila los diseños más rápido que cualquier otra herramienta de desarrollo para PLDs (*Programmable Logic Devices*, Dispositivos Lógicos Programables), permitiendo implementar cambios velozmente en su editor de

texto o editores esquemáticos. Igualmente, debido a sus sistemas de simulación y analizador de tiempos de propagación, permite desarrollar distintas funciones de la tarjeta de interfaz por separado, probando de inmediato sus resultados. Además, la inclusión de macros preestablecidos permite una interacción más productiva.

Los CPLDs están basados en una arquitectura de interconexión de alto desempeño, compuesta por LABs (*Logic Array Blocks*, Bloques de Arreglos Lógicos). Su arquitectura básica es parecida a un enorme PAL (*Programmable Array Logic*, Arreglo Lógico Programable), con la capacidad añadida de poder incrementar la cantidad de términos AND para cualquier término OR fijado en su estructura (como en los PLAs). Así, los múltiples LABs pueden ser interconectados mediante un arreglo de interconexión programable y un *bus* global formado por terminales dedicadas, directamente desde un *software* especializado. La principal ventaja para este trabajo que se obtiene utilizando LABs es que sus macroceldas pueden ser configuradas de manera individual, entre una salida combinacional o secuencial. Esto permite implementar casi cualquier diseño dentro de estos dispositivos, teniendo como característica adicional, la capacidad de ser regrabables para cualquier modificación que fuera necesaria en un futuro.

El desarrollo del presente trabajo está estructurado en cuatro capítulos que se resumen a continuación:

En el primer capítulo se describen los aspectos principales a tener en cuenta con respecto al funcionamiento de los telescopios y su automatización, con la finalidad de establecer un panorama con el cual se comprenda mejor la funcionalidad de la interfaz, y se plantea la necesidad de ésta de acuerdo a los requerimientos del sistema de control del telescopio de 1m. del Observatorio Astronómico Nacional en Tonantzintla y su demanda por otros observatorios. Por último, se explica la importancia del proyecto con base en los objetivos y tendencias de la tecnología digital en la actualidad.

En el capítulo segundo se describen los elementos que conformarán la interfaz utilizada en este trabajo, en conjunto con los microprocesadores i86 de INTEL, con especial enfoque en el Pentium. Se analizan las características del *bus* ISA, que es el empleado por la tarjeta para comunicarse con la computadora y su protocolo. Igualmente se analizan las principales características de las transmisiones de datos, las interrupciones utilizadas, y la decodificación de puertos. Además, se resumen las funciones de los elementos generales de los CPLD, particularizando en la programación con el *software* Max+Plus II de Altera, en los dispositivos MAX7000 y en sus características distintivas para su utilización en este proyecto.

Dentro del capítulo tercero se describe la lógica del diseño de la interfaz, explicándose la lógica digital empleada para llevarla a cabo, organizada en módulos específicos.

En el capítulo cuarto se diseña la interfaz mediante la organización e implementación de las distintas subrutinas de programación del CPLD, su simulación y, por último, su integración posterior en un único programa. Asimismo, a partir del tratamiento de las especificaciones operativas de la tarjeta, se expone el diseño del circuito esquemático y el circuito impreso de la interfaz electrónica embebida.

Finalmente, se evalúa la operación de la tarjeta de interfaz en el capítulo cinco, y se presentan los resultados y conclusiones del presente trabajo en el capítulo sexto; además de la bibliografía consultada y la información complementaria en los apéndices A, B, C, D y E. Igualmente se incluye un glosario de términos astronómicos, cuyo contenido está orientado a facilitar la comprensión de algunos conceptos básicos que serán empleados en este trabajo.

I

AUTOMATIZACIÓN DE LOS TELESCOPIOS

La comprensión de la evolución de los telescopios, sus principales características, y su funcionamiento, permite vislumbrar el panorama en el que se desenvuelve la interfaz electrónica presentada en este trabajo y su finalidad. Igualmente, con este antecedente, se puede proceder al planteamiento de la necesidad de esta interfaz, de acuerdo a los requerimientos específicos del sistema de control del telescopio de 1m del Observatorio Astronómico Nacional en Tonantzintla y su posible demanda por otros observatorios, siguiendo las últimas tendencias de la tecnología digital.

1.1. INTRODUCCIÓN A LOS TELESCOPIOS

La tecnología actual que se utiliza para la operación de los telescopios es producto de una larga lista de mejoras e innovaciones a lo largo de la historia. La evolución del telescopio y sus periféricos obedece a la búsqueda de instrumentos más precisos y/o con mayor capacidad de recolección de luz. Asimismo, la razón de su automatización se debe a la búsqueda de que estos instrumentos no requieran de una mayor atención en su operación, para que así los astrónomos puedan dedicar toda su atención a sus observaciones.

La característica clave de un telescopio es su abertura. La abertura es el diámetro del espejo principal (o la lente principal en el caso de los telescopios pequeños) y determina la cantidad de luz que puede recoger y enfocar en una imagen. A mayor apertura mejor resultado, con lo que se explica la tendencia de construir enormes telescopios en el milenio pasado (se llegaron a construir aparatos tan enormes que, en el año de 1839, el hijo del eminente astrónomo William Hershel desmontó el enorme telescopio de su padre y celebró una fiesta familiar en el interior del tubo). Igualmente, gracias a le evolución en los sistemas ópticos de los telescopios, en particular, gracias a la implementación del telescopio de tipo reflector, con base en los efectos ópticos de los espejos, se ha permitido ampliar notablemente la abertura y la capacidad de los telescopios.

Simultáneamente, desde el primer momento en que se iniciaron las observaciones astronómicas con el telescopio, se vió la necesidad de construirles monturas especializadas para proporcionar estabilidad en sus imágenes y facilidad de su manejo. Conforme los telescopios aumentaban de abertura y, por lo tanto, de volumen, los sistemas con los que los telescopios eran guiados se fueron sofisticando para facilitar las

observaciones. El refinamiento de estas monturas y sus sistemas de posicionamiento y guiado, ha permitido una mayor automatización de los telescopios que soportan.

Podría ser redundante señalar que, al referirse a un telescopio, no se suela tocar el tema de los aumentos. Esta razón se debe a que es falsa la creencia de que un telescopio es mejor cuantos más aumentos proporciona. Los aumentos se consiguen cambiando simplemente el ocular, o dispositivo empleado para las observaciones, y éstos pueden utilizarse en cualquier telescopio si se adaptan los diámetros de los tubos. Por lo tanto, un telescopio pequeño podría, en teoría, hacer que proporcionara tantos aumentos como el de un gran reflector, aunque no se podría ver nada con él, ya que las potencias tienen un límite máximo en cada telescopio. Este límite máximo está en función de la abertura del telescopio que, a su vez, está en función del diámetro del espejo o lente principal del telescopio.

Así, la búsqueda de instrumentos con mayor abertura y mayor definición, ha permanecido una constante desde la creación del primer telescopio.

1.1.1. ANTECEDENTES HISTÓRICOS

Como se mencionó anteriormente, el elemento básico que constituye a un telescopio es su espejo o lente principal y, mientras mayor sea su diámetro, se podrá colectar más luz con ellos. Aparentemente, la invención de la lente debe atribuírsele a los chinos. Ya en 1249, el inglés Roger Bacon mencionaba el uso de pequeñas lentes en China para mejorar la visión de personas ancianas que tenían dificultades para ver de cerca (presbicia). La primera utilización de la lente para llevar más lejos el rayo de luz y corregir dificultades de visión lejana se debe al alemán Nicolás de Cursa en 1451, cuando propuso el empleo de lentes cóncavos, más delgados en el centro que en los bordes.

Sobre el origen del telescopio (cuyo nombre proviene del griego *tele scopeo*: observación a distancia) hay polémica. Aunque es seguro que el primero se construyó en Holanda, el deseo de sus inventores de beneficiarse ellos solos de su creación, impidió conocer el detalle de su origen. De cualquier forma, la autoría del invento es disputada simultáneamente por Johann Lipperhey, óptico de Middelburg y el anteojero Zacharias Jansen. Se sabe que ambos solicitaron una patente casi al mismo tiempo para el mismo invento: "un aparato que podía ver de lejos".

Aparentemente Lipperhey fue el primero en lograr el primer telescopio, en 1608, luego que un aprendiz del óptico, al jugar en sus tiempos libres con algunos lentes encontró que, si sostenía un lente sobre otro, veía el lejano campanario de una iglesia como si estuviera más cerca. La anécdota cuenta que el aprendiz, asombrado, se lo contó a su patrón, quien captó de inmediato la importancia del descubrimiento montando las lentes en un tubo, implementando así el telescopio refractor.

En esa época, los Estados Generales de los Países Bajos se encontraban en rebelión contra España, y Lipperhey, dándose cuenta de que el telescopio sería una importante arma de guerra, se lo presentó al jefe militar Mauricio de Nassau, quien en vano trató de mantener en secreto las características del invento. El aparato era demasiado sencillo y cualquiera podía construirlo.

Así, unos pocos meses más tarde, en 1609, el físico y astrónomo italiano Galileo Galilei oyó hablar del extraordinario invento holandés. Como no sabía nada de su construcción, Galileo se puso a experimentar y tuvo la satisfacción de construir un primer telescopio de pequeña abertura que aumentaba tres veces el tamaño de los objetos. Inmediatamente construyó mejores telescopios y fue la primera persona en utilizarlo para la astronomía, al enfocarlo al cielo y descubrir los cráteres de la luna, las fases de Venus, las manchas del sol y los satélites de Júpiter (también encontró que Saturno tenía unas

especies de "orejas" que luego serían identificados como los anillos que orbitan el planeta). El mayor de los telescopios de Galileo aumentaba en treinta veces la imagen, pero era muy rudimentario y se apoyaba en un sencillo tripié de madera.

Poco a poco el telescopio fue evolucionando y a su desarrollo contribuyeron científicos como Nicolás Zucchius, Johannes Kepler, Christian Huygens, Giovanni Domenico Cassini, Johannes Hevelius, William Hershell, Alvan Clark, entre otros, pero en especial gracias al escocés James Gregory que, en 1661, inventó el telescopio reflector (aunque fuera Issac Newton el que lo construyera con éxito la primera vez en 1668). Todos estos científicos aportaron significativas innovaciones o refinamientos en los telescopios y su óptica, tales que se han convertido en los fundamentos de los instrumentos utilizados hoy en día en los modernos observatorios.

1.1.2 TELESCOPIOS Y SU ÓPTICA

Existen varias clases de telescopios. Estas diferentes clases obedecen, tanto a la evolución que ha presentado el telescopio a lo largo de la historia, como a la especialización a los que los mismos instrumentos se han dedicado para distintas observaciones. Sin embargo, todos los telescopios pueden catalogarse dentro de dos grandes grupos: los del tipo refractor y los del tipo reflector. En ambos tipos de telescopio, la imagen final que se obtiene con ellos está invertida pero, mientras el telescopio se utilice en observaciones astronómicas, esta característica no tiene importancia, sobre todo si se considera que el trabajo de éstos es siempre de tipo fotográfico.

- *EL TELESCOPIO REFRACTOR*

Los primeros telescopios eran del tipo refractor. Un telescopio refractor era aquel telescopio de visión directa, los típicos catalejos, con una lente-objetivo, situada en la parte superior de un largo tubo (fig. 1.1). El lente-objetivo concentraba la luz del astro en un punto de la parte inferior del tubo donde se sitúa el ocular, elemento que proporcionaba su poder de magnificación.

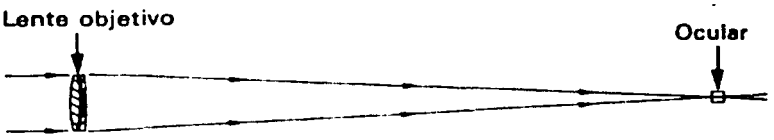


Figura 1.1. Telescopio refractor.

Un inconveniente con este tipo de telescopio era que el tallado óptico de los objetivos refractores ofrecía bastantes dificultades cuando se trataba de tamaños grandes. Además, las dimensiones del tubo ocasionaban serias dificultades de instalación. El refractor más grande del mundo, por ejemplo, tiene una abertura de 1.02 metros y obliga a emplear un tubo de más de 10 metros de largo (este telescopio se encuentra en el Observatorio Yerkes de la Universidad de Chicago y fue construido en 1892). Otro problema era el peso de la lente. Debido a los grandes pesos que tenían que soportar, las monturas de estos telescopios debían estar diseñados para soportar enormes esfuerzos, lo que no siempre se traducía en una mejor capacidad de guiado, además de que las lentes sólo podían ser sujetadas por su periferia, propiciando deformaciones de ésta.

Igualmente, los telescopios refractores perdían mucha luz debido a que cada lente cuenta con dos superficies que reflejan inintencionadamente los haces de luz, además de

las pérdidas que se presentan en el medio, o cristal, de la lente, ya que una lente debe ser completamente transparente, estar libre de burbujas internas, etc.

Adicionalmente, los telescopios reflectores presentaban aberración cromática (ver fig. 1.2). Debido a que la trayectoria óptica de un rayo de cierta longitud de onda es ligeramente distinta al camino que sigue un rayo con otra longitud de onda, los distintos haces de luz con distintas longitudes de onda provenientes de un objeto celeste no podían ser enfocadas en un mismo punto en este tipo de telescopio. Esto es debido al fenómeno de dispersión causado por el índice de refracción del material de la lente, que depende a su vez de la longitud de onda de la luz incidente. Esta aberración se refiere básicamente al deterioro en la calidad de la imagen y no en su forma geométrica.

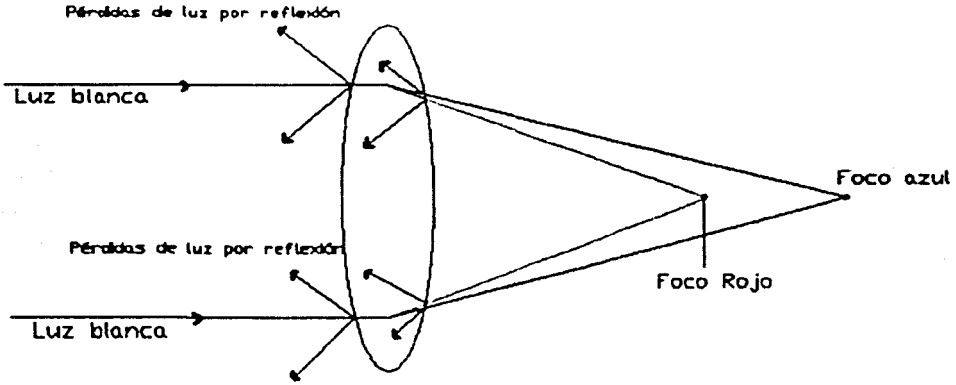


Figura 1.2. Pérdidas de luz y aberración cromática en un telescopio refractor.

De las razones expuestas anteriormente, se deduce que al telescopio refractor, a pesar de ser el paradigma de telescopio al que se recurre popularmente, se le tenga ahora

solamente un interés histórico y/o pedagógico, por lo que, evidentemente, no es el tipo de telescopio utilizado en los modernos observatorios.

- *EL TELESCOPIO REFLECTOR*

El telescopio reflector está basado en la utilización de un espejo como objetivo, cuya cara reflectante tiene una suave curvatura esférica o parabólica que concentra toda la luz en un punto donde se coloca el ocular. La ventaja que ofrece esta disposición, es que se tiene una mayor recolección de luz, ya que ésta se encuentra con una sola superficie que la refleja casi en su totalidad, con pérdidas imperceptibles, además de que no existe un medio que absorba la energía procedente de los astros.

Existen tres variaciones fundamentales del telescopio reflector, y se diferencian entre sí por la forma en que el haz de luz es llevado a la parte externa del tubo. Estas variaciones son: el tipo Newtoniano, el tipo Gregoriano y el tipo Cassegrainiano.

En el telescopio reflector tipo Newtoniano se utiliza un espejo secundario diagonal, o un prisma, para llevar el rayo luminoso a la parte externa del tubo (ver fig. 1.3), donde éste se puede fotografiar, analizar espectralmente o procesar fotoeléctricamente con una cámara CCD (*Charged Coupled Devices*).

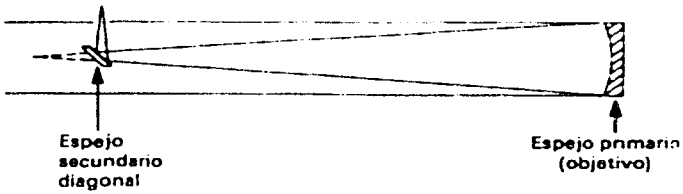


Figura 1.3. Telescopio reflector tipo Newtoniano.

En el telescopio reflector tipo Gregoriano (ver fig. 1.4), se utiliza un espejo secundario elipsoidal cóncavo que regresa la imagen a través de un agujero en el espejo primario. La característica distintiva de este tipo de telescopio es que las imágenes que proporciona no son invertidas.

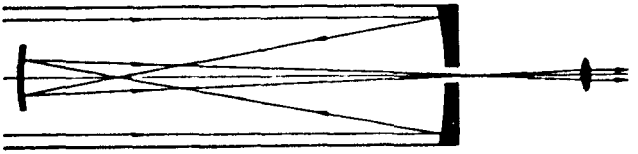


Figura 1.4. Telescopio reflector tipo Gregoriano.

En el telescopio reflector tipo Cassegrainiano, llamado así debido a su inventor, Guillaume Cassegrain, escultor al servicio de Luis XIV, en Francia, en 1672, (ver fig. 1.5) se utiliza un espejo secundario hiperboloidal convexo para aumentar la distancia focal efectiva. Así, esta variación permite que el espejo primario actúe como si estuviera a una distancia focal más grande pero con la misma abertura, lo que permite telescopios de menor volumen (el punto en el cual se reúnen los rayos procedentes de un punto en el infinito, para formar una imagen, se llama foco, ya que los rayos se enfocan en él y su distancia al espejo se llama distancia focal (f); esta distancia focal se relaciona directamente con el diámetro del espejo principal e, indirectamente, con la capacidad de aumento del telescopio).

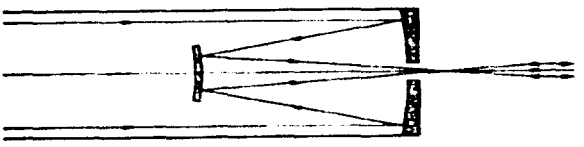


Figura 1.5. Telescopio reflector tipo Cassegrain.

Adicionalmente existe una variación del sistema reflector denominado sistema Ritchey-Chrétien. En este tipo de telescopio el espejo secundario tiene una forma tal que elimina la aberración esférica, mientras que la forma del espejo primario reduce la aberración en cabellera o coma. Esto propicia que la forma del espejo primario tenga una forma menos acusada en comparación a los paraboloides para algunas longitudes de onda. El telescopio Ritchey-Chrétien se utiliza para conseguir fotografías de amplia apertura, pero, desgraciadamente, el sistema Ritchey-Chrétien sencillo sufre de astigmatismo. Esto puede solucionarse con dispositivos correctores, como fue el caso del Telescopio Espacial *Hubble*.

Debido a que en los telescopios reflectores no existe refracción, al no existir un lente primario, estos telescopios no sufren aberración cromática.

Adicionalmente, los telescopios reflectores tienen la facilidad de que, como el ocular se coloca en la parte superior del tubo, se permiten montajes mucho más bajos que los telescopios refractores, lo que proporciona mayor estabilidad y menores esfuerzos mecánicos en la estructura del telescopio y su montura. A esta propiedad debe sumársele la mejor capacidad de soporte del espejo primario ya que, al estar éste más cerca del plano del observador, el espejo se encuentra solamente depositado sobre su base, de tal manera que no necesite sujetarse y así evitar esfuerzos mecánicos que provocarían deformaciones en el espejo.

Así, este sistema óptico facilita la realización de espejos-objetivos con grandes aberturas y focos reducidos, lo que se traduce en la utilización de tubos más cortos. Todas estas características hacen del telescopio reflector uno de los modelos más populares para los grandes observatorios modernos, como se muestra en la figura 1.6, donde se muestra cómo el sistema reflector permite telescopios con mayor abertura que

los refractores y de mayor eficiencia. El reflector mostrado es del Observatorio Monte Palomar, California; se inauguró en 1948 y es el segundo más grande del mundo. El refractor mostrado es del Observatorio de Cincinnati, Ohio, y fue el primer telescopio profesional de América, inaugurado en 1845.

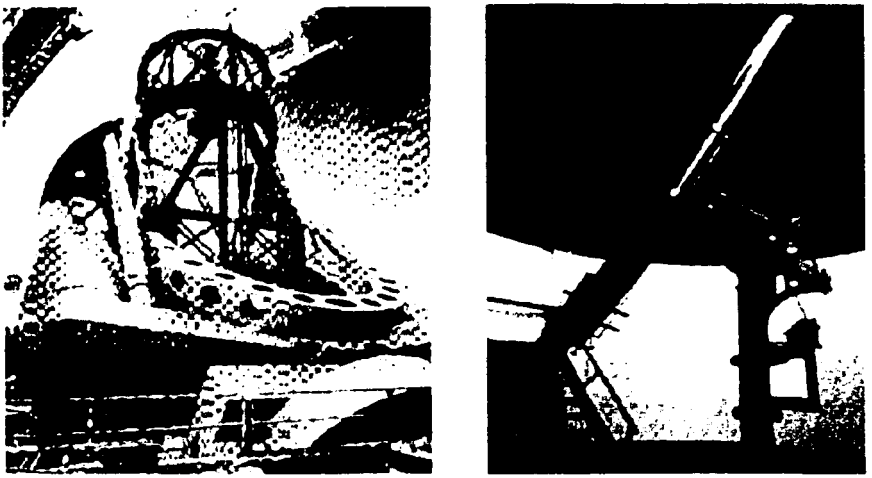


Figura 1.6. Comparación entre el sistema reflector (izq.) y el refractor (der.).

1.1.3 MONTURAS

El mecanismo que soporta y facilita el movimiento del tubo del telescopio se denomina montura. Al igual que los sistemas ópticos de los telescopios, las monturas de éstos han presentado una mayor sofisticación conforme su evolución para facilitar las observaciones y el guiado de los instrumentos colocados en ellos. Una montura automatizada resulta imprescindible para la obtención de fotografías, análisis espectrales

o procesos fotoeléctricos con cámaras CCD. Es sabido que, para registrar las débiles imágenes estelares, deben otorgarse tiempos de exposición largos durante los cuales los pequeñísimos focos luminosos de cada estrella deben caer siempre en el mismo punto del CCD o de la película fotográfica; en caso contrario, el análisis o la fotografía resultaría borrosa o movida. Sólo un mecanismo muy preciso y la pericia del usuario controlándolo permiten conseguir excelentes imágenes estelares. Entre los tipos de monturas que sobresalen para alcanzar este fin han destacado dos tipos: la de tipo acimutal y la de tipo ecuatorial, con sus múltiples variantes.

- *LA MONTURA ACIMUTAL*

La montura acimutal es la más simple de las monturas. Se compone de un eje horizontal, paralelo al horizonte, y otro eje perpendicular (eje vertical) que permiten dirigir el tubo del telescopio a cualquier punto. Un tripié fotográfico es un ejemplo sencillo de una montura acimutal, pudiéndose considerar como el principio del que evolucionaron distintas versiones, desde la del telescopio de William Hershel a principios del siglo XIX (fig. 1.7), hasta su desarrollo completo en versiones como la del telescopio de 2 mts. del Observatorio Astronómico Nacional de San Pedro Martir, en Baja California.

Los movimientos de los ejes no obedecen sólo al proceso de localizar y centrar una imagen en el telescopio, sino que también se utilizan para seguir a los astros en sus movimientos aparentes en la bóveda celeste, cuando recorren una trayectoria circundante al polo norte celeste (con excepción de los planetas, cometas y meteoritos, que siguen su propia trayectoria). La bóveda celeste es como una inmensa cúpula y pareciera que girara lentamente. En realidad, es la rotación de la Tierra sobre su propio eje el causante de estos "recorridos". De esta manera, las trayectorias de los astros variarán con respecto al

polo norte celeste dependiendo de la latitud en que se encuentre localizada la montura del telescopio, describiendo grandes circunferencias, como se analizará en el siguiente apartado.

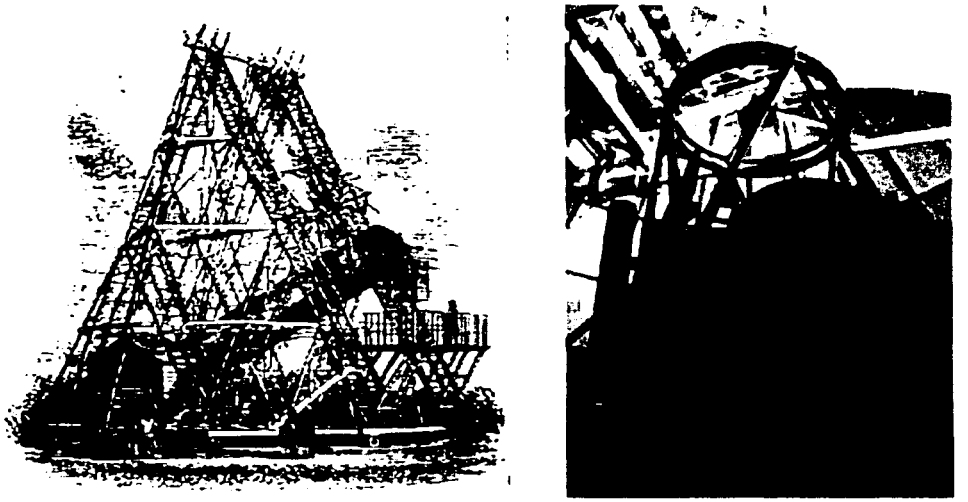


Figura 1.7. Evolución de la montura acimutal: de Hershel (izq.) al OAN San Pedro Mártir [1] (der.).

Para el seguimiento de un astro que se mueve en trayectorias esféricas (fig. 1.8) se hace necesario un giro sincronizado y constante de los ejes X,Y de la montura acimutal. Coordinar estos movimientos con la rotación de la bóveda celeste se logra gracias al control proporcionado por las computadoras. Es importante mencionar que las monturas acimutales en los grandes observatorios son una necesidad cuando el diámetro de los espejos supera los 6 metros. Sin embargo, cuando se trabaja con telescopios de menor

[1] Foto cedida por Abel Bernal Bejarle

diámetro, existe otro tipo de montura que, si bien no es tan simple como la acimutal, ofrece una solución mucho más elegante para contrarrestar el movimiento aparente de la bóveda celeste.

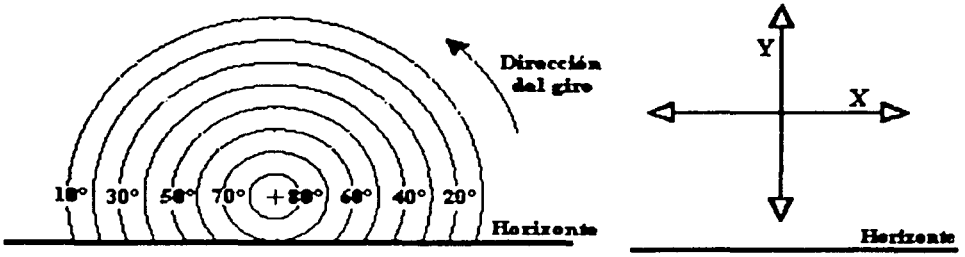


Figura 1.8. Comparación entre el movimiento aparente de los astros en la latitud 20° Norte (izq.) y los movimientos posibles de la montura acimutal (der.).

- *LA MONTURA ECUATORIAL*

Una montura ecuatorial es un sistema giratorio que sirve para contrarrestar el movimiento de rotación de la Tierra. Gracias a ella, un telescopio puede "fijarse" artificialmente en dirección a una región del cielo. Básicamente, una montura ecuatorial puede compararse a una montura acimutal a la que se le haya inclinado su eje vertical hasta situarlo paralelo al eje de rotación de la Tierra. De tal manera, cuando se da un movimiento de giro a este eje, con velocidad de una vuelta cada 24 horas en sentido contrario al de rotación terrestre, se consigue que el telescopio siga automáticamente cualquier astro en su desplazamiento aparente por la bóveda celeste (fig. 1.9). Así, al apuntar a cualquier astro el telescopio ecuatorial en el momento de su salida (horizonte Este a primera hora de la mañana), lo irá siguiendo todo el día, para terminar en el horizonte Oeste al atardecer, sin necesidad de ajustar ningún eje de la montura. El eje

motor de la montura ecuatorial que da un giro completo cada 24 horas se llama eje polar u horario. Este eje, también llamado eje de ascensión recta (AR), es el utilizado para guiar el telescopio por la coordenada celeste denominada ascensión recta, cuyas unidades son las horas, minutos y segundos. Este eje sirve en astronomía para desplazar al telescopio por las coordenadas celestes que equivalen a la longitud terrestre. El otro eje, perpendicular al anterior, se le denomina eje de declinación (DEC) y sus unidades son los grados de ángulo.

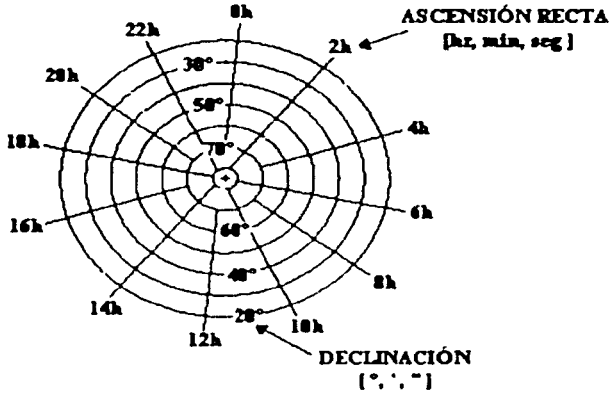


Figura 1.9. Movimientos de la montura ecuatorial

La declinación puede ser positiva o negativa, en un rango de +90°, que es el polo norte de la bóveda celeste, hasta -90°, que corresponde al polo sur celeste. El ecuador celeste se localiza a 0° y es la proyección del ecuador celeste en la esfera celeste.

La ascensión recta describe una circunferencia de 360° que, para fines prácticos de localización y seguimiento de los astros, se divide en segmentos de 15°. Cada segmento equivale a una hora y ésta se divide en 60 minutos y estos a su vez se dividen

en 60 segundos. La hora cero, el equivalente al meridiano de Greenwich en la Tierra, es el punto de la bóveda celeste donde se cruza la eclíptica y el ecuador celeste. Una de las razones de utilizar estas unidades para el eje de ascensión recta es debido a la facilidad que proporciona este sistema para seguir a un astro con una montura ecuatorial. En esta montura, una vez localizado un astro celeste, el eje de declinación no es vuelto a alterarse, mientras que, el eje de ascensión recta, moviéndose a una velocidad exactamente igual a la rotación de la Tierra, pero en sentido contrario, realiza un seguimiento automático del astro a observar.

La rotación de la tierra se mide en segundos terrestres. Sin embargo, como la rotación de un punto sobre la Tierra varía según la latitud en que se encuentre, en Astronomía se utiliza una frecuencia sideral que está determinada por la duración de la rotación de un punto específico sobre la Tierra; por ejemplo, la duración de la rotación de un punto sobre la Tierra sobre su propio eje en el ecuador es de 23.9 hrs. Este punto específico es la ubicación geográfica de cada Observatorio. La unidad de esta frecuencia sideral es el Hertz Sideral y la duración del segundo sideral es distinto al segundo terrestre. Por este motivo, en los Observatorios se cuenta siempre con un reloj que ofrece la hora local sidérea, que es función de la latitud del observatorio, su altura sobre el nivel medio del mar, la velocidad de rotación de la Tierra y otros factores, para proporcionar posteriormente, la frecuencia sideral con la que se moverá el eje horario de la montura ecuatorial.

La montura ecuatorial es de simple concepción, aunque en la práctica resulta bastante compleja porque su centro de gravedad varía según sea la posición del telescopio. Para su correcto funcionamiento es absolutamente imprescindible que el eje horario (el eje que equivaldría al vertical en las monturas acimutales) esté exactamente orientado como el eje de rotación terrestre. La operación es relativamente fácil dado que,

previamente, estas monturas han inclinado este eje tantos grados como sea la latitud del lugar.

Una ventaja que presenta este mecanismo es la capacidad de localizar cualquier astro cuyas coordenadas sean conocidas. En teoria, si la montura está correctamente orientada y si se dispone de la hora sidérea (sistema horario del firmamento) puede buscarse el astro mediante la utilización de unos círculos graduados que la montura contiene en ambos ejes.

Existen diversos tipos de monturas ecuatoriales en función de la clase de telescopio que deben aguantar o de su peso, pero la mayoria son modificaciones de diseños básicos. Las dos versiones que más se han desarrollado son la montura ecuatorial tipo alemana y la montura ecuatorial de horquilla.

La montura ecuatorial tipo alemana es aquella con la que se soportaban todos los grandes refractores del milenio pasado. Básicamente estaba conformado por un eje de ascensión recta que apuntaba al polo norte y un eje perpendicular formando una T con el anterior, en una de cuyas puntas se instalaba el telescopio y en la otra contrapesos para que el centro de gravedad del conjunto quedara sobre el eje de ascensión recta. Esta configuración era apropiada para telescopios de pequeña abertura, ya que este tipo de montura es muy voluminosa. En la actualidad existe una versión de montura mucho más robusta.

La montura ecuatorial de horquilla (ver fig. 1.10), tiene el eje de ascensión recta constituido por un diapasón rematado por dos pivotes en los extremos de sus brazos. De esta manera, el telescopio se mueve en el interior del diapasón, por lo que esta disposición presenta una gran solidez para aguantar el enorme peso de los grandes telescopios (del orden de las Toneladas), haciendo que la montura de horquilla sea uno de

los sistemas predominantes de montura ecuatorial en los observatorios astronómicos modernos.

Figura 1.10. Montura ecuatorial de horquilla del OAN, Tonantzintla

Finalmente, habrá que hacer notar que cada telescopio aporta una solución diferente para compensar el gran esfuerzo de palanca que ha de soportar la caja de cojinetes.

La interfaz a desarrollar en este trabajo estará diseñada para mover los dos ejes de un telescopio con montura ecuatorial. Esto significa que los movimientos que controlará son cuatro: ascensión recta positiva, ascensión recta negativa, declinación positiva y declinación negativa. El movimiento de cada eje será independiente del otro y, una vez localizado el astro a observar con el telescopio, la ascensión recta será el único movimiento que continuará alterándose. Este eje girará a una velocidad de guiado específica, determinada en función de la frecuencia sideral. Esta frecuencia ingresará a la interfaz directamente del reloj astronómico del Observatorio.

1.1.4. USO DE COMPUTADORAS CON LOS TELESCOPIOS

La mecanización de las monturas ha tenido una gran evolución desde el siglo XVIII, en el cual se comenzaron a utilizar mecanismos de relojería para automatizar los movimientos de los telescopios. Principalmente, esta automatización iba orientada al eje de ascensión recta para que tuviera un movimiento tal que siguiera a los astros, una vez localizados, por toda la bóveda celeste, sin necesidad de intervención del usuario. Esta automatización representó un gran avance comparado con los mecanismos de guiado de monturas anteriores, basados en un sistema de manivelas giradas manualmente por un operador.

Sin embargo, los mecanismos de relojería sufrían en la precisión de su operación debido a cambios climáticos tales como la temperatura o la humedad, añadiéndose a sus propias limitaciones naturales, debido a los engranajes del mecanismo. Estas limitaciones se fueron perfeccionando conforme mejoraban los métodos de manufactura de los mecanismos, pero la introducción de motores eléctricos para mecanizar las monturas fue el gran avance que permitió seguimientos más precisos.

Los sistemas eléctricos permitieron resolver un inconveniente siempre presente en los grandes observatorios, el cual no contaba, hasta ese momento, con una solución satisfactoria. Debido a la desigual distribución del peso en el tubo del telescopio, como en su montura, los esfuerzos mecánicos a los que son sometidos los mecanismos de guiado son diferentes, según la posición del telescopio. Es decir, los mecanismos de las monturas realizan menos esfuerzo al seguir un objeto cerca del cenit, el punto más alto del firmamento, que cerca del horizonte. Debido a esta diferencia de esfuerzos, las velocidades de guiado podían diferir según la posición del telescopio, resultando esto en un seguimiento deficiente de los astros. Sin embargo, con la llegada de los motores

eléctricos, los sistemas se fueron sofisticando al integrar dispositivos para el control automático, tales como los controladores con retroalimentación negativa.

Pronto, conforme los sistemas de control analógicos y digitales fueron aumentando, se vió la utilidad de gobernarlos por medio de las computadoras. La utilización de éstas no sólo permite un mejor y más sencillo control de guiado de las monturas sino que, además, son adaptables a distintos observatorios localizados en diferentes localizaciones geográficas, impedimento que tenían los antiguos sistemas de control, que eran diseñados específicamente para determinadas latitudes. Igualmente, gracias a las bases de datos, los telescopios actuales se pueden dirigir automáticamente a cualquier astro cuyas coordenadas celestes se conozcan y sean ingresadas a la computadora. Adicionalmente, la utilización de módem conectados a las computadoras de los observatorios, con su conexión a la red de Internet, permiten realizar observaciones remotas, al controlar el telescopio desde centros localizados a muchos kilómetros de distancia del observatorio. Esto se traduce en una mejor accesibilidad al telescopio para distintos astrónomos y una mayor difusión del mismo.

1.2. PLANTEAMIENTO DE LA NECESIDAD

La interacción de la computadora con los distintos periféricos del telescopio tales como el controlador manual (*handset*, por sus pronunciación en inglés), el controlador de los motores de los ejes del telescopio, el control de los movimientos de la cúpula del observatorio, los distintos sensores, indicadores, etc., hacen necesario un *hardware* que sirva de interfaz entre estos periféricos y la computadora. Este *hardware* es conocido como tarjeta de interfaz y está diseñado para operar con un *hardware* específico que interrelacione los distintos dispositivos a controlar.

Estas interfaces son medios de captura y transmisión de señales digitales y/o analógicas cuyo diseño está en función de las necesidades de la computadora del observatorio. Así, las características para los cuales están diseñadas estas interfaces deben cumplir ciertos requisitos particulares para que se adecuen a cada observatorio. Pero, igualmente, estas particularidades deben englobarse dentro de un concepto de diseño general, común a las monturas ecuatoriales y sus controladores de motores, que hagan que una interfaz bien diseñada, junto con su *hardware*, pueda adaptarse fácilmente a distintos observatorios.

Existen en el mercado tarjetas de propósito general que son ofrecidas para el control de distintas máquinas en la industria. Sin embargo, la adecuación de este tipo de *hardware* al control de un telescopio provoca varios escenarios posibles que presentan los siguientes inconvenientes:

a) Necesidad de comprar computadoras específicas al fabricante, compatibles con su producto.
b) Incapacidad de operar junto con otros controladores de distinta marca simultáneamente, debido al uso de distintos protocolos de comunicación.
c) Falta o difícil acceso al código fuente del programa, lo que impide mejoras que lo adecuen particularmente al observatorio para lograr una mayor eficiencia.
d) Imposibilidad de realizar modificaciones a la tarjeta que permitan cambiar la operación de la interfaz.
e) Probable subutilización de las capacidades del *hardware* ya que, al ser tarjetas de propósito general, vienen implementadas en ellas funciones que pueden no ser empleadas.
f) Difícil adecuación del sistema para operar en las muy particulares características de un observatorio.

g) Largos tiempos de espera, tanto para obtenerlos como para repararlos, cuando los tiempos de observación son valiosos.
h) Alto costo.

De lo anterior se hace necesario la creación de una interfaz que salve los inconvenientes mencionados de manera eficiente y con tecnología de punta.

1.2.1. CASO DEL OBSERVATORIO ASTRONÓMICO NACIONAL DE TONANTZINTLA

El origen del Observatorio Astronómico Nacional (OAN) se remonta a 1867 cuando se fundó un pequeño observatorio en la azotea del Palacio Nacional en el centro de la Ciudad de México. Como consecuencia del crecimiento de la ciudad, este observatorio fue trasladado primero al Castillo de Chapultepec en 1878 y posteriormente al edificio conocido como el Observatorio de Tacubaya, inaugurado en 1908. Cuando se expidió el decreto de autonomía de la UNAM en 1929, el OAN fue incorporado a la Universidad, aunque se mantuvo su sede en Tacubaya. En el momento que las condiciones atmosféricas en Tacubaya se convirtieron en poco favorables a la observación en 1951, se trasladó la estación del Observatorio Astronómico de Tacubaya a Tonantzintla, Puebla, contiguo al Observatorio Astrofísico Nacional de la Secretaría de Educación Pública. En 1967, se le reconoció la categoría de Instituto de Investigación al OAN, por lo que se creó el Instituto de Astronomía de la UNAM (IAUNAM). Así, el nombre OAN se ha reservado para las estaciones de observación que dependen de este Instituto, como es el caso del Observatorio Astronómico Nacional de Tonantzintla.

Cuando ocurrió el traslado de los telescopios de Tacubaya a Tonantzintla, el principal instrumento era el telescopio bautizado como "Carta del Cielo". Sin embargo,

en 1961 se inauguró un nuevo telescopio: un instrumento moderno de 1 metro de diámetro (fig. 1.11) y desde entonces es el principal telescopio, junto con sus instrumentos de apoyo, en esta localidad.

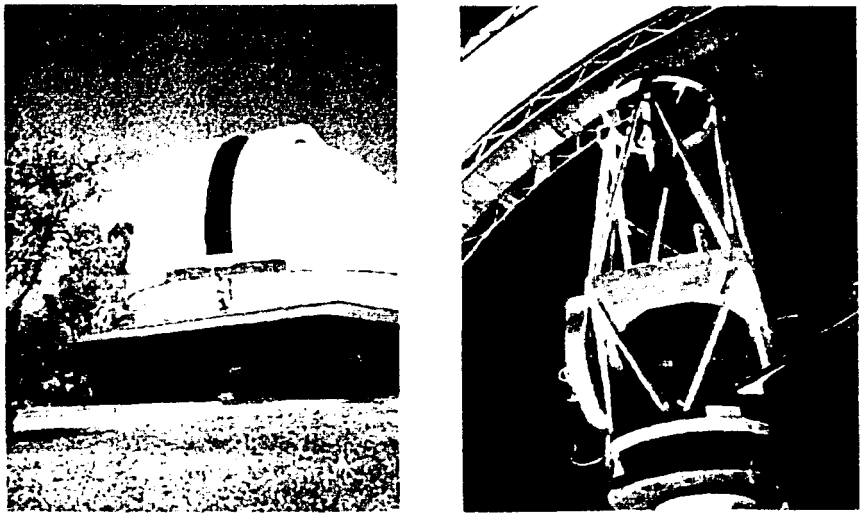


Figura 1.11 OAN Tonantzintla (izq) y su reflector de 1 m con montura ecuatorial de horquilla (der)

Este telescopio es del tipo reflector, conocido como "1 metro", a pesar de que su espejo principal tiene un diámetro de 1.016 m, ya que el diámetro óptico es de 1.000 m. El espejo secundario tiene un diámetro óptico de 371.8 mm y está colocado a una distancia de 3.691 m del espejo primario. El movimiento de enfoque se realiza con una carrera de diseño de 80 mm. Esta carrera, similar al movimiento del tornillo que se utiliza para calibrar el enfoque en unos binoculares, es suficiente para poderle adaptar distintos tipos de oculares, cámaras, espectrógrafos y demás instrumentos científicos.

El telescopio está colocado en una montura ecuatorial de tipo horquilla y sus límites de apuntado son entre -5.6 y 5.6 horas en el eje de la ascensión recta y entre – 60° y +80° en el eje de la declinación. Estos límites son fijados en el programa de computadora que controla los movimientos del telescopio para asegurar la integridad del espejo principal ya que éste, para evitar deformaciones, no se encuentra sujeto de ninguna manera al armazón del telescopio, sino que está superpuesto solamente. De esta manera, al fijar los límites de apuntado del telescopio, se evita que éste se llegara a inclinar tanto como para que el espejo principal pudiera volcarse y, por lo tanto, romperse.

La máxima velocidad de apuntado de este telescopio es de 1 grado/segundo y su mínima resolución de posicionamiento es de 1/3". La velocidad de control manual es definida por la posición de potenciómetros en el controlador manual del telescopio.

El sistema de control encargado del posicionamiento del telescopio y de su guiado se denomina consola de control. No sólo contiene el programa con el que se opera el telescopio, sino que, además, cuenta con el catálogo *Brigth Star*, una base de datos de las posiciones de la mayoría de los astros celestes con el cual se puede orientar rápidamente el telescopio. En caso de que el astrónomo deseara utilizar otro catálogo, éste puede llevar en un disco su propia secuencia de observación. La característica principal de esta consola de control es que originalmente fue diseñada con una computadora personal (PC) tipo PC-286 en el año de 1992.

- *ACTUALIZACIÓN DE LA CONSOLA DE CONTROL*

Conforme pasó el tiempo, se vio la necesidad de actualizar la computadora del sistema de control del telescopio del OAN Tonantzintla. En el año de 1999, se intentó el

cambio de una PC-286 a una PC-486. Sin embargo, la tarjeta de interfaz de la PC-286 presentó fallas en su funcionamiento en la PC-486. Estas fallas fueron, principalmente, la pérdida de reconocimiento de las señales provenientes del controlador manual. Fue en este punto, y debido a la rápida evolución de los microprocesadores, que se comenzó a planear la creación de una nueva tarjeta de interfaz mucho más ambiciosa, que funcionara en una PC tipo Pentium. Este plan que se concretó a principios del año 2000, cuando se creó el Programa Sistema de Interfaz y Control de Telescopio

En el programa Sistema de Interfaz y Control de Telescopio se desarrolló la tarjeta MAGALI (Medio Acoplador para Guiado Astronómico Lógico como Interfaz) en el Laboratorio de Electrónica del Instituto de Astronomía de la UNAM. Esta tarjeta no sólo funciona en las tarjetas madre tipo Pentium, sino que también presentó varias ventajas comparadas con su antecesora, tales como la capacidad de decodificar la dirección del puerto al que se van a enviar los datos por medio de *dip-switches*, y la inclusión de los dispositivos necesarios para leer y calibrar las mediciones analógicas provenientes de distintos sensores.

La primera ventaja, la opción de poder variar las direcciones a la que se envía la información, proporciona una mayor flexibilidad para la instalación de la tarjeta ya que, si algún otro dispositivo conectado a la computadora utilizara las mismas direcciones que la interfaz (o existiera un traslape de direcciones), y a ésta no se le pudieran cambiar por otras nuevas, existiría un mal funcionamiento en la comunicación entre estos dispositivos con la computadora.

La ventaja de implementar sensores que se comuniquen directamente con el programa de control hace que, segun un monitoreo constante de las condiciones climáticas, el programa tome decisiones o advierta al operador cuando existan condiciones adversas que atenten contra la integridad del telescopio. Estas condiciones

adversas son aquellas que dañan los lentes del telescopio y pueden ser muy diversas, siendo las principales la condensación, es decir, cuando la humedad es tan alta que se forman gotas de agua en el espejo del telescopio (normalmente en la madrugada, antes del rocío) y las cenizas provenientes del volcán Popocatépetl. En particular, el monitoreo constante de la humedad en el ambiente y la temperatura, permiten establecer y predecir si las condiciones climatológicas dañan o no los espejos del telescopio y sus mecanismos. Así, al contar con mayor información de las condiciones ambientales, la subjetividad en la apreciación de las condiciones climatológicas es menor y de esta manera el factor de equivocación humana, para decidir entre operar o no el telescopio, puede reducirse. Una consecuencia directa de este control es que se alargan los tiempos de operación útiles del observatorio, al espaciar más sus períodos de mantenimiento.

1.2.2 DEMANDA DE UNA NUEVA INTERFAZ ELECTRÓNICA

Una vez actualizada la consola de control del OAN Tonantzintla con el cambio a una computadora Pentium III y su nueva tarjeta de interfaz, a principios del 2001, surgió el interés de implementar otras tarjetas MAGALI para poder actualizar, igualmente, las computadoras de otros observatorios, en particular el Observatorio de la Luz de la Universidad de Guanajuato.

Sin embargo, durante la implementación física de la tarjeta MAGALI no se contempló la posibilidad de fabricar más de una de manera practica. Esto es debido a las siguientes razones principalmente:

a) La tarjeta MAGALI fue realizada sobre una tarjeta de desarrollo PCL-750, cuya adquisición no es sencilla al ser un producto ya no disponible en el mercado.
b) El alambrado de todos los componentes añadidos a la tarjeta fue por técnica de *wire-wrap* (en castellano: enroscado de cables). Esta técnica consistió en

utilizar finos cables con un ligero recubrimiento de plata para que, al torcer el cable revestido con las terminales de la bases de los circuitos integrados, se provocara una pequeña fusión a nivel molecular al derretirse la plata entre el cable y las terminales de las bases en el momento de alambrarlos, lo que garantizaba un perfecto contacto entre ellos. Esta técnica implica un enorme tiempo de alambrado y verificación de éste, debido principalmente a que el número de circuitos integrados a entrelazar es considerable.

Así, reproducir a la tarjeta MAGALI con estos métodos artesanales se hace impráctico e inefectivo, por lo que se hace necesario la creación de otra interfaz, compatible con las PC-Pentium, que pueda elaborarse con menos componentes y por técnicas diferentes para optimizar su producción y abreviar sus periodos de pruebas.

Cabe mencionar que la opción de crear un circuito impreso de la anterior interfaz no fue considerada debido a la filosofía de los miembros del Laboratorio de Electrónica del Instituto de Astronomía. Esta filosofía busca la investigación de nuevos métodos de diseño y técnicas de manufactura de productos electrónicos, para innovarse e incrementar su radio de conocimiento. La opción que representa la interfaz que será desarrollada en este trabajo resulta renovador en este sentido, ya que obedece a la tendencia que existe de optimizar cada vez más los *hardware* existentes en aspectos fundamentales tales como el consumo de energía, la reducción de tamaño, la reducción de costos, la facilidad de instalación, etc.

Una interfaz para un telescopio de las dimensiones de un OAN no se encuentra disponible en el mercado, como se encuentran las populares tarjetas de video y de sonido y, por las razones expuestas anteriormente, las tarjetas industriales de propósito general no son las ideales para los Observatorios. Esto hace de la opción de una interfaz electrónica, embebida y personalizada, una opción atractiva para los observatorios que

buscan mantener al máximo su infraestructura existente, minimizando el número de componentes a actualizar en sus sistemas para estar acorde a los requerimientos de la tecnología actual. Algunos de estos observatorios pueden optar por la tarjeta de interfaz propuesta en este trabajo o utilizar este proyecto como base de su propio desarrollo. Teniendo en cuenta que con el paso del tiempo cada vez son más las consolas de control que necesitan actualizarse, existirán en el futuro otros observatorios, además del OAN Morelia y la Universidad de Guanajuato, que requieran una nueva interfaz electrónica.

Igualmente, esta nueva tarjeta puede ser utilizada en nuevos observatorios que estén próximos a construirse, como es el caso del Telescopio Infrarojo Mexicano (TIM), proyecto del IAUNAM a desarrollarse prontamente en San Pedro Mártir, Baja California.

1.3. OBJETIVOS E IMPORTANCIA DE LA ELABORACIÓN DEL PROYECTO

La importancia de la elaboración de una interfaz electrónica, tanto para datos analógicos como digitales, para un telescopio, radica en desarrollar un nuevo *hardware* de simple producción con la tecnología de punta disponible. Esta interfaz debe ser, a la vez, de sencilla adaptación al *hardware* y al *software* ya existentes en los observatorios, para su actualización, de tal manera que cumpla con los siguientes objetivos:

- Proporcionar una propuesta de diseño de una interfaz competitiva en funcionamiento, disponibilidad y costo.

- Optimizar y ampliar los recursos disponibles para la automatización de los telescopios según las necesidades y las demandas que la modernización de los Observatorios implica, con la mayor eficiencia posible.

- **Mostrar la aplicación de los avances tecnológicos que presenta la electrónica digital, en particular los Dispositivos Lógicos Programables Complejos, a una aplicación concreta de la Ingeniería.**

- Continuar con la promoción del desarrollo de *hardware* especializado, acorde a los requerimientos de la sociedad mexicana, en la Universidad Nacional Autónoma de México.

Es un hecho que la ingeniería electrónica se beneficia continuamente de las profundas actualizaciones y transformaciones producto de las innovaciones en otras ingenierías y las ciencias de los materiales. Es importante la búsqueda de la aplicación de estas tecnologías y su adaptación al medio y a la realidad de nuestra sociedad. De aquí la proyección de este trabajo a las tecnologías que tienden a predominar en el mercado de la electrónica actual.

1.3.1. LOS SISTEMAS EMBEBIDOS

Existen dos grandes campos en la electrónica: la analógica y la digital. La primera, que se pensaba sería reemplazada paulatinamente por la segunda, ha tenido un resurgimiento importante en las últimas fechas. Sin embargo, desde la aparición del circuito integrado en 1960, la electrónica digital ha tenido un enorme avance, tanto en su capacidad de optimizarse, como en su capacidad de empleo. Un sistema digital sólo trabaja con dos estados lógicos que se representan por dos tipos de tensión o voltaje. A uno de estos estados se le denomina *alto* (High) y, por lo general, supone la existencia de una tensión de +5V que representa al bit 1 en el sistema binario. Al otro estado se le denomina *bajo* (Low) y suele estar definido por una tensión cercana a 0V que representa

al bit 0 del sistema binario. A partir de 1971, con la fabricación del microprocesador en un circuito integrado, su manejo se popularizó mundialmente.

Debido a los altos niveles de producción de estos microprocesadores, su costo se ha visto reducido dramáticamente. Este abaratamiento se ha visto reflejado en la cantidad de dispositivos basados en una versión de los microprocesadores, los microcontroladores, que se han vuelto parte de la vida cotidiana, ya que un microcontrolador es un microprocesador con memoria y chips de entrada/salida en estructura similar a una micro computadora, pero con menor capacidad, fabricado en un solo integrado y sin la cantidad de periféricos de una micro computadora.

Así, esta disponibilidad permite múltiples aplicaciones que, con la natural evolución de estos dispositivos, van presentando mejoras en su funcionamiento y en su presentación. El efecto más reconocible debido a estas mejoras es la miniaturización de los aparatos electrónicos. Esta miniaturización no sólo es debido a la reducción de tamaño de los componentes electrónicos, sino, principalmente, a la integración, cada vez mayor, de funciones en los microcontroladores.

La integración no sólo cumple fines estéticos y ergonómicos, sino que, al integrar los circuitos electrónicos en un microcontrolador, se presentan las siguientes ventajas:

- Existe un aumento de fiabilidad por la eliminación de la lógica cableada, al tener menor número de elementos e interconexiones.
- Se pueden reemplazar un número elevado de circuitos integrados, lo cual presenta ventajas en cuanto a la facilidad del diseño.
- Se permite desarrollar sistemas en forma modular y estructurada.
- Se pretende disminuir el consumo de energía.

- Se trabaja con estructuras de información más desarrolladas, es decir, las operaciones lógicas y aritméticas pueden ser más complejas, aumentando la capacidad de decisión del sistema.
- Se permiten modificaciones que permitan futuras expansiones del sistema al ser éste más flexible.
- Se aumenta la capacidad de desarrollo al tener un gran número y diversificación de sistemas de entradas y salidas.
- Se simplifica el mantenimiento, la diagnosis de fallos y las reparaciones.
- El período de desarrollo y comercialización del equipo se ve reducido.
- Se reducen los costos de producción.

Estas características son las que buscan los sistemas embebidos.

Las principales clases de dispositivos lógicos programables que permiten la creación de sistemas embebidos, bajo el concepto de microcontrolador, en orden ascendente de complejidad en su estructura interna son: los PALs (*Programmable Array Logic*), los GALs (*Generic Array Logic*), los PLAs (*Programmable Logic Arrays*) los PICs *(Programmable Integrated Circuits)*, los PLDs *(Programmable Logic Devices)*, los FPGAs *(Field Programmable Gate Arrays)* y los CPLDs *(Complex Programmable Logic Devices)*. En esta lista no se incluyen, intencionalmente, ni las matrices de compuertas *(Gate Arrays)*, ni los ASICs *(Application Specific Integrated Circuits)* ya que éstos sólo se justifican para volúmenes grandes de producción, siendo su tiempo de construcción de unos meses a unos años, lo que los hace inviables para la proyección de este proyecto.

En los CPLDs es posible implementar casi cualquier diseño, permitiendo la adaptación del circuito a una aplicación determinada, pero manteniendo su función básica. Así, estos circuitos están muy optimizados para la función para la que han sido desarrollados. Es por estas razones que el presente trabajo propone el diseño de una

interfaz para datos, analógicos o digitales, para el control de un telescopio, instalado éste en una montura ecuatorial, bajo el concepto de un sistema embebido, con la lógica de un microcontrolador, realizado en un CPLD.

Una vez analizados todos estos puntos, se puede proceder a realizar una descripción de los elementos que realizan la interfaz electrónica, particularizando en los requerimientos para la comunicación con la micro computadora y en las características de operación del CPLD utilizado en este trabajo, además de tratar su *hardware* de programación.

II

DESCRIPCIÓN DE LOS ELEMENTOS DE LA INTERFAZ

El análisis de las principales características del microprocesador Pentium de INTEL, permite garantizar la correcta comunicación entre éste y un *hardware* con ancho de *bus* más pequeño mediante la interfaz electrónica. La interfaz utiliza el *bus* ISA de la microcomputadora, por lo que la comprensión de las principales características de éste, de las transmisiones de datos, de las interrupciones utilizadas, y de la decodificación de puertos es importante para apreciar el papel que desempeñará el CPLD en la interfaz. Igualmente, al realizar un examen detallado de los dispositivos MAX7000, se vislumbran las ventajas que ofrece la programación con el *software* Max+Plus II de Altera para la realización de este proyecto.

2.1. *ANÁLISIS DE ASPECTOS GENERALES*

La comunicación de una computadora con el mundo exterior requiere la entrada de datos *a* la computadora *desde* sus periféricos y la transferencia de datos procesados, o señales de control, *de* la computadora *a* los periféricos. En particular, los periféricos con los que se realizará la interfaz y la computadora de la consola de control del Observatorio son: el controlador de movimientos del telescopio, el reloj astronómico, el controlador de motores denominado *Galil* y los sensores ambientales.

Para realizar la interfaz entre la computadora y estos periféricos, se requiere, básicamente, de un dispositivo que proporcione cuatro funciones: una decodificación de direcciones para la selección de periféricos, ya sean de entrada o de salida, una decodificación de instrucciones, un registro para el control de tiempos de operación (*timing* en inglés) y una codificación/decodificación de los datos enviados/recibidos por los periféricos, a un formato que tanto el microprocesador como el *software* sean capaces de manipular.

Las razones de ser de cada una de las tres primeras funciones son las siguientes:

1.- Para que la computadora pueda acceder a los periféricos, se utiliza un *bus* de datos externo. El *bus* es el conjunto de cables con el que están conectados los distintos componentes de un sistema, para la transmisión de señales digitales entre ellos, mediante un protocolo establecido. Para utilizar el *bus* de datos externo, el microprocesador de la computadora debe suministrar la dirección del registro donde se encuentra codificado el periférico, para que posteriormente sea decodificada. Sin este registro correcto, la transferencia de datos puede equivocarse de periférico o simplemente no tener comunicación con él.

2.-Una vez que se ha seleccionado el periférico deseado, mediante su dirección, el microprocesador debe informar al periférico lo que debe hacer. En muchos casos, la decodificación de instrucciones es una simple señal de habilitación que activa o desactiva determinada función del dispositivo.

3.- Debido a que las computadoras y los periféricos raramente funcionan a la misma velocidad, se necesitan registros de datos para mantener, por ciertos periodos, los datos durante las transferencias de datos, ya sean de entrada o de salida. El objetivo de los registros de datos es compensar la diferencia de operación entre el microprocesador, generalmente más rápido, con los periféricos más lentos. Así, se sincronizan las comunicaciones entre el microprocesador y los periféricos.

El dispositivo que cumple la tarea de adquirir los datos provenientes de los periféricos solventando las tres funciones anteriores, se conoce como tarjeta de adquisición de datos.

Cuando existe la opción de que el tipo de datos que envían los periféricos hacia la interfaz electrónica sean datos analógicos o digitales, siendo que la comunicación entre la interfaz electrónica y el microprocesador es en todo momento digital, se hace necesaria la cuarta función a realizar por la tarjeta: la decodificación de los datos recibidos a un formato capaz de compatibilizar esta información con la estructura interna de la computadora, antes de ingresar al *bus* de datos que se dirige al microprocesador. Así, cuando la interfaz electrónica cumple adicionalmente este cuarto punto y opera como el dispositivo de transferencia entre los periféricos y la computadora, recibe el nombre de interfaz electrónica (fig. 2.1).

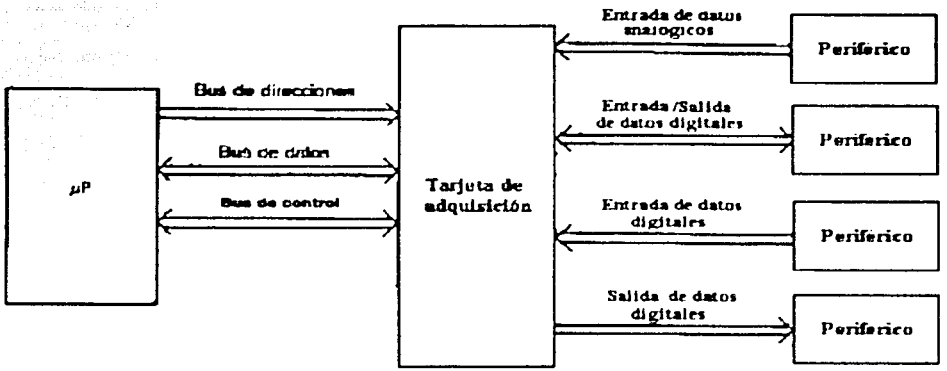


Figura 2.1. Diagrama de bloques de una interfaz electrónica.

El conocimiento de la estructura interna, tanto del microprocesador como de la interfaz electrónica con la que se va a comunicar, y del protocolo de intercambio de información entre otros, sienta las bases para el diseño de la interfaz desarrollada en este trabajo.

2.1.1. ARQUITECTURA DEL MICROPROCESADOR PENTIUM

Gracias a que Intel buscó garantizar la total compatibilidad entre el microprocesador Pentium con los procesadores existentes 80486, 80386, 80286 y anteriores, la arquitectura del microprocesador es muy similar a la arquitectura de los módulos funcionales del microprocesador 80486.

En la tabla 2.1 se muestra la comparación entre la PC-286, la PC-486 y la PC tipo Pentium. De esta comparación parecería evidente que el microprocesador Pentium es el simple resultado de la evolución de los microprocesadores de Intel. Sin embargo, el

microprocesador Pentium se distancia de sus antecesores ya que en éste se pudo alcanzar un grado de integración de 3.1 millones de transistores, presentando así una arquitectura interna que posibilitó que dos instrucciones puedan ser procesadas en un solo ciclo de reloj.

Características	*80286*	*80486 SX*	*Pentium*
Año de introducción	1982	1989	1993
Bus de datos	16 bits	32 bits	64 bits
Bus de dirección	24 bits	32 bits	32 bits
Vía interna de datos	16 bits	32 bits	64 bits
Frecuencia de Reloj	8 –12 MHz	16-100 MHz	200 + MHz
Razón de transferencia de datos	12.5 MB/s	106 MB/s	528 MB/s
Memoria virtual	sí	sí	sí
Administración de memoria y funciones de protección	sí	sí	sí
Posibilidad de direccionamiento de E/S	64 KB	64 KB	64 KB
Modos de dirección	8	11	11
Cache integrada	no	sí (1)	sí (2)
Registro aritmético	8	8	8
Registro de indexado	4	7	7
Registro general	8	8	8

Tabla 2.1. Comparación entre los microprocesadores 80286, 80486 y Pentium.

Básicamente, con el microprocesador Pentium, la velocidad de procesamiento de las instrucciones fue mejorado dramáticamente, con respecto a sus antecesores, debido al completo rediseño de la unidad de punto flotante, con sus más rápidos y eficientes algoritmos, permitiendo la ejecución de operaciones aritméticas y lógicas hasta diez veces más rápido que en el 80486.

Adicionalmente, se le agregó otra memoria caché (una memoria muy rápida ubicada entre el microprocesador y la memoria principal, que es mucho más lenta, utilizada para aumentar el rendimiento de la computadora), una unidad de pronóstico de bifurcación con su respectiva memoria intermedia para aumentar la velocidad de transferencia de datos entre las aplicaciones más utilizadas y un controlador de *pipelines*.

El *pipelining instruction processing* o direccionamiento tubular, consiste en la lectura, decodificación y ejecución de cada instrucción, superponiéndose estas etapas de procesamiento entre sí, con la finalidad de obtener una mayor velocidad de proceso (en el caso del microprocesador Pentium este proceso se realiza con 5 etapas simultáneas).

Elementalmente, los subsistemas que presenta cualquier microprocesador de utilización general se resumen en tres (la memoria, elemento imprescindible de los microprocesadores, no se incluye entre los subsistemas analizados ya que, en el caso del Pentium, no es accesible sino para los ingenieros de Intel): un subsistema de control que coordina el funcionamiento de todo el sistema y que toma decisiones en función de resultados previos, un subsistema de cálculo para la operaciones de operaciones lógicas y aritméticas y un subsistema de entradas y salidas para establecer comunicación con el mundo exterior. Estos subsistemas se relacionan como se muestra en la figura 2.2:

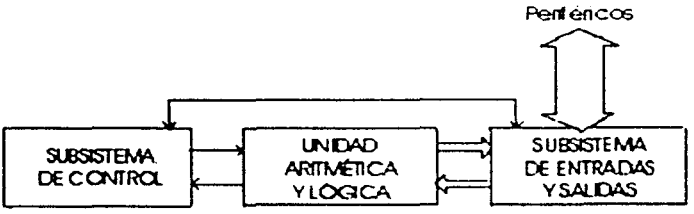


Figura 2.2. Diagrama de bloques de un microprocesador.

En la práctica, para lograr el funcionamiento anterior a las mayores velocidades y con la mejor eficiencia posible, el microprocesador Pentium está compuesto de doce módulos internos:

1) Módulo de control.
2) Módulo de punto flotante.
3) Módulo de codificación.
4) Módulo de interfaz del *bus*.
5) Módulo de segmentación y unidad de paginación.
6) Módulo de generación de direcciones para los pipelines.
7) Módulo de entornos con la unidad aritmética y lógica para los pipelines.
8) Módulo de memoria caché de código y datos.
9) Módulo de memoria interna de cargado de instrucciones con anticipación (*prefetch*).
10) Módulo de memoria interna de escritura.
11) Módulo de memoria interna de *writeback*.
12) Módulo de memoria interna para pronóstico de bifurcación

La relación entre estos módulos se muestra en la figura 2.3.

Una descripción detallada de cada módulo y la operación completa del microprocesador Pentium sobrepasa los objetivos propuestos para este trabajo. Para los fines de este proyecto, lo que es importante analizar de la arquitectura del microprocesador Pentium, es el módulo interno listado con el número 4: el módulo de interfaz del *bus* (BIU por sus siglas en inglés, *Bus Interface Unit*) o, según la nomenclatura de la fig. 2.3, la unidad del *bus*.

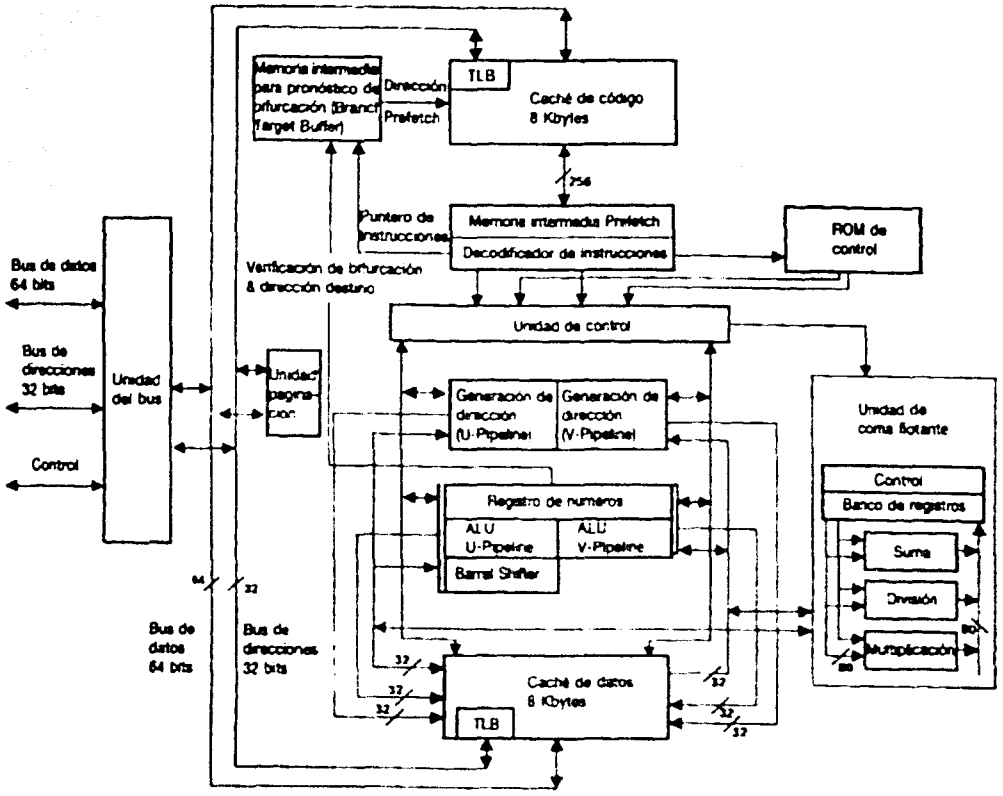


Figura 2.3. Arquitectura del procesador Pentium[2].

[2] Tomado de la pag. 53 de Birmelin Michael, *MANUAL DE LOS PROCESADORES 80xxx Y PENTIUM.*

- *EL MÓDULO DE INTERFAZ DEL BUS*

El módulo de interfaz del *bus* coordina la búsqueda de instrucciones, la transferencia de datos y las funciones de control entre el mundo exterior y los módulos del microprocesador, antes mencionados. Así, el BIU está conformado para cumplir con las siguientes características: recepción del *bus* de direcciones, recepción del *bus* de datos, control de la memoria caché, generación de una señal de paridad para supervisión interna, almacenamiento de solicitudes de escritura (*write-buffering*), control del tamaño del *bus*, control de ciclos del *bus* y control de sistemas de interfaz de datos.

Externamente, el BIU transfiere todas las señales procedentes de los periféricos al *bus* del microprocesador mientras que, internamente, se comunica con los otros módulos del microprocesador por medio de un *bus* de datos de 64 bits y un *bus* de direcciones de 32 bits (en la figura 2.3 no se muestran las líneas de control por simplicidad).

Elementalmente, el proceso en el que interactúa el BIU, junto con los demás módulos del microprocesador, para lograr la interfaz de datos con los periféricos, es el siguiente (ver fig. 2.3):

- Los datos externos acceden a los módulos internos del microprocesador por un *bus* de datos de 64 bits a través del BIU.
- Ya dentro del microprocesador, los datos son transferidos, otra vez por el módulo de interfase del *bus* y por el módulo de memoria caché, al módulo de decodificación.
- En el módulo de decodificación se traducen las instrucciones en microcódigos, siendo éstos ejecutados por el módulo de control.
- Los resultados internos son almacenados en registros internos, dentro del módulo de punto flotante y la caché.

Así, el módulo de interfaz del *bus*, sólo opera en el caso de que la unidad de control, a indicación, por ejemplo, del *software*, requiera el acceso al *bus* del procesador. La familia de microprocesadores Pentium tiene una capacidad de realizar, a una velocidad de 66MHz, transferencias de datos a 528 Mbytes/segundo. Igualmente, puede dirigir de manera directa 4 Gbytes de memoria física, o más, y hasta 64 kbytes de espacio de direccionamiento de entrada/salida (la cual se tratará más detalladamente en la sección 2.1.5).

Como ya se ha mencionado, la creación de la interfaz electrónica desarrollada en este trabajo, se debe a la necesidad de actualización de las consolas de control de los observatorios. Esta actualización implica que los nuevos dispositivos instalados, en particular los nuevos microprocesadores Pentium, operen con aquellos dispositivos que ya se encuentran en los observatorios, especialmente con los controladores de los motores de las monturas de los telescopios y sus sistemas de guiado (incluyéndose aquí tanto el *hardware* como el *software*). Esta capacidad de funcionamiento se denomina compatibilidad.

Debido a la gran cantidad de información que debe comunicarse entre el microprocesador y los componentes adicionales, tanto internos como externos, ocasionada por los complejos programas de aplicación actuales, el ancho del *bus* en las computadoras se ha ido incrementado con el desarrollo de los microprocesadores, como se ve en la tabla 2.2.

Procesador	*Arquitectura interna*	*Bus externo*
80286	16 bits	16 bits
80386	32 bits	32 bits
80486	32 bits	32 bits
Pentium	Arquitectura superescalar 32 bits	64 bits

Tabla 2.2. Incremento de tamaño de los sistemas de *bus*.

La compatibilidad de los microprocesadores Pentium con las arquitecturas anteriores a él radica, entre otros factores, en la organización del sistema de interfaz del BIU implementado por Intel. En particular, debido a que la interfaz electrónica desarrollada en este trabajo opera mediante un sistema de interfaz de entradas/salidas programadas (ver fig. 2.4), vía un *bus* de datos y un *bus* de direcciones de menor tamaño que los *buses* del microprocesador Pentium, el BIU debe garantizar la compatibilidad mediante un método de partición de datos.

La comunicación entre sistemas de *bus* de distinto tamaño se basa en que, si un *bus* de datos de 64 bits sólo encuentra disponible un *bus* de 32 para su comunicación, entonces el *bus* de 64 bits del BIU debe partir primero los datos en dos datos de 32 bits. El mismo caso se da para un *bus* de datos de 32 bits, cuando el procesador accede a tarjetas de expansión o memoria externa con *bus* de datos de 16 bits u 8 bits. Adicionalmente, el ancho de bus puede ser variado de ciclo a ciclo mediante el controlador de tamaño del bus del BIU, pero esta facilidad no se utiliza para la interfaz electrónica. En la fig. 2.5 se muestra el caso de la partición de datos para el *bus* de direcciones y en la fig. 2.6 se muestra el equivalente para el caso del *bus* de datos. En la figura 2.5 se muestran las señales auxiliares que se generan para cada caso de compatibilización: para sistemas de 32 bits se requiere una lógica externa que genere las señales[3] denominadas A2, BE3'# - BE0'#; para memorias de 16 bits se deben generar A2, A1, BHE# y BLE#, para sistemas de 8 bits se deben generar A2, A1 y A0.

[3] Para un análisis exhaustivo de estas señales consultar el Capítulo 19 de *Embedded Pentium Processor Family Developer's Manual* de Intel. Las figuras 2.13 y 2.14 son ilustraciones reproducidas de la pags. 19-321 y 19-322 respectivamente de este manual.

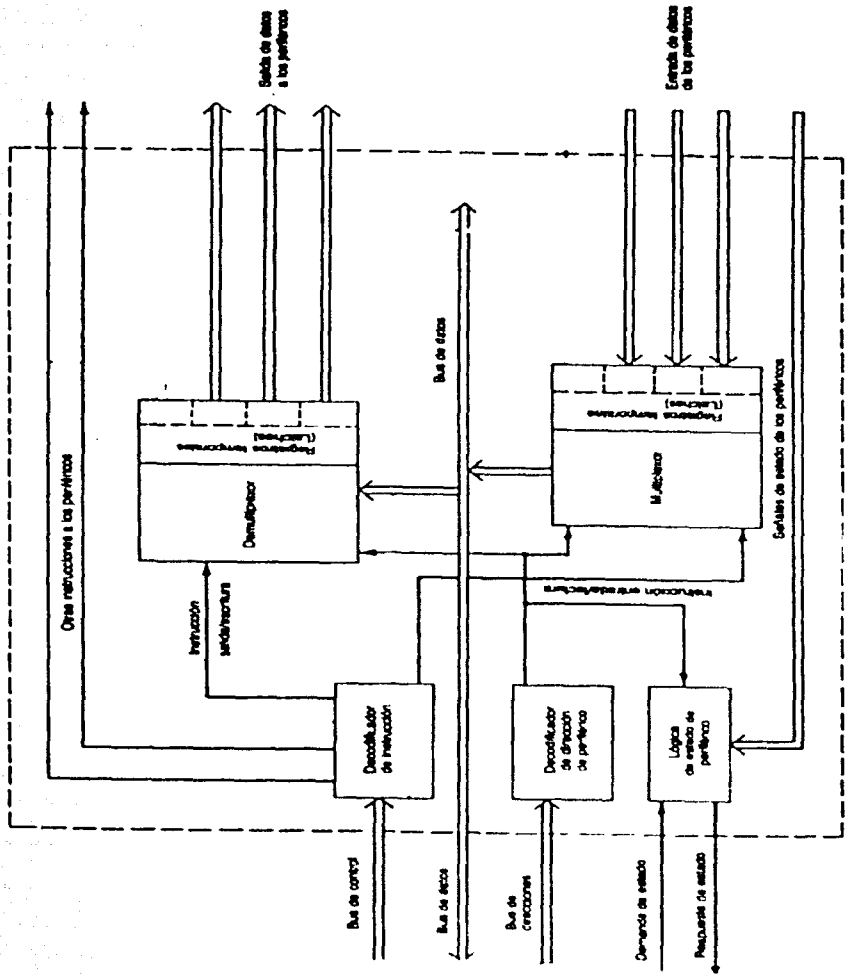


Figura 2.4. Diagrama de bloques de un sistema de interfaz de entrada/salida programada.

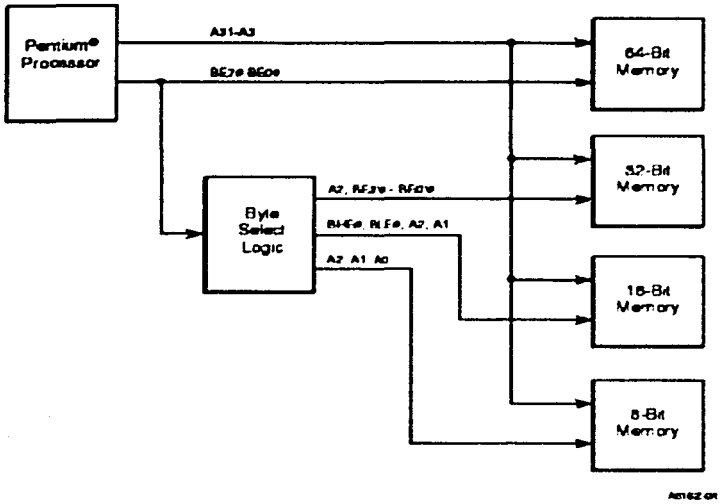


Figura 2.5. Partición de datos en el direccionamiento con memorias de 64, 32, 16 y 8 bits.

Para el usuario o el programador, la electrónica que realiza este proceso aparece para él como una gran caja negra de cual no debe preocuparse ya que ésta realiza todas las conversiones automáticamente. Sin embargo, no sobra señalar que toda aquella comunicación con sistemas que operen con menos de 64 bits (que es el caso de los periféricos con los que interactúa la interfaz electrónica embebida), requieren un proceso extra para dirigir la información de manera correcta y que, sin embargo, debe realizar la interfaz en transferencias unitarias, lo que puede hacer necesaria la implementación de lógica externa para generar señales auxiliares.

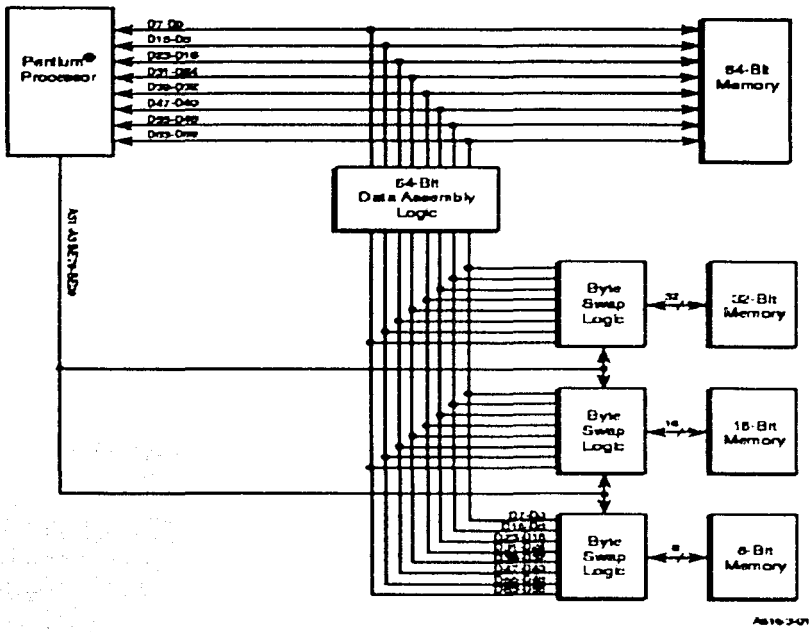


Figura 2.6. Integración/Partición en la interfaz de un *bus* de datos con memorias de 64, 32, 16 y 8 bits.

Así, este procedimiento permite utilizar tanto el *hardware* como el *software* desarrollado para plataformas anteriores con un microprocesador Pentium, lo que implica grandes posibilidades de actualizar sistemas diseñados con microprocesadores anteriores.

Ahora bien, cualquier dispositivo cuyo objetivo sea la transferencia de datos al módulo de interfaz del *bus* del microprocesador Pentium, deberá cumplir ciertas normas de *bus*, ya sea el estándar ISA (*Industrial Standard Architecure*) de 8-16 bits, el EISA (*Extended Industrial Standard Architecture*) de 16-32 bits o, en el caso de aplicaciones

multitareas, el estándar MCA (*Micro Channel Architecture*) o el *bus* PCI (*Peripheral Component Interconnect bus*). La interfaz electrónica embebida desarrollada en este trabajo utiliza el estándar de *bus* ISA.

2.1.2. EL BUS ISA

Toda la información que recibe, procesa y proporciona el microprocesador a otros componentes es digital. Para lograr esta comunicación entre los distintos componentes del sistema, como ya se ha explicado, se utiliza un conjunto de líneas llamado *bus* o ducto que transporta los 1 y los 0 que representan la información.

Un *bus* completo estandarizado, está compuesto por tres ductos internos (ver fig. 2.8):

1) Un *bus* de direcciones.- Son líneas unidireccionales que sirven para enviar desde el microprocesador, vía el módulo de control, la dirección del dispositivo seleccionado para una transferencia de datos. El módulo de control entrada/salida se comunica tanto con los sistemas dentro de la Unidad Central de Proceso (CPU), por ejemplo las memorias, como los que están fuera del CPU, es decir, los periféricos.
2) Un *bus* de datos.- Son líneas bidireccionales para el envío de instrucciones y datos entre el microprocesador, la memoria y los periféricos.
3) Un *bus* de control.- Son líneas bidireccionales que permiten coordinar todo el sistema.

El *bus* ISA cuenta con un *bus* de direcciones de 20 bits y un *bus* de datos de 8 bits. Su principal característica es que permite trabajar con la mayoría de las señales de interrupción del microprocesador (de las cuales se tratará más extensamente en el siguiente apartado) o utilizar los circuitos de Acceso Directo a Memoria (DMA). Esto

supone una gran ventaja sobre la comunicación realizada, por ejemplo, desde el puerto paralelo, que sólo permite controlar una señal de interrupción (IRQ7), con 8 bits de datos de salida y 5 señales de entrada, o desde el puerto serial, que sólo se comunica con dispositivos que trabajen con el estándar RS-232.

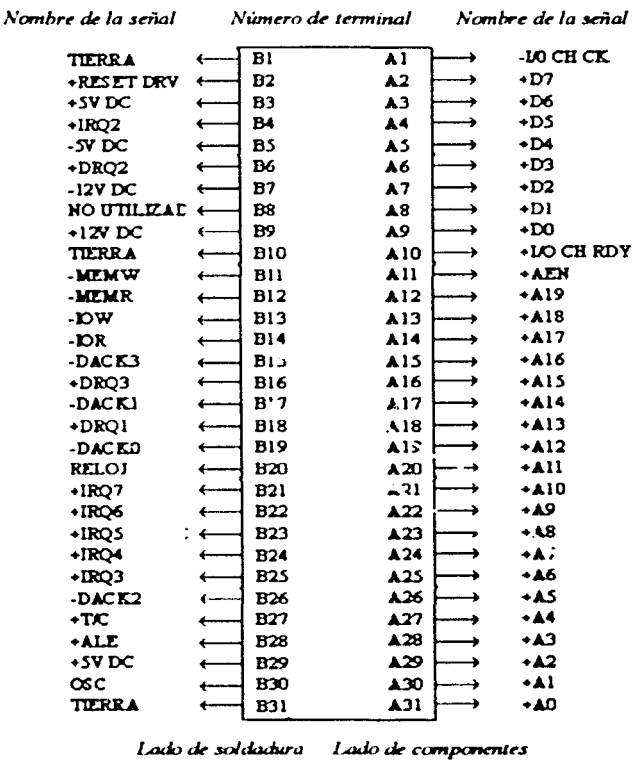


Figura 2.7. Señales del *bus* ISA.

El *bus* de control del *bus* ISA está compuesto de diversas señales, las cuales se encuentran distribuidas, junto con el *bus* de datos y el de direcciones, como se desarrolla a continuación:

Físicamente el *bus* está dividido en dos caras (ver fig. 2.7). En la primera cara las terminales se denominan desde A1 hasta A31 y es la cara de los componentes. En esta cara se encuentran:

- El *bus* de datos, terminales D9 a D7.
- El *bus* de direcciones, terminales A0 a A19.
- El habilitador de memoria o AEN (*Address Enable*), terminal A11. Es la señal de salida, activada por un nivel alto, utilizada por el controlador DMA para indicar que tomará el control de los *buses* de datos y direcciones.
- El verificador de canal de entrada/salida o I/O CH CK (*Input/Output Channel Check*), terminal A1. Es una señal de entrada, activada por un nivel bajo, utilizada para reportar condiciones de error en las tarjetas de interfaz adheridas al *bus*. Cuando esta señal está en bajo, genera una interrupción no enmascarable al microprocesador.
- El preparador de canal de entrada salida o I/O CH RDY (*Input Output Channel Ready*), terminal A10. Es una señal de entrada utilizada para extender la duración de los ciclos del *bus* para que, la memoria o los puertos de entrada/salida que no sean suficientemente rápidos para responder a la velocidad normal del ciclo del *bus*, que es de cuatro ciclos de reloj (840 nseg), se puedan adherir al sistema del *bus*.

La segunda cara del *bus* contiene las terminales de alimentación, así como las señales relacionadas con las interrupciones y las transferencias de datos vía DMA.

Primeramente, están las señales que son activadas por el módulo de interfaz del *bus*, BIU, del microprocesador. Estas son:

- Escritura a memoria o MEMW (*Memory Write*), terminal B11: Esta señal, prendida por un nivel bajo, la activa el BIU del microprocesador para poder escribir datos, desde el *bus* del sistema, en la memoria. Sirve para indicar que el *bus* de direcciones contiene una dirección de una localidad de memoria, a la cual la información contenida en el *bus* de datos será escrita.
- Lectura de memoria o MEMR (*Memory Read*), terminal B12: Esta señal, prendida por un nivel bajo, la activa el BIU del microprocesador para poder leer datos desde la memoria. Sirve para indicar que el *bus* de direcciones contiene una dirección de una localidad de memoria, de la cual la información debe ser leída.
- Escritura de entrada/salida o IOW (*Input Output Write*), terminal B13: Esta señal la activa el BIU del microprocesador para poder escribir en un puerto. Sirve para indicar a los puertos de entrada/salida que el ciclo iniciado por el BIU es de escritura y que la dirección colocada en el *bus* de direcciones es una dirección de puerto de entrada/salida.
- Lectura de entrada/salida o IOR (*Input Output Read*), terminal B14: Esta señal la activa el BIU del microprocesador para poder leer de un puerto. Sirve para indicar a los puertos de entrada/salida que el ciclo iniciado por el BIU es de lectura y que la dirección colocada en el *bus* de direcciones es una dirección de puerto de entrada/salida.
- El habilitador del *latch* de las direcciones o ALE (Address Latch Enable), terminal B28. La señal de salida ALE la activa el BIU del microprocesador para indicar que la dirección colocada en el *bus* de direcciones es la válida para comenzar un ciclo de comunicación con el *bus*. Esta señal se activa en alto justo antes de que se valide el *bus* de direcciones y cae a bajo justo

después de que se valida el *bus* de direcciones. Igualmente, esta señal también puede ser utilizada por el microprocesador para retener los 16 bits menos significativos del bus de datos en un latch, durante un ciclo de lectura/escritura en la memoria o en un puerto.

Están también las señales que el controlador de DMA activa. El DMA (*Direct Memory Access*) se utiliza sólo en el caso de que algún periférico envíe datos a la memoria mucho más rápido que lo que el microprocesador puede manejar. Su controlador es un circuito integrado dedicado que, cuando está activado, controla el bus de direcciones, el bus de datos, la memoria y las líneas de comando de entrada/salida y lectura/escritura capaz de enviar y recibir datos más rápido que el microprocesador, para que, posteriormente, otros dispositivos lo utilicen para acceder a la memoria del sistema. Las señales que el DMA activa son:

- Reconocedores del DMA del DACK0 al DACK3 (*DMA Acknowledge*), terminales B15, B17, B19 y B26. Estas cuatro señales de salida, prendidas por nivel bajo, las activa el controlador de DMA para hacer saber a un dispositivo que el controlador de DMA tiene el control de los *buses*
- Cuenta final o T/C (*Terminal Count*), terminal B27. Esta señal de salida, prendida por nivel alto, es activada por el controlador de DMA cuando ha terminado un número programado de ciclos, para hacer saber a un periférico que el número programado de bytes ha sido enviado.
- Solicitudes de interrupción del IRQ2 al IRQ7 (*Interrupt Request*), terminales B4, B21, B22, B23, B24 y B25. Estas seis señales de entrada son señales que los periféricos activan para reclamar la atención del microprocesador. De estas solicitudes de interrupciones IRQ2 tiene la prioridad más alta mientras que IRQ7 tiene la prioridad más baja.

- Solicitudes para DMA del DRQ1 al DRQ3 (*DMA Request*), terminales B6, B16 y B18. Estas tres terminales de entrada, activadas en alto, son utilizadas por la interfaz para permitir a un periférico el uso de los *buses*.

Finalmente están las siguientes señales:

- Oscilador u OSC, terminal B30. Esta señal de salida es la señal de frecuencia más alta en el *bus* y es de 14.31818 MHz. Su principal finalidad es generar la frecuencia de reloj del sistema u otras señales periódicas, pero también puede ser utilizada íntegramente por las tarjetas de entrada/salida. Esta señal es simétrica, es decir, su ciclo de trabajo es aproximadamente de 50%, y tiene un período de aproximadamente 70 nseg.
- Reloj o *Clock*, terminal B20. Esta señal de salida es el reloj del sistema. Esta señal se genera dividiendo entre tres la señal OSC, lo que da una frecuencia de 4.77 MHz. Esta señal tiene un período de 210 nseg con un tiempo en alto de 70 nseg y un tiempo en bajo de 140 nseg. Por lo tanto, esta señal de reloj no es simétrica, sino que tiene un ciclo de 1/3 ó 2/3.
- Reset de los dispositivos o RESET DRIVER, terminal B2. Esta señal de salida se activa en alto durante las secuencias de encendido de la computadora. Su función, al permanecer activa mientras que todos los niveles de la computadora han alcanzado sus rangos de operación específicos, sirve para proveer una señal de reset a todos los dispositivos adheridos al *bus*, para tenerlos en un estado lógico conocido antes de que el sistema comience a operar. Una vez establecidos los niveles, la señal se desactiva.

- *PROTOCOLO DE COMUNICACIÓN*

La comunicación entre el microprocesador y un periférico, unidos mediante el *bus* ISA, sigue un orden de instrucciones, o protocolo, establecido. Así, un ciclo de lectura, desde un periférico conectado a un puerto de la interfaz electrónica, al microprocesador, por medio del protocolo del *bus* ISA, consta de siete pasos (ver fig. 2.8)[4]:

1) El microprocesador, vía el BIU, pone la señal ALE en un nivel alto.
2) El microprocesador envía la dirección del puerto a leer a través de las señales A0 a A19.
3) Al quedar la dirección retenida en un *latch*, el microprocesador pone a la señal ALE en bajo y en adelante la dirección del puerto a ser leído queda retenida en un *latch*.
4) El microprocesador pone la señal IOR en nivel bajo para poder leer del puerto seleccionado.
5) El dispositivo que ha sido seleccionado mediante la dirección, envía un byte de datos a través de las líneas D0 a D7 del *bus* de datos.
6) El microprocesador lee el *bus* de datos.
7) Después de un cierto tiempo suficiente para que se haya terminado de leer los datos, el microprocesador pone la señal IOR en nivel alto de nuevo.

[4] Las ilustraciones de las figuras 2.8, 2.9 y 2.10 son reproducción de las pags. 45-47 de Eggebrecht Lewis C., *INTERFACING TO THE IBM PERSONAL COMPUTER*, Howard W. Sams & Co., E.E.U.U, 1987

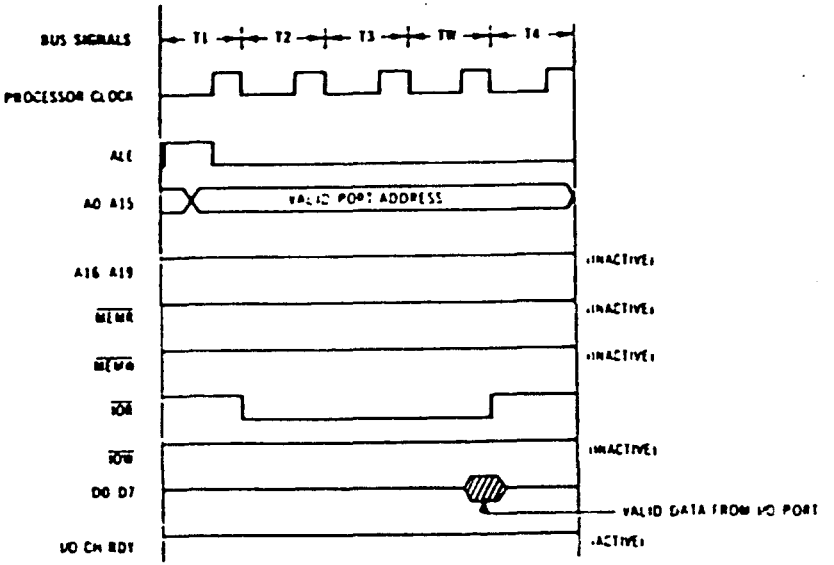


Figura 2.8. Ciclo de lectura a puerto

Un ciclo de escritura, desde el microprocesador a un periférico conectado a un puerto de la interfaz electrónica, por medio del protocolo del *bus* ISA (ver fig. 2.9), es el siguiente:

1) El microprocesador, vía el BIU, pone la señal ALE en un nivel alto.
2) El microprocesador envía la dirección del puerto al cual escribirá a través de las señales A0 a A19.
3) Terminado el envío de la dirección, el microprocesador pone la señal ALE en nivel bajo.
4) El microprocesador pone la señal IOW en nivel bajo para poder escribir al puerto seleccionado
5) El microprocesador envía el byte de datos que será escrito.

6) Después de un tiempo suficiente para que el periférico haya leído los datos enviados, el microprocesador pone la señal IOW en nivel alto de nuevo.

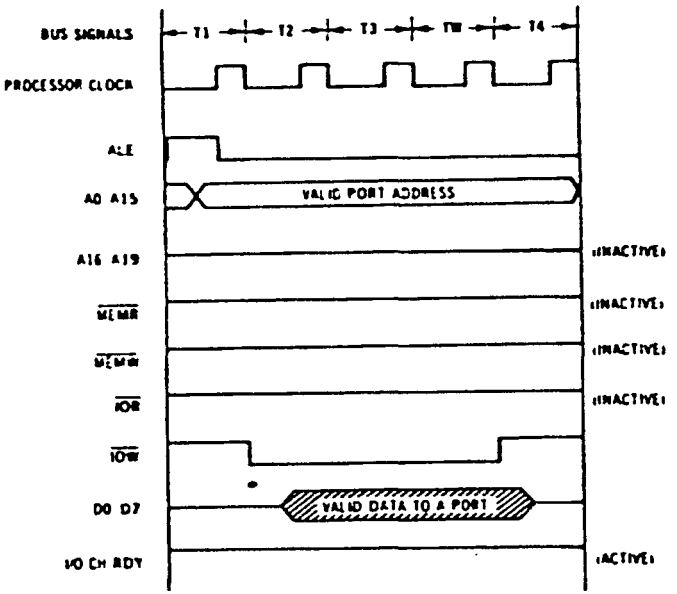


Figura 2.9. Ciclo de escritura a puerto.

El ciclo de lectura o escritura a memoria es casi idéntico de los ya descritos de lectura o escritura a puertos. La única diferencia es que, para un ciclo de lectura o escritura a memoria, se utilizan las señales MEMR en vez de IOR y MEMW en vez de IOR (ver figuras 2.10 y 2.11).

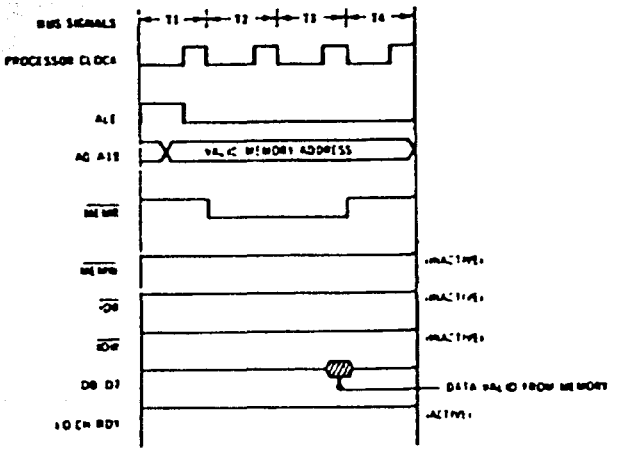


Figura 2.10. Ciclo de lectura a memoria.

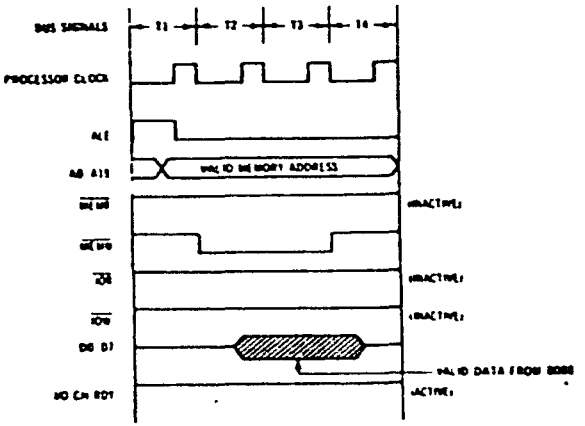


Figura 2.11. Ciclo de escritura a memoria.

Los ciclos de lectura o escritura, tanto a puertos como a memoria, son llevados a cabo gracias a subrutinas de *software* que controlan al microprocesador de la computadora, específicamente diseñadas para lograr tal comunicación. Las subrutinas para la transferencia de datos entre los periféricos y el telescopio con montura ecuatorial, siguiendo los pasos explicados en esta sección, están desarrolladas dentro del programa de guiado y de control de la consola del Observatorio, el cual se analizará en la sección 2.1.4.

2.1.3. LAS INTERRUPCIONES

Para la transferencia de datos entre el microprocesador y sus periféricos se utiliza un mecanismo basado en un sistema de interrupciones. Existen dos tipos de interrupciones: las interrupciones por *software* y las interrupciones por *hardware*. Las interrupciones por *software* son las transferencias de datos en paralelo más ampliamente utilizadas en las computadoras y están directamente relacionadas con las transferencias de entrada/salida programadas; siendo éstas, aquellas instrucciones que son ejecutadas por un programa, o una subrutina, para que el microprocesador sea el que inicie siempre la comunicación hacia los periféricos. Por otro lado, las interrupciones por *hardware* se dan a solicitud expresa de cada periférico hacia el microprocesador, en cualquier momento, para iniciar una transferencia de datos.

Las solicitudes de interrupción por *hardware* se realizan en aquellos casos en que es más eficiente que el periférico avise al microprocesador que tiene información para transmitirle, en vez de que el programa de la computadora pierda tiempo preguntándole si tiene o no información disponible. Sin embargo, aunque la transferencia se inicia con la solicitud del periférico, es el microprocesador el que realiza la transferencia en paralelo, siempre bajo el control del programa. Igualmente, si se reciben varias solicitudes simultáneamente, existen interrupciones con mayor prioridad, además de que el programa

puede prohibir en determinados casos las solicitudes. Como se vio anteriormente, en el bus ISA las solicitudes de interrupción se ejecutan mediante las señales IRQ, activadas por los periféricos, para solicitar una transferencia de datos al microprocesador.

Cuando ocurre cualquiera de los dos tipos de interrupciones, el programa que está ejecutando el microprocesador es detenido y ejecuta una subrutina contenida en la memoria. Como los microprocesadores Pentium soportan diferentes tipos de interrupciones, cada tipo de interrupción con un nivel de prioridad diferente, se les asignan vectores que apuntan a la direcciones de memoria donde se encuentran las subrutinas correspondientes. El segmento de memoria de la computadora donde se localizan estos vectores, también llamada Tabla de Vectores de Interrupción, se ubica en los primeros 1024 bytes de memoria, ocupando las localidades de memoria desde la $000000_{hexadecimal}$ hasta la $0003FF_{hex}$.

Debido a que el programa de control con el que interactúa la interfaz electrónica desarrollada en este trabajo utiliza las interrupciones por *software* para adquirir o enviar los datos provenientes del controlador de movimientos del telescopio, el reloj astronómico, el controlador de motores y los sensores ambientales, el desarrollo a continuación se enfocará a ese tipo de interrupciones.

Las interrupciones por *software* son útiles en aquellos casos en que se deben realizar determinadas tareas en un intervalo regular, sin importar qué otros eventos estén ocurriendo en la computadora. Éstas se generan cuando la instrucción INT es activada. Esta señal se programa para que sea activada cada determinado número de pulsos de sincronía o "ticks" de temporización. Para explicar la lógica detrás de este procedimiento se debe tener en cuanta que la computadora tiene dos tipos de temporizadores o *timers*: el temporizador del sistema y el reloj de tiempo real.

El temporizador del sistema es un contador que nace de la necesidad de la memoria RAM dinámica de estarse recargando cada intervalo de tiempo regular (*refresh frequency*) ya que, por lo general, los dispositivos de memoria dinámica requieren que cada una de las localizaciones de memoria primarias sean leídas por lo menos una vez cada 2 milisegundos para no perder la información allí almacenada. Adicionalmente, este temporizador se utiliza para que, cada vez que se recargue el sistema operativo BIOS de la computadora, el reloj de tiempo real obtenga de él la información necesaria para desplegar el tiempo en horas, minutos y segundos, tal como estamos familiarizados.

La frecuencia utilizada para mantener la hora y el calendario en tiempo real de la computadora se obtiene a partir del temporizador del sistema al ingresar la señal, originalmente proveniente del cristal oscilador de la computadora, a un contador descendente, el cual genera "ticks" de temporización cada vez que el contador alcanza el valor de cero. Cada vez que se inicia la computadora, el BIOS es el encargado de programar al contador y , por ende, al temporizador, para que oscile a una frecuencia de 18.206 "ticks" cada segundo, ya que cada vez que el contador llega a cero se genera un "tick". Esta frecuencia se obtiene de la división de la frecuencia del circuito de temporización de la computadora de 1.19318 MHz entre 65,536 kHz, que es la frecuencia que se obtiene al decrementar la cuenta del temporizador 2 veces entre cada "tick", con contadores internos de 16 bits, o sea 2^{16}. La frecuencia del circuito de temporización de 1.19318 MHz es un submúltiplo de la frecuencia básica del cristal oscilador de 4,772,727 Hz (la razón de esta frecuencia básica se remonta a la frecuencia básica utilizada comúnmente en la electrónica de las televisiones, electrónica que se utilizó originalmente al implementar las primeras computadoras personales).

Cada vez que se genera un "tick", éste se utiliza para generar una interrupción; la 0008_{hex}, que a la vez genera la señal denominada INT. De esta manera, al activarse la señal INT, se ejecuta una subrutina contenida en memoria llamada Controlador de

Interrupciones que a su vez llama a la interrupción $001C_{hex}$. Hay que notar que la interrupción 0008_{hex}, generada en cada "tick", hace que el controlador de interrupciones provoque la señal IRQ0 (Interrupt Request 0). Así, a los "ticks" se les asigna la interrupción de *hardware* con la más alta prioridad (con excepción de las interrupciones no enmascarables que no pueden deshabilitarse), por lo que el código correspondiente a los manejadores de las interrupciones 0008_{hex} y $001C_{hex}$ toman precedencia sobre cualquier otro sistema del *software*. Este desenlace es muy importante, ya que un programa puede detectar cuándo ocurre un "tick" utilizando la interrupción $001C_{hex}$.

La serie de eventos ocasionada por la intervención del controlador de interrupciones se inicia cuando el microprocesador transfiere los contenidos de los registros y punteros que estaba utilizando el programa interrumpido a una pila o *stack*, para que el controlador de las interrupciones pueda reiniciar estos valores según lo demanden las subrutinas de las interrupciones, y así no perder los valores originales de éstos, para que puedan ser reutilizados posteriormente. Estos registros son: los registros de las banderas, el registro de código (CS por sus iniciales en inglés: *Code Segment*), y el puntero de las instrucciones (IP: *Instruction Pointer*)

Posteriormente, el controlador de interrupciones cede el control del programa a las subrutinas programadas que se realizarán durante el interrupción. Es en estas subrutinas que se realizan las transmisiones de datos de entrada/salida con los periféricos. Al terminar, el controlador de interrupciones envía una instrucción de retorno de la interrupción IRET (*Interrupt Return*) que hace que lo contenido en la pila o stack (las banderas, el CS y el IP del programa interrumpido), regresen a sus registros, transfiriendo de esta manera el control al programa de control.

Así, una vez que el controlador de interrupciones ha realizado su tarea, el microprocesador continúa el procesamiento del programa en el punto en el que la interrupción ocurrió.

2.1.4. TRANSFERENCIAS PROGRAMADAS INCONDICIONALES

Es necesario mencionar que la comunicación por medio de las transferencias de datos programadas pueden ser, a su vez, de dos tipos: las condicionales o asíncronas y las incondicionales o síncronas.

En las transferencias condicionales de datos se verifica primero el estado en que se encuentra el periférico con el que se va a comunicar, para comprobar si se encuentra listo o no para la transferencia de datos. Si el periférico le responde al microprocesador que no se encuentra habilitado para la transmisión, el programa encargado de la comunicación comienza un ciclo en el que repite la pregunta hasta que el periférico le conteste afirmativamente. Cuando ha sucedido esto, el microprocesador inicia la transmisión, ya sea de entrada o de salida, de la información.

En las transferencias incondicionales de datos no se verifica el estado en que se encuentran los periféricos, por lo que éstos deben estar siempre listos. Este tipo de transferencia se utiliza cuando se trabaja con periféricos dedicados, es decir, cuando la señal proveniente del periférico, o la falta de señal, aportan siempre datos al programa de control.

De aquí que se utilicen las transferencias programadas incondicionales para comunicar a la computadora, vía la interfaz electrónica, con el controlador de movimientos del telescopio, con el controlador de motores *Galil*, con el reloj astronómico y con los sensores ambientales, ya que estos dispositivos se encuentran siempre listos y

su funcionamiento no tiene variaciones imprevisibles en sus tiempos (es decir, estos dispositivos son capaces de cumplir con la transferencia de información requerida dentro del tiempo de instrucción del microprocesador).

- *EL PROGRAMA DE CONTROL PARA LA CONSOLA*

Como se vio en la descripción del *bus* ISA, es necesario una secuencia de instrucciones y eventos denominado protocolo para lograr la correcta transferencia de datos entre el microprocesador y los periféricos. Estas secuencias de instrucciones están contenidas en aquellas subrutinas que son llamadas al acontecer las interrupciones antes mencionadas.

El programa de control que contiene éstas subrutinas se basa en una filosofía multitareas, con la cual el microprocesador realiza las transferencias, ya sean de entrada o de salida, cumpliendo con el protocolo establecido para el *bus* ISA, visto en la sección anterior, y con las capacidades del BIU de la computadora sincronizadas en el programa con la interrupción $001C_{hex}$. Este programa fue desarrollado en el IAUNAM y es utilizado en las consolas de algunos telescopios de los Observatorios Astronómicos Nacionales y en observatorios de otras universidades[5].

Este programa, además de contar con las funciones básicas de bases de datos y mandos a distancia o comunicación remota, está especializado en el control de movimientos y guiado de la montura de un telescopio, en el correcto despliegue de la información de los movimientos y guiado del telescopio, y sus accesorios, y en la coordinación de movimientos con la cúpula del observatorio.

[5] Para un estudio completo de este programa de control y guiado, consultar: Bernal A. y Gutiérrez L., *Sistema Computarizado de Control del Telescopio de Tonantzintla. Los Programas*, Reporte Técnico RT-95-01, Instituto de Astronomía de la UNAM.

Para los fines de este trabajo, sólo es necesario analizar, del programa de control y de guiado, o mejor dicho, de los distintos programas que interactúan en común para conseguir el control y el guiado del telescopio, las subrutinas que controlan las transferencias de datos. El diagrama de flujo elemental del programa de control que opera en la computadora de la consola de control del Observatorio se muestra en la fig. 2.12.

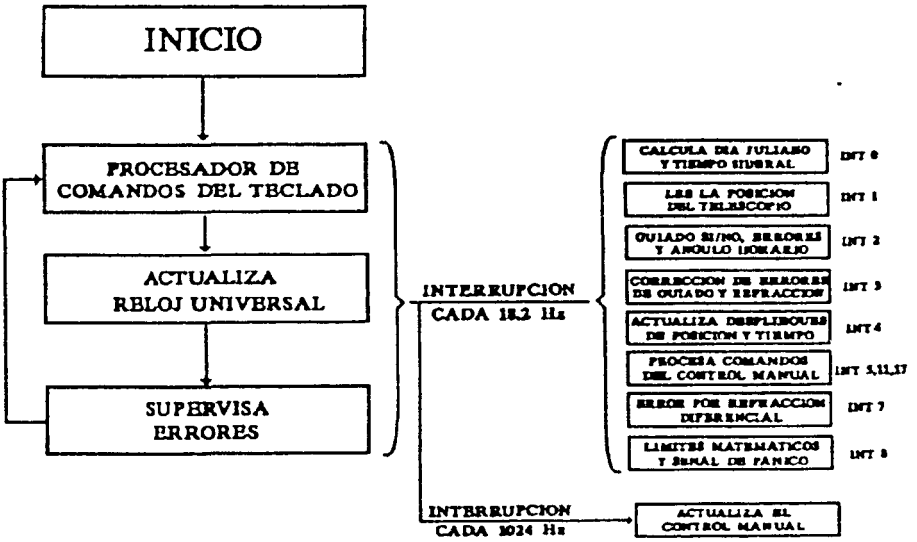


Figura 2.12. Diagrama de flujo del programa principal de la consola de guiado.*

* Esquema reproducido de la pag. 7 de Bernal A. y Gutiérrez L., *Sistema Computarizado de Control del Telescopio de Tonantzintla. Los Programas*, Op. Cit.

La subrutina de interrupción, cada vez que es llamada (18.206 veces cada segundo), sincronizada con la interrupción $001C_{hex}$, reparte varias tareas realizando un control estructurado tanto de las interrupciones como del número de "ticks" transcurridos. Para lograr la organización de la fig. 2.12, esta subrutina contiene una estructura de control que permite 18 casos mediante la instrucción *case*. Los casos que interesan para la interfaz electrónica son:

Caso 2: Lee el puerto 0308_{hex} para obtener el número de pulsos provenientes del reloj sideral, habilitando el guiado mediante la escritura en el puerto 0308_{hex}

Caso 5: Lee el puerto 0309_{hex} y el $030A_{hex}$ para muestrear los pulsos provenientes del controlador manual para mover el telescopio.

Caso 8: Lee el puerto $030B_{hex}$ para revisar desde el controlador de motores *Galil* la terminación de los movimientos.

Caso 9: Escribe al puerto 0309_{hex} para seleccionar el canal del Convertidor Analógico/Digital y lee el puerto $030C_{hex}$ para obtener la información de los sensores de humedad y de temperatura.

Caso 11: Igual que en el caso 5.

Caso 17: Igual que en el caso 5.

Como se ve, el control manual del telescopio por medio del controlador manual se verifica en los casos 5, 11 y 17, lo que se traduce en que los pulsos provenientes de éste (para ambos ejes del telescopio), se leen a una frecuencia aproximada de 100 veces cada segundo.

El lenguaje de programación en el que está desarrollado el programa de la consola y, por lo tanto, el control de las interrupciones y las transferencias programadas incondicionales a utilizar, por medio de la interfaz electrónica embebida, es C, compatible con el compilador Turbo C++ de Borland y con el sistema operativo MS-DOS.

En su forma elemental, una subrutina para la obtención de datos desde un periférico hacia el microprocesador, por medio de una transferencia programada de datos, es la siguiente (los comentarios o explicaciones se incluyen después del símbolo //, siguiendo las norma de programación en el lenguaje C):

a) Primeramente se establece una función llamada lee_canal en el programa general para que sea utilizada en todas las lecturas a puertos por donde ingresen los datos desde los periféricos.

```
char lee_canal (int canal)
        {
        unsigned char lectura,
        int i;
                outport(0x309, canal),          // pon address de canal
                outport(0x309,8 | canal);       // Address Latch Enable (ALE)
                outport(0x309,24),              // inicia conversión
                outport(0x309,16),              // apaga ALE
                outport(0x309,0),
                for (i=0, i<25000, i++)         // espera a que esté lista la
                        {,}                     // conversión
                lectura=inport(0x30C),
        return lectura,
        }
```

b) Posteriormente sólo es necesario crear una función para cada periférico deseado de la manera que sigue (en este ejemplo, el dato a adquirir es el proveniente del sensor de temperatura):

```
void Temperatura (void)
        {
```

```
unsigned char dato_digital,
double voltaje_del_sensor_temperatura, tem peratura,temp;
char buf_temperatura[6].

        dato digital=lee_canal(0);                    // lee el puerto

voltaje_del_sensor_de_temperatura=(factor_de_conversion*dato_digital);
temperatura=(voltaje_del_sensor_de_temperatura*10).
gcvt(Temperatura,3,buf_hume).
mdf(humedad,&temp).
}
```

Cabe mencionar que, en estas subrutinas de ejemplo, no se han incluido las instrucciones para el despliegue en pantalla, con el fin de simplificar la subrutina a su mínima expresión.

De la misma manera, en su forma elemental, una subrutina para enviar datos desde el microprocesador hacia un periférico por medio de una transferencia programada de datos es la siguiente (en este ejemplo, el dato a enviar es el que habilita o deshabilita el guiado del telescopio):

```
void guiado(void)
        {
        if(guiar && ¡limite[0])          // verifica que el guiado esté dentro de los
                {                        //         límites del telescopio
                while(!guiando)          // espera a que el caso 2 de la int8
                        {;}              //         sincronice el guiado
                oprime(2).
                }
        else
                {
                while(guiando)                  // espera el caso 2 de la int8
                        {.}
                botones[2].posicion=1.
                suelta(2).
                p_308&=0xfe.
                outportb(0x308,p_308);          // escribe al puerto
                }
        }
```

2.1.5. *DECODIFICACION DE PUERTOS*

El análisis del diseño del decodificador de dirección de puertos de entrada/salida es igualmente importante ya que, mediante la implementación del diseño expuesto en este apartado, la tarjeta de adquisición electrónica desarrollada en este trabajo consigue una correcta transferencia de direcciones de datos.

Un microprocesador de la familia Pentium es capaz de operar un Espacio de Memoria Física (*Physical Memory Space*) dirigible de 4 GB o más, como se muestra en la fig. 2.13. A nivel *hardware*, el espacio de memoria está organizado como una secuencia de bloques de 64 bits (referido como *64-bit Wide Memory Organization* en la fig. 2.13); cada bloque de 64 bits utiliza ocho bytes consecutivos en la memoria de direcciones que sirven para dirigir el bloque.

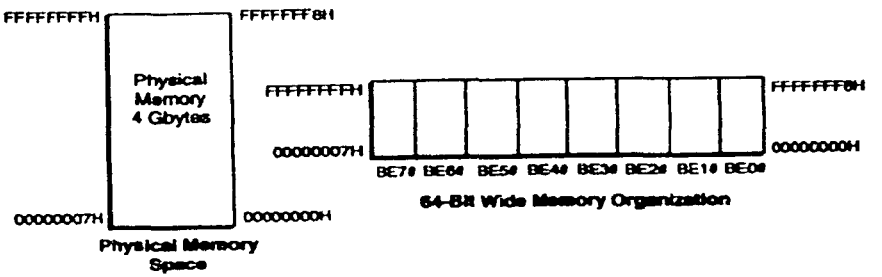


Figura 2.13. Organización de la memoria física.[7]

Adicionalmente, el microprocesador Pentium tiene 64 KB de espacio de direccionamiento para puertos entrada/salida, que es distinta y está separada de la memoria física. A nivel *hardware*, el espacio de direccionamiento de entrada/salida está

[7] Las figuras 2.13 y 2.14 son ilustraciones reproducidas de la pags. 19-321 y 19-322 respectivamente de *Embedded Pentium Processor Family Developer's Manual* de Intel.

estructurado como una secuencia de bloques de 32 bits. Cada bloque de 32 bits de espacio de direccionamiento de entrada/salida utiliza 4 bytes consecutivos en la memoria de direcciones para dirigir el bloque; estos bloques se numeran del 0000_{hex} al $FFFF_{hex}$ (ver fig. 2.14).

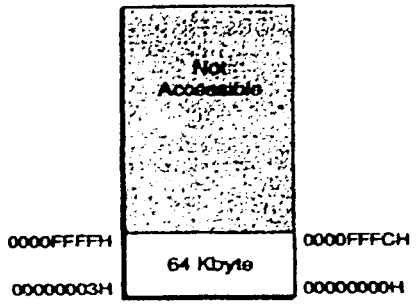


Figura 2.14. Espacio de direccionamiento de entrada/salida.

Como se ve en la fig. 2.14, existen direcciones inutilizables en este espacio. Estas son las direcciones para el puerto entrada/salida que van de la $00F8_{hex}$ a la $00FF_{hex}$, que se encuentran reservadas.

El microprocesador puede transferir 8, 16 ó 32 bits hacia o desde cualquier espacio de direccionamiento de entrada/salida, ya que cualquier par de puertos consecutivos de 8 bits puede ser tratado como un puerto de 16 bits y cualesquiera cuatro puertos consecutivos de 8 bits pueden ser tratados como un puerto de 32 bits.

A nivel *hardware*, el direccionamiento de entrada/salida se realiza a través de las líneas de dirección del microprocesador. Dentro del microprocesador Pentium, un *bus* interno indica si las líneas de dirección están siendo manejadas por una dirección de

memoria física o una de direccionamiento de entrada/salida. Cuando se selecciona esta última, el *hardware* debe decodificar automáticamente a los puertos de entrada/salida en vez de a la memoria física.

Las líneas de dirección (de A31 a A0) están divididas en dos partes: la de los bits más significativos (A31 a A3) y la de los bits menos significativos (A2 a A0). Esta última parte permite que se pueda dirigir un *bus* de $2^3 = 8$ bytes de datos. El bit de habilitación AEN (Address Enable) se utiliza para indicar si el *bus* de datos a dirigir será de entrada (para los ciclos de lectura) o de salida (para los ciclos de escritura).

El circuito más sencillo que realiza la decodificación de la dirección de un puerto es aquel que ha sido diseñado específicamente para decodificar unas direcciones preestablecidas y fijas. Esto se logra mediante la selección de un espacio vacío para las direcciones en el mapa de memoria de la computadora y, a partir de él, construir mediante demultiplexores y compuertas lógicas, un arreglo que decodifica las líneas, dependiendo de las condiciones de los valores de las entradas y de los habilitadores (*enables*) de estos circuitos integrados.

En la fig. 2.15 se muestra el mapa de memoria detallado del espacio de direccionamiento de entrada/salida. En el caso de los periféricos con los que opera la interfaz electrónica embebida desarrollada en este trabajo, se utilizan las direcciones 0308_{hex} a la $030B_{hex}$ para dirigir los puertos de entrada y las direcciones 0308_{hex} y la 0309_{hex} para dirigir los puertos de salida. Estas direcciones se eligieron ya que éstos espacios se encuentran sin utilizar en el mapa de memoria y además no interfieren con los demás periféricos con los que interactúa el telescopio.

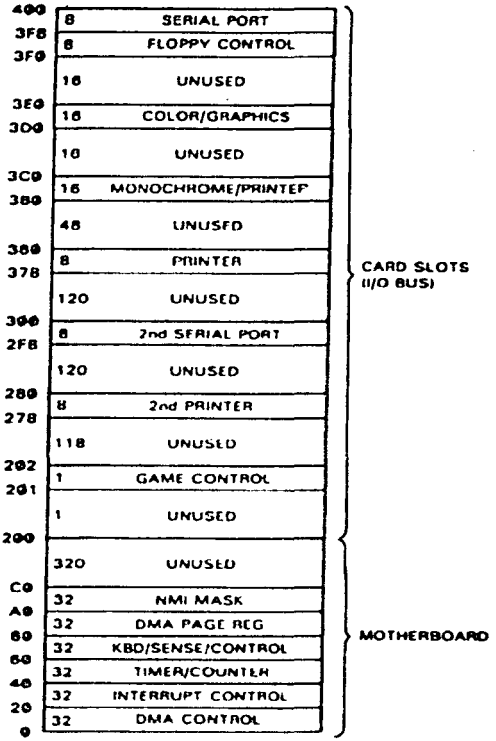


Figura 2.15. Utilización del espacio para direcciones de entrada/salida.[8]

El inconveniente de implementar un diseño de decodificación fija es que las direcciones que decodifica pueden solaparse con las de otras tarjetas implementadas en el sistema o con futuras tarjetas que pudieran utilizarse en el sistema. De aquí la necesidad

[8] La ilustración es una reproducción de la pag. 687 de Horowitz Paul, Hill Winfield, *THE ART OF ELECTRONICS*, Cambridge University Press, E.E.U.U., 1989.

de recurrir a un circuito de decodificación que permita mover el espacio de las direcciones, mediante la manipulación sencilla de un banco de *dip switches*.

En la fig. 2.16 se muestra el circuito básico con el que se consigue un decodificador de puerto entrada/salida cuya dirección a decodificar es seleccionada por interruptores. Utilizando un comparador (el SN74LS688 en este caso, que es un comparador octal), se tiene que los bits de la dirección del *bus* A3 al A9 y la señal AEN son comparadas con la salida de las tensiones que se tienen según se seleccione las posiciones de los interruptores en el *dip switch*. Gracias a un banco de resistencias conectado en un arreglo tal que, cuando uno de los interruptores está abierto (apagado), se compara con un nivel alto a la señal asociada a la dirección del *bus*. Igualmente, un decodificador de dirección de memoria física opera de manera equivalente a la antes expuesta, salvo que la señal AEN no se utiliza para las decodificaciones y que se utilizan bits extras de mayor orden, ya que se necesita dirigir una mayor cantidad de direcciones. Entre los decodificadores / demultiplexores más comúnmente utilizados para este diseño están los SN74LS138 y el SN74LS139, aunque también se puede utilizar un bus controlador (*bus transceiver*) para habilitar o deshabilitar la decodificación.

La lógica detrás de este diseño es el siguiente: cuando el valor fijado en el *dip switch* equivale al valor de la dirección del *bus*, la salida del comparador es activada a nivel bajo y se utiliza como señal de control que elige entre el proceso de lectura o escritura a puerto. Esto se consigue ingresando la señal de salida del comparador en el habilitador de los demultiplexores SN74LS138 para que, cuando coincida la señal baja proveniente del comparador, junto con la señal IOR ó IOW, ambas negadas, opere el demultiplexor para que A0, A1 y A2 se traduzcan en los registros de lectura de entrada/salida o en los registros de escritura de entrada/salida.

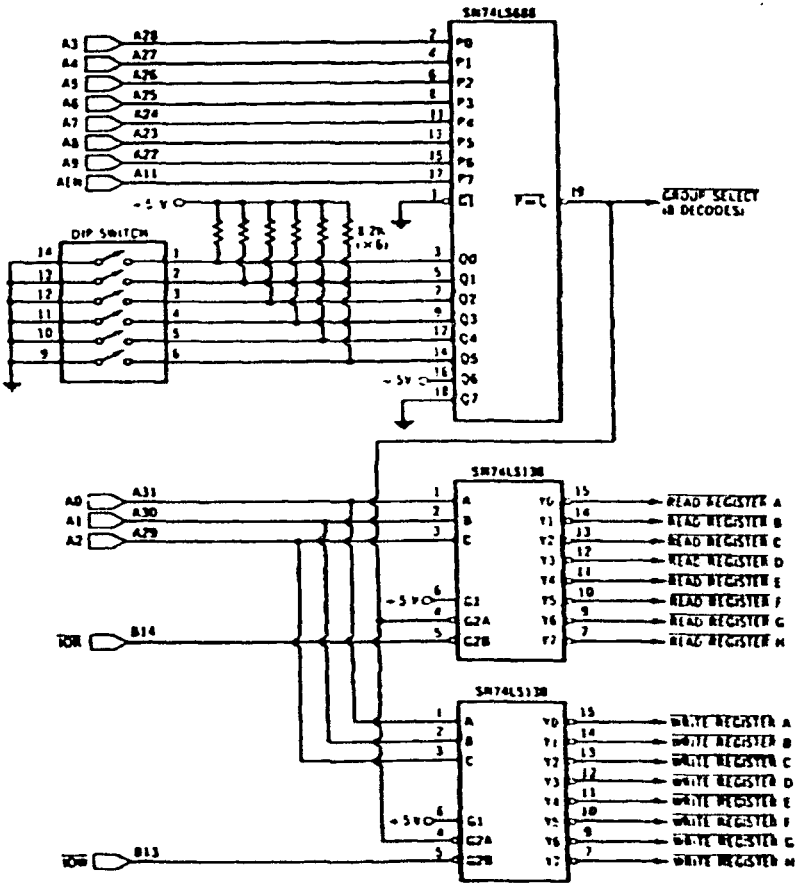


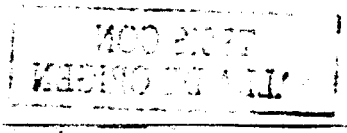
Figura 2.16. Decodificador de puerto entrada/salida seleccionado por interruptores.*

* Circuito esquemático reproducido de la pag. 131 de Eggebrecht Lewis C., *Interfacing to the IBM Personal Computer* Howard W. Sams & Co., E.E.U.U, 1987.

2.2. *EL DISPOSITIVO LÓGICO PROGRAMABLE COMPLEJO*

Por muchos años, los ingenieros electrónicos utilizaron dispositivos electrónicos separados para realizar funciones lógicas. Combinando todos estos circuitos integrados en un solo circuito se creaban sistemas completos, pero éstos presentaban varios inconvenientes. La lógica discreta, implementada con circuitos integrados TTL (*Transistor Transistor Logic*), CMOS (*Complementary Metal Oxide Semiconductor*), etc., si bien cumple las funciones que se le requieran, necesitaba de la implementación e interconexión de muchos y distintos circuitos integrados (CI). El proceso completo de diseño incluía muchos pasos, que involucraban principalmente las pruebas y la verificación, lo cual consume gran cantidad de tiempo y la propensión a errores que implicaba esto.

Conforme avanzó la tecnología, los fabricantes de semiconductores desarrollaron arreglos de compuertas lógicas dentro de un mismo dispositivo, alcanzando a integrar más de 100,000 compuertas en un solo dispositivo. Así, se podía mandar fabricar arreglos de compuertas lógicas por pedido, para lo cual era necesario crear planes esquemáticos que conectaran las compuertas en determinada lógica para realizar las funciones. Sin embargo, los ASICs (por sus iniciales en inglés, que significan Circuitos Integrados para Aplicaciones Específicas), fabricados a la medida, sólo son justificables cuando se va a tener un alto número de producción. Además, cada cambio en el diseño requiere una reconfiguración completa, involucrando todo el proceso de diseño, para determinar que todas las partes se comuniquen y operaren correctamente. Se pueden necesitar más de 30 semanas para completar, probar y verificar el diseño necesario para crear un ASIC. Esta tardanza es una de las principales desventajas que presentan estos dispositivos ya que, económicamente hablando, los ciclos de vida de un producto se han vuelto cada vez menores y el tiempo en que se tarda en lanzar un producto al mercado afecta las ganancias que se puedan obtener de él.



El largo tiempo que toma la creación de las ASICs, hizo que se desarrollaran los PLDs (Dispositivos Lógicos Programables). Estos son circuitos digitales que pueden personalizarse fácilmente por los diseñadores de un sistema. Los PLDs son dispositivos que pueden ser reprogramados fácilmente utilizando *software* de desarrollo operando en una PC o en una estación de trabajo. Con esto se logra combinar bajos costos, facilidad de uso y facilidad de acceso. Igualmente, este proceso elimina cientos de horas de pruebas y reconfiguraciones logrando de esta manera que un diseñador pueda verificar sus nuevos resultados en pocas horas.

2.2.1. INTRODUCCIÓN A LOS CPLDs

Los primeros productos lógicos programables eran implementados utilizando una tecnología que consistía en quemar fusibles. Actualmente sólo se utilizan las tecnologías de fusibles para algunos dispositivos pequeños. Sin embargo, la lógica que encierran los fusibles es la misma que rige a las nuevas tecnologías de interruptor programables, por lo que es conveniente analizarla.

En la figura 2.17 se muestra una representación simplificada de una matriz de compuertas AND, de seis entradas, cuyas salidas están conectadas a una compuerta OR. La intersección de las líneas producto con las líneas de entrada forman una matriz de puertas AND programable de 6x3 fusibles. En esta figura se muestra que las compuertas AND tienen sólo una línea de entrada denominada línea producto. Esta línea se cruza con dos líneas por cada entrada, ya sea directa o invertida, pudiendo existir un fusible en cada intersección. Aunque en la figura sólo se muestra una línea de entrada por cada compuerta AND, en la realidad esta compuerta tiene tantas entradas como intersecciones de la línea producto. Si, en una intersección hay una X, significa que el fusible está intacto y por lo tanto hay una conexión; si no hay una X, el fusible está fundido y no existe conexión. De esta manera, mediante la quema o no de fusibles, el circuito de la fig.

2.17 se ha programado para realizar la función XOR (or exclusiva) entre las entradas A y B.

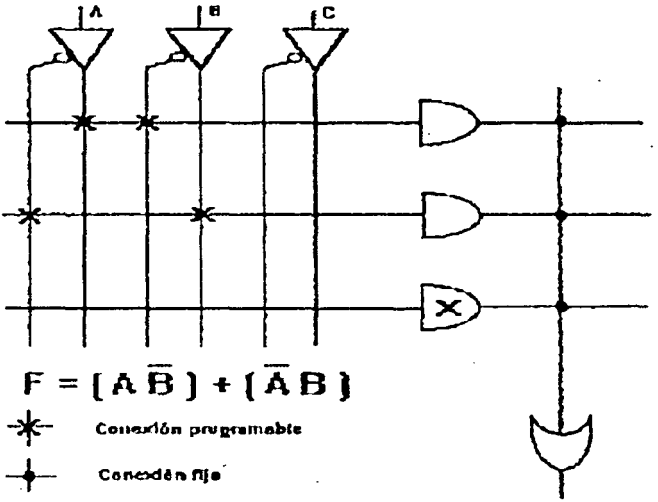


Figura 2.17. Ejemplo de la programación de una matriz de compuertas.

La introducción de esta tecnología de quema de fusibles fueron los comienzos de los PLDs, los cuales eliminaron el tiempo de desarrollo y redujeron los costos, pero presentaron otros inconvenientes:

a) La programación de fusibles era ineficiente con respecto a la utilización del silicio.
b) Debido a su alta potencia de disipación, no se podía conseguir la densidad de integración deseada.

c) Los dispositivos sólo podían ser programados en una ocasión, lo que provocaba que, cuando existían errores en el desarrollo, estos dispositivos debían desecharse, provocando nuevos gastos.
d) El *software* utilizado para reprogramar era primitivo, tedioso y no muy amigable para utilizar.

Así, para lograr dispositivos de mayor densidad, de mayor rapidez, regrabables y de más fácil programación se desarrollaron diferentes técnicas para llevar a cabo interruptores programables que sustituyeran a los fusibles. Estos desarrollos generaron nuevas tecnologías para la fabricación de los dispositivos y su grabado, pero la lógica de la programación de matrices de compuertas se ha mantenido virtualmente idéntica. Estas nuevas tecnologías dieron origen al CPLD.

En los CPLD, las tecnologías de interruptor programables se basan en los transistores de compuerta como aquellos utilizados en las EPROM (*Eraseable Programmable Read Only Memory*) y en las EEPROM (*Electrical Erasable Programmable Read Only Memory*). En estas tecnologías el transistor se coloca de una determinada manera entre dos cables tal que se facilite la implementación de compuertas AND cableadas. En la figura 2.18 se muestran transistores de EPROM que podrían conectarse en un nivel AND de un CPLD.

Así, como se muestra en la fig. 2.18, una entrada al nivel AND puede manejar un cable del producto a nivel lógico cero a través de un transistor de EPROM, en caso de que esa entrada sea parte del término de producto correspondiente. En el caso de aquellas entradas que no estén involucradas en un término del producto, los transistores de EPROM apropiados se programan para ser apagados permanentemente.

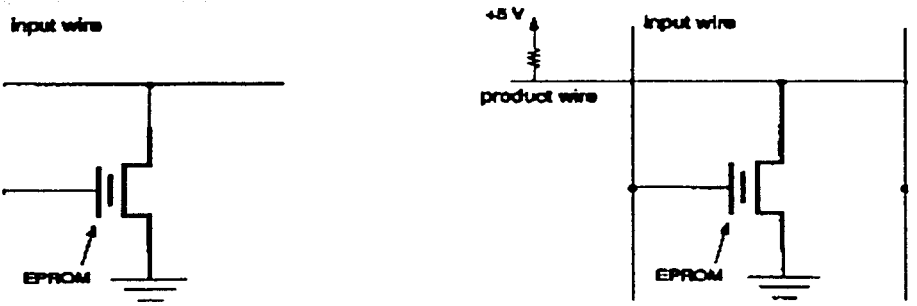


Figura 2.18: Interruptores programables tipo EPROM utilizados en un CPLD[10]

Tal como lo indican sus nombres, la diferencia entre la tecnología EPROM y la EEPROM para la programación de un CPLD difiere principalmente en el método de borrado. En el caso de tecnología EPROM, se utiliza tecnología UVCOMS, lo que implica borrar el chip con luz ultravioleta, mientras que con la tecnología EEPROM la tecnología es EECMOS, lo que implica que el borrado es eléctrico. Los dispositivos utilizados en este trabajo están basados en tecnología EEPROM.

- *DISEÑO CON CPLDs*

Las principales ventajas obtenidas con la utilización de CPLDs se derivan directamente de las ventajas que presentan los sistemas embebidos, desarrolladas anteriormente en el punto 1.3.1. Más específicamente, las cualidades de los proyectos que utilizan los CPLD, en comparación con otras tecnologías se muestran en la tabla 2.3.

[10] Figura reproducida de *Introducción a la Lógica Reconfigurable*, BRASS-Berkeley Reconfigurable Architectures, Systems and *Software*, University of California at Berkeley.

Característica	*Lógica Discreta*	*ASICs*	*CPLDs*
Tiempo de desarrollo	Muy rápido	Lento	Rápido
Densidad	Baja	Muy alta	Muy alta
Creación de prototipos y tiempo de simulación	Muy lenta	Lenta	Muy rápida
Velocidad	Lenta	Muy rápida	Muy rápida
Costo	Bajo	Alto	Bajo
Tiempo de fabricación	Mediano	Largo	Corto
Futuras modificaciones	Difíciles	Imposibles	Fáciles
Soporte técnico	Grande	Bajo	Grande

Tabla 2.3. Comparación de los CPLDs con otras tecnologías.

De la anterior comparación se pueden apreciar las enormes virtudes que presentan los CPLDs, ocasionalmente también llamados EPLDs (Dispositivos Lógicos Programables Eléctricamente). Las causas de estas ventajosas características se deben en gran parte a que, en el diseño de circuitos con CPLDs, se ha vuelto esencial el empleo de programas de diseño asistidos por computadora de alto desempeño (CADs). Estos CADs, sin embargo, son a la vez capaces de operar en una computadora personal con el mínimo de requerimientos técnicos. Un CAD de alto nivel típico para la programación de CPLDs debe incluir el *software* que realice las siguientes tareas: ingreso del proyecto inicial, optimización de la lógica, dispositivo de encajado (*fitting*), simulación y configuración.

En la figura 2.19 se muestra un diagrama de flujo en el que se muestra el diseño de los CPLDs. De este proceso, el diseñador sólo interviene en los pasos de ingreso del programa en la programación y en la simulación. En los otros pasos es el *software* el que realiza automáticamente los demás pasos, a petición expresa del diseñador.

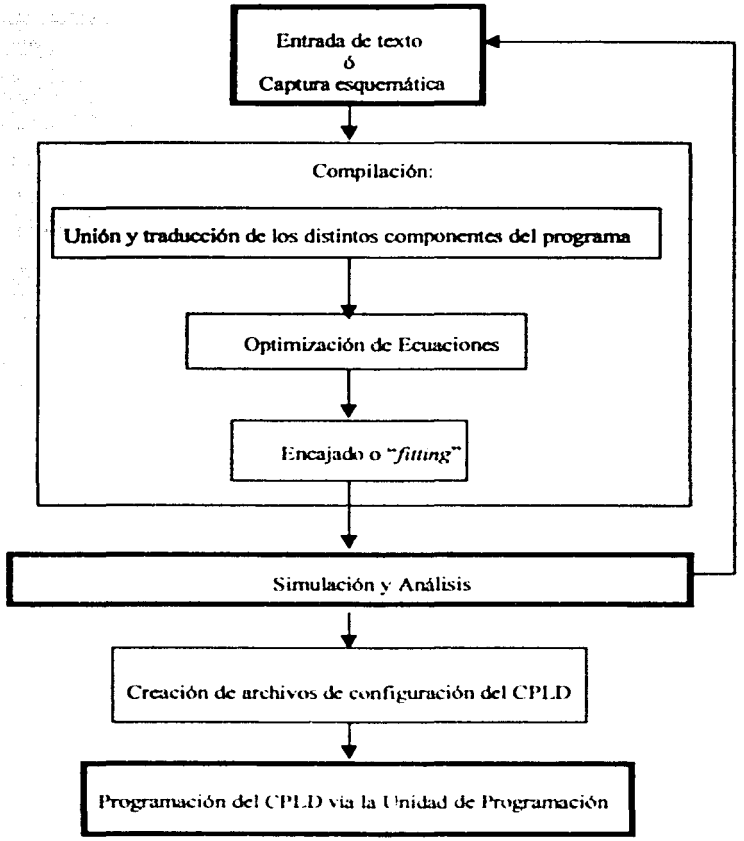


Figura 2.19. Diagrama de flujo para el diseño de un CPLD.

La descripción general de cada paso mostrado en la fig. 2.19 se desarrolla a continuación:

1) La entrada de datos del proyecto puede ser realizada de distintas maneras aunque, básicamente, se emplean dos formas principales:

a) Mediante un sistema de texto para describir el programa mediante un idioma de descripción de *hardware* y funciones lógicas

b) Mediante un diagrama esquemático proveniente de una herramienta CAD gráfica.

En el caso de Max+Plus II, el *software* de programación empleado para este trabajo, el programa de diagramas esquemáticos es capaz de interpretar el editor de esquemáticos ORCAD. Sin embargo, la utilización de diagramas esquemáticos para programar un CPLD no es muy eficiente ya que no permite integrar algoritmos que provean una mayor flexibilidad u optimización al diseño, además de que es más difícil integrar en un solo programa diferentes subrutinas. Es por esta razón que en este trabajo se desarrollará el diseño del CPLD en el sistema de texto *Text Design File* (TDF) con el lenguaje de programación AHDL (*Altera Hardware Decription Language*).

2) En la compilación del programa ingresado, se unen en un solo proyecto de diseño todas las variables, librerías y componentes definidas en el programa. Como normalmente la entrada inicial de la lógica programada no está optimizada, se emplean algoritmos para hacer más eficiente la electrónica de la lógica programada después de que se analizan las ecuaciones lógicas resultantes. Una vez concluido este proceso, realizado por el compilador de Max+Plus II, éste último trata de hacer caber la programación del CPLD en un dispositivo específico de la manera más eficiente mediante un proceso denominado *fitting* o encajado.

3) La simulación se usa para verificar el correcto funcionamiento de la lógica programada en todos los casos posibles y, en caso de no obtener los resultados deseados, el diseñador puede volver a la etapa de entrada para corregir los errores. También se pueden analizar los tiempos de retraso entre las distintas terminales del CPLD, la cantidad de ciclos de reloj que le tomó al CPLD desarrollar una función, etc.

4) Una vez que el diseño ha sido simulado correctamente, se procede a cargarse el proyecto en una unidad de programación o, en su defecto, cargarse vía un cable de interfaz conectado a una base previamente alambrada.

Así, este método de diseño asistido por una computadora, para los CPLDs, permite, entre otros beneficios:

- menores tiempos de producción
- gran facilidad de cambios en los diseños
- una alta densidad de funciones para lograr la mayor integración y calidad en el producto resultante
- una enorme flexibilidad en cada circuito integrado que proviene de la arquitectura programable
- la facilidad de borrar y reprogramar programas según sea conveniente
- la capacidad obtener sofisticados sistemas de desarrollo autocontenidos

Debido a estas grandes ventajas, la preferencia por los CPLDs se ha visto reflejada en los movimientos del mercado. Se estimaba que para el año 2000 el mercado de lógica programable crecería a más de 4.7 mil millones de dólares[11], lo que hace de los PLDs y los CPLDs el sector con mayor crecimiento dentro del mercado de semiconductores.

2.2.2. DISPOSITIVOS MAX7000 DE ALTERA

Altera fue el primer fabricante que superó las dificultades de la lógica programable que utilizaba la tecnología de fusibles cuando introdujo su línea de EPLDs (*Erasable* PLDs), que incorporaban compuertas con tecnología CMOS. La utilización de compuertas CMOS permitió aumentar la velocidad de los dispositivos y además bajar la

potencia de disipación. Esto se tradujo en menores temperaturas de operación lo que, a su vez, permitió a los diseñadores de circuitos embebidos un mayor número de funciones lógicas en un solo circuito integrado. Estos dispositivos de Altera utilizan el mecanismo de programación de un CPLD mencionado anteriormente, con la ayuda de un CAD, lo que permite que el dispositivo pueda ser reprogramado en cualquier momento para modificar su diseño.

Altera ha desarrollado tres familias de circuitos integrados que pertenecen a la categoría de los CPLD. Estas son la familia MAX5000, la familia MAX7000 y la familia MAX9000. El dispositivo que se utilizará en este trabajo es de la familia MAX7000 porque es el más usado y ofrece una muy buena capacidad de lógica y una gran relación de velocidad-actuación. El MAX5000 representa una tecnología más vieja que en ocasiones ofrece una buena solución a un costo muy eficaz, mientras que el MAX9000 es similar al MAX7000, sólo que ofrece una capacidad de lógica más alta, la más alta en el mercado de CPLDs.

La confiabilidad de estos dispositivos de Altera queda demostrada por la aplicación que le dan varios de los principales fabricantes de equipo electrónico que utilizan estos dispositivos para elaborar sus productos. Entre estos fabricantes se encuentran Sony, Siemens, Samsung, Cisco Systems, Bay Networks, NEC y 3Com.

Las principales características que ofrecen los CPLDs de Altera son:

a) Son los más rápidos en el mercado, con velocidades que superan los 200 MHz, pulsos de reloj a más de 622 MHz, y retardos programables entre terminales.

[11] Para más información respecto a la divulgación de los CPLDs en el mercado, consultar la página principal de Altera Products: www.altera.com

b) Tienen una alta capacidad de cambios en el diseño; pueden ser reconfigurados según las necesidades del diseño para entradas/salidas de baja tensión estandarizadas.

c) Operan con un alto rendimiento ya que Altera utiliza los procesos CMOS más avanzados, según se van produciendo, para tener una manufactura confiable. Estos procesos tienen como resultado una alta eficiencia y rendimiento.

d) Debido a la integración lógica de alta densidad, pueden integrar fácilmente la lógica de dispositivos compuestos por lógica discreta, FPGAs, ASICs, etc., reduciendo así espacio en las bases terminadas de los circuitos y en costo.

e) Se obtiene una buena relación producto-costo. Las altas tecnologías que se involucran en la creación de los CPLDs, permite a Altera ofrecer los CPLDs más rápidos, con el más alto rendimiento, a costos equivalentes a los circuitos integrados de amplia difusión.

f) Facilidad de implementación debido a las Macrofunciones (o sencillamente: Macros). Éstas están diseñadas para aumentar la productividad ya que son funciones en bloques prefabricados, ya probados, utilizando la arquitectura del CPLD que reducen el tiempo de desarrollo.

g) Cuentan con uno de los más avanzados *software* de desarrollo para programación de CPLDs: el programa Max+Plus II.

Existen muchos tipos de arquitecturas diferentes para los CPLDs, según su especialización. Sin embargo, existe una arquitectura general, normalmente aplicable para cualquier CPLD, que consiste en varios bloques de entrada/salida, numerosos bloques parecidos a una función PAL, y una matriz de interconexión. La arquitectura general de la familia MAX7000 no es la excepción a esta representación.

Una PAL, o Arreglo Lógico Programable, consiste en un arreglo de términos programables AND que alimentan a términos OR fijos. Así, una macrocelda puede ser programada como entrada, salida o entrada/salida si se utiliza una lógica de tres estados

(alto, bajo y alta impedancia). Sus registros de salida pueden, o no, utilizarse con un a entrada asociado de entrada/salida. Esta arquitectura de las PALs hacen que sean muy populares y son, probablemente, el dispositivo lógico programable más utilizado. Si un dispositivo contiene macroceldas, esto normalmente implica que tiene una arquitectura PAL o una arquitectura equivalente.

En la fig. 2.20 se muestra la arquitectura general del MAX7000. La descripción de este esquema parte de una serie de Bloques de Arreglos Lógicos (LABs), y unas matrices de conexión interconectables llamados Arreglos Programables de Interconexión (PIA).

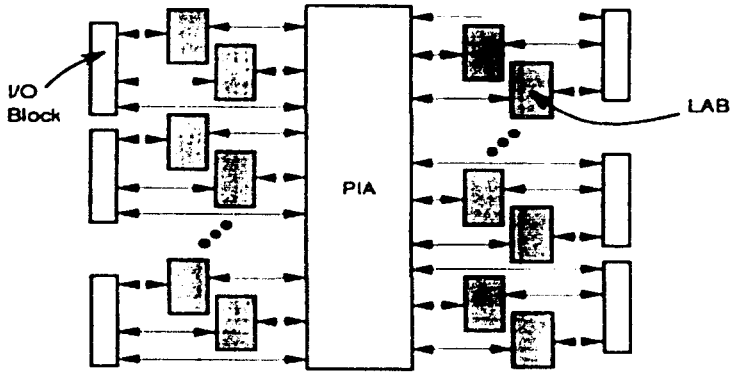


Figura 2.20. Diagrama interno de la serie MAX7000.[12]

El Arreglo Programable de Interconexión es capaz de conectar cualquier entrada o salida del LAB a cualquier otro LAB. Igualmente, las entradas y salidas del dispositivo se pueden conectar directamente al PIA y a los LABs.

[12] Las figs. 2.20, 2.21 y 2.22 son reproducidas de *ALTERA DATABOOK*, Op. Cit.

En la fig. 2.21 se muestra la estructura de un LAB. Éstas son parecidas a una enorme PAL, pero con la capacidad añadida de poder incrementar la cantidad de términos AND para cualquier término OR fijado en su estructura, como en los PLAs. Los PLAs son dispositivos programables que permiten que cualquier término AND alimenten cualquier término OR. Esto permite una enorme flexibilidad con respecto a una funcionalidad lógica.. La analogía se extiende a LAB, sólo que éste está compuesto de macroceldas en vez de términos lógicos. Los LABs de la serie MAX7000 están conformados por dos juegos de ocho macroceldas.

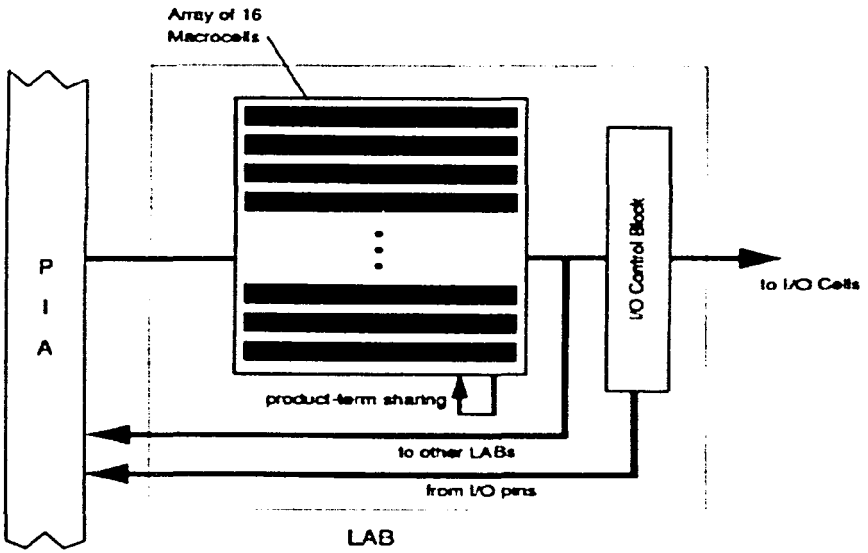


Figura 2.21. LAB utilizado en la arquitectura de la serie MAX7000.

Así, la principal ventaja para este trabajo que se obtiene utilizando LABs es que sus macroceldas se pueden configurar de manera individual, entre una salida

combinacional o una salida secuencial. Esto permite implementar casi cualquier diseño dentro de estos dispositivos.

Una macrocelda de la familia MAX7000, mostrada en la fig. 2.22, está conformada por un juego de términos de productos programables, que parten de los niveles de compuertas AND, que alimentan una compuerta OR y un *flipflop*. Max+Plus II puede configurar estos *flipflops* como tipo D, JK, T o SR.

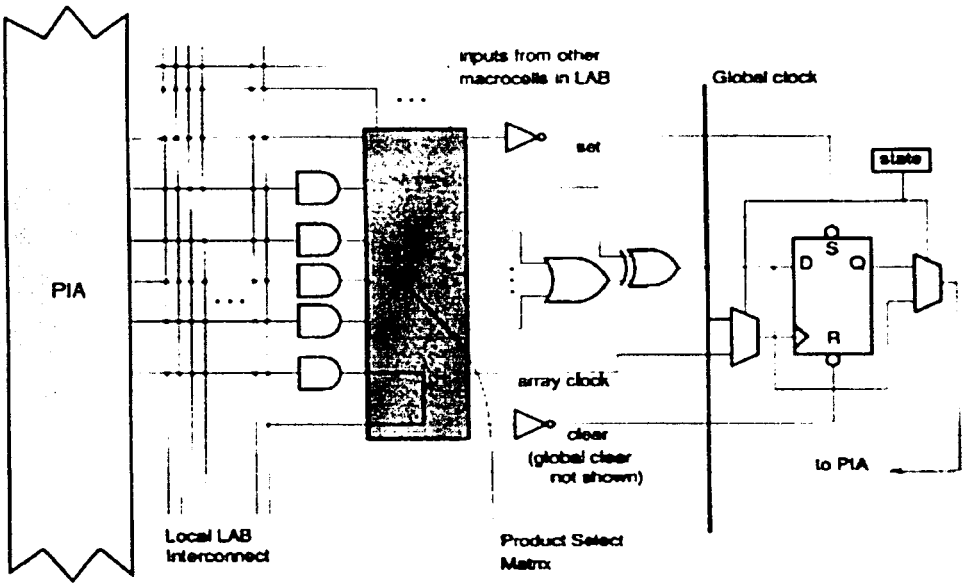


Figura 2.22. Macrocelda de la familia MAX7000.

El número de entradas a la compuerta OR en la macrocelda es variable. Esta compuerta OR puede conectarse a cualquiera, o todos, los cinco productos de términos dentro de la macrocelda. Se puede tener hasta 15 términos extras del producto de macroceldas en el mismo LAB. Esta flexibilidad de los productos de términos hace que el LAB de la serie MAX sea el más eficaz en términos de área del dispositivo, porque las funciones de lógica típicas pre programadas, las macrofunciones, no necesitan más de cinco productos de términos. Igualmente, en caso de que fuera necesario, la arquitectura es capaz de operar con productos de términos más anchas.

- *EL CPLD EPM7064LC68*

Una vez que se han analizado los principales características de la familia MAX7000, se puede proceder a examinar más específicamente el CPLD EPM7064LC68, ya que éste será el dispositivo que se utilizará como elemento principal en el desarrollo del presente trabajo. Las características distintivas del EPM7064LC68 se resumen en tabla 2.4:

Número de compuertas utilizables	1,250
Número de Macroceldas	64
Número de LABs	4
Cantidad de terminales del CPLD	68
Cantidad de terminales de entrada/salidas disponibles para el programador	52
Tiempo mínimo entre una terminal de entrada a una de salida: t_{PD}	6 ns
Tiempo del pulso instalado en el reloj global: t_{SU}	5 ns
Frecuencia máxima a la que pueden operar sus contadores internos: f_{CONT}	151.5 MHz

Tabla 2.4: Características del EPM7064LC68.

Las condiciones de operación normales más importantes del EPM7064LC68 se resumen en la tabla 2.5.

Parámetro	*Valor mínimo*	*Valor máximo*
Tensión de alimentación: V_{CC}	-2.0 V	7.0 V
Corriente DC de salida, por terminal: I_{OUT}	-25 mA	25 mA
Tensión de suministro para la lógica interna y de alimentación para los buffers de entrada : V_{CCINT}	4.75 V	5.25 V
Tensión de alimentación para los buffers de salida: V_{CCIO} (en operación de 5.0V)	4.75 V	5.25 V
Tensión de las señales de entrada: V_I	-0.5 V	V_{CCINT} + 5V
Tensión de las señales de salida: V_O	0 V	V_{CCIO}
Temperatura ambiente de operación recomendada: T_A	0°C	70°C
Tiempo de levantamiento de entrada: t_R		40 ns
Máximo tiempo de caída de entrada: t_f		40 ns
Corriente requerida por una terminal de entrada/salida de tres estados: I_{OZ}	-40 µA	40 µA

Tabla 2.5. Condiciones de operación normales del EPM7064LC68.

Este CPLD tiene la capacidad de realizar interfazs a una tensión distinta a la de 5.0V. Si esto se establece en las preferencias de programación, el CPLD es capaz de reconocer niveles lógicos de 3.3V ya sea de entrada o de salida. En este trabajo, sin embargo, sólo se trabajará con la lógica de 5.0V. También cuenta con registros programables, extensores paralelos con fines de ampliación (productos de términos invertidos que son retroalimentados al arreglo lógico) y extensiones compartidas con fines de ampliación (productos de términos prestadas de macroceldas adyacentes) para cuando la lógica no cabe en un solo producto de términos. Incluso tiene la capacidad de operar en un estado de ahorro de energía, en el caso de que la mayoría de sus aplicaciones lógicas operen a una frecuencia muy pequeña, comparada con la frecuencia máxima de operación del CPLD. Igualmente, tiene la capacidad de ser programado con un bit de

seguridad que puede ser activado para evitar que el CPLD sea leído ni copiado. El EPM7064LC68 puede ser programado y borrado hasta 100 veces.

El diagrama del EMP7064LC68 se muestra en la fig. 2.23.

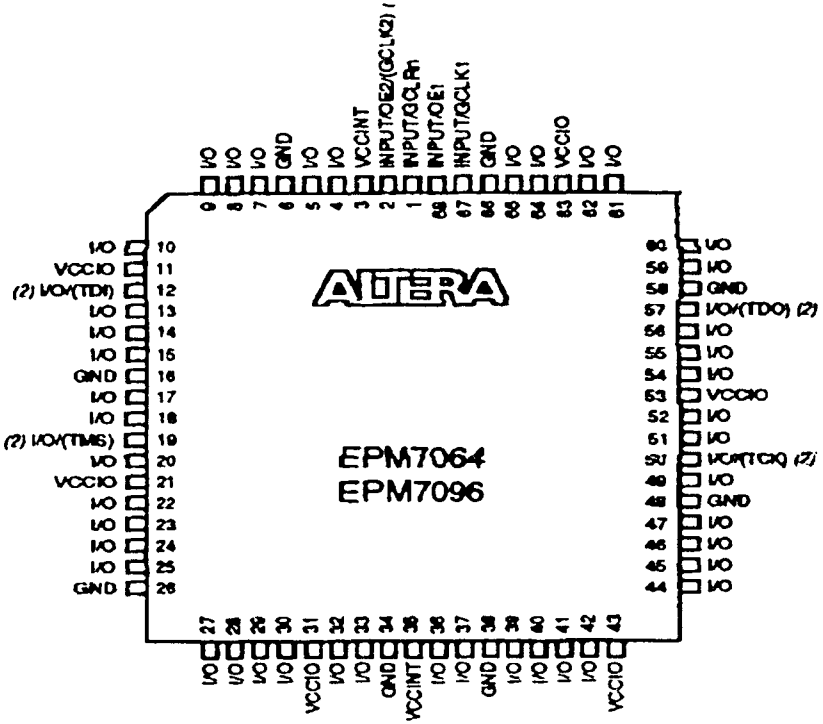


Figura 2.23. Diagrama del EPM7064LC68.

Existen ciertas terminales del EPM7064LC68 que no están disponibles para el programador. Éstas son: 3 y 35: VCCINT; 6, 16, 26, 34, 38, 48, 58 y 66: GND; y 11, 21,

31, 43, 53 y 63: VCCIO. Existen otras cuatro terminales que sólo pueden configurarse como entradas, éstas son: 67, 1, 68 y 2. Existen también otras cuatro terminales que cumplen una doble función, ya que, aunque pueden ser programados indistintamente, también sirven para la programación del CPLD (ver sección 2.2.3). Estas son: 12 (TDI), 19 (TMS), 50 (TCK) y 57 (TDO). En todas las demás terminales no existen restricciones.

La utilización de los recursos del CPLD puede ser analizada con el *software* Max+Plus II. El análisis de la utilización de los distintos bloques del EPM7064LC68, permite acomodar en él la lógica de los programas y distribuirlos de una manera eficiente.

Debido a que parte de la justificación de la utilización específica del CPLD EPM7064LC68 tiene que ver con las capacidades que ofrece y los requerimientos que necesita el programa desarrollado en la sección 4.2, es necesario adelantarse un poco a éste, resumiendo el aprovechamiento de los recursos del EPM7064LC68, después de programar el programa de la sección 4.2, como se muestra en las tablas 2.6 y 2.7.

Total de celdas lógicas utilizadas	49 de 64
Número de *flipflops* requeridos	36
Número de productos de términos requeridos	148
Total de terminales de entrada/salida utilizados:	48 de 48
Terminales de entrada	31
Terminales de salida	9
Terminales bidireccionales	8
Terminales dedicados de entrada utilizados	4/4 (100%)

Tabla 2.6. Utilización de los recursos del EPM7064LC68.

LAB	*Celdas Lógicas*	*Terminales Entrada/Salida*	*Extensores Compartidos*	*Conexiones Externas*
A	LC1 - LC16 9/16 (56%)	12/12 (100%)	7/16 (43%)	33/36 (91%)
B	LC17 - LC32 15/16 (93%)	12/12 (100%)	2/16 (12%)	27/36 (75%)
C	LC33 – LC48 14/16 (87%)	12/12 (100%)	4/16 (25%)	26/36 (72%)
D	LC49 – LC64 11/16 (68%)	12/12 (100%)	5/16 (31%)	20/36 (55%)

Tabla 2.7. Aprovechamiento de los recursos del EPM7064LC68 después de programarle el programa de la sección 4.2.

Como el CPLD de la familia MAX7000 inmediatamente inferior al EPM7064LC68 en cantidad de terminales es el EMP7064LC44, que cuenta con solo 36 terminales disponibles para el programador, mientras que el inmediatamente inferior en capacidad es el EPM7032LC44, el cual cuenta con sólo 2 macroceldas, es evidente que estos dos modelos resultan insuficientes para este trabajo. De igual manera, el CPLD inmediatamente superior en terminales al EPM7064LC68 es el EPM7064LC100, que cuenta con 76 terminales disponibles para el programador, mientras que el inmediatamente superior en capacidad es el EPM7096LC68, el cual cuenta con 6 macroceldas, resultando excesivos para este trabajo.

Del anterior análisis y de la comparación de las capacidades que ofrece el EPM7064LC68, resumidas en la tabla 2.4, con las necesidades demandas por este trabajo, resumidas en la tabla 2.6, puede observarse que el EPM7064LC68 resulta el idóneo para la realización de la interfaz, ya que casi no existe subutilización de sus capacidades, y la cantidad de entradas y salidas que ofrece son las adecuadas según las necesidades de la programación.

2.2.3. DESARROLLO CON EL SOFTWARE MAX+PLUS II

Max+Plus es la abreviatura de *Multiple Array matriX Programmable Logic User System*, que en castellano puede expresarse como Sistema de Usuario para la Lógica Programable de Matrices de Múltiples Arreglos. Es un *software* que permite un diseño de arquitectura de múltiples plataformas, permitiendo un fácil ingreso del diseño, una rápida compilación y programación directa del dispositivo. Está incluido completamente en un solo paquete de *software* que sirve para programar los CPLDs de Altera, incluyendo las familias MAX5000, la MAX7000, la MAX7000E, la FLEX8000, la FLEX8000M y la Classic.

El *software* Max+Plus II ofrece un espectro completo de capacidades de diseño lógico. Entre estas se incluyen la variedad de ingresos de diseños, una poderosa lógica de síntesis, capacidad de realizar particiones, simulaciones funcionales, análisis de tiempos, localizaciones automáticas de errores y programación de dispositivos y verificaciones.

El lenguaje de programación utilizado es el AHDL o Lenguaje de Descripción para el *Hardware* de Altera Éste es un lenguaje de programación modular de alto nivel, similar al VHDL (*Verilog Hardware Description Language*), integrado en el sistema Max+Plus II. Está especialmente orientado al diseño de lógica combinacional compleja, tablas de verdad, lógica parametrizada y circuitos secuenciales. Utilizando este lenguaje se puede desarrollar la programación del CPLD en cualquier editor de texto ASCII o, preferentemente, en el Editor de Texto de Max+Plus II, el cual presenta un ambiente más amigable para la programación.

Una vez que se ha ingresado un diseño, el sistema, completamente automatizado de Altera, denominado Procesador de Diseño de Altera (ADP), traduce el diseño a un archivo de programación para PLDs estandarizado por la industria, el JEDEC. Este archivo se utiliza para programar directamente el dispositivo a programar, por medio de un cable de interfaz entre la PC y el CPLD.

El traslado del diseño desde su formato de ingreso original, ya sea archivo de texto, esquemático u otro, a un mapa de programación para un dispositivo específico, es un proceso que se inicia cuando se compila el programa. Este proceso también busca optimizar las funciones programadas utilizando procesos de minimización, para ayudar al diseñador a minimizar y optimizar su lógica y así ingresar el programa a un dispositivo específico (el manual de Max+Plus II refiere escuetamente que uno de los procesos de minimización que utiliza es un algoritmo llamado "Inversión de DeMorgan").

La forma en que el ADP de Max+Plus II logra la inserción de los programas en los CPLDs de una manera que minimice los productos, *flipflops*, etc. Se realiza mediante tres funciones:

- Primero, sin importar la forma en que se hayan ingresado el programa, se traduce el material del diseño a ecuaciones lógicas internas. En esta etapa, la mayoría de los errores de sintaxis son detectados y reportados al diseñador.
- En segundo lugar, el ADP realiza reducciones con lógica booleana en el diseño ya traducido para optimizar los recursos del CPLD.
- Finalmente, el ADP asigna los requerimientos a los diseños específicos de Altera. Automáticamente o manualmente (en el caso de que se asignen las terminales de conexión). El ADP ubica las funciones lógicas en su óptima localización, seleccionando las rutas adecuadas de las interconexiones interiores.

Todo este proceso arroja distintos reportes del proceso y un archivo de programación estandarizada JEDEC. Los reportes informan al diseñador cómo se implementó el diseño y señala aquellos recursos no utilizados en el dispositivo.

El proceso de diseño y desarrollo sigue un orden jerárquico establecido, comprobable en el la ventana de jerarquía , cuya secuencia básica de programación es la siguiente:

1) Se crea o se abre el proyecto.

2) Se ingresa un diseño en un archivo de texto cuya extensión es .tdf (*text design file*). Este archivo contiene el programa escrito en AHDL.

3) Se compila el programa. En este paso se realizan las siguientes funciones, en el orden mencionado:

- El compilador extrae el listado en que está contenido el programa.
- Se construye una base de datos.
- Se procesa el programa con un sintetizador lógico.
- Se encaja el programa en el dispositivo (*fitting*).
- Se crea un archivo con la información de la propagación de tiempos de las señales (SNF).
- Se ensambla el programa.

4) Al terminar la compilación, en caso de no existir errores de programación que serían notificados en la ventana de mensajes del compilador, se crean los siguientes archivos:

- *rpt*: Contiene toda la información teórica y técnica del proyecto y del dispositivo programado.
- *fit*: Contiene la información con la que el compilador realizó la inserción del programa en el dispositivo.
- *pin*: Contiene la correspondencia entre las señales de entrad/salida del programa y las terminales del dispositivo.

5) Para comprobar y simular la programación, se asocian señales de entrada a las terminales del dispositivo en el editor de formas de onda, guardándose en un archivo *.scf (signal create file)*.

7) Se corre el simulador de tiempos y se verifican los resultados

8) Se programa el dispositivo vía una interfaz que una la PC con el programador.

Las interfazs que Altera ofrece para programar el CPLD desde la PC son los siguientes:

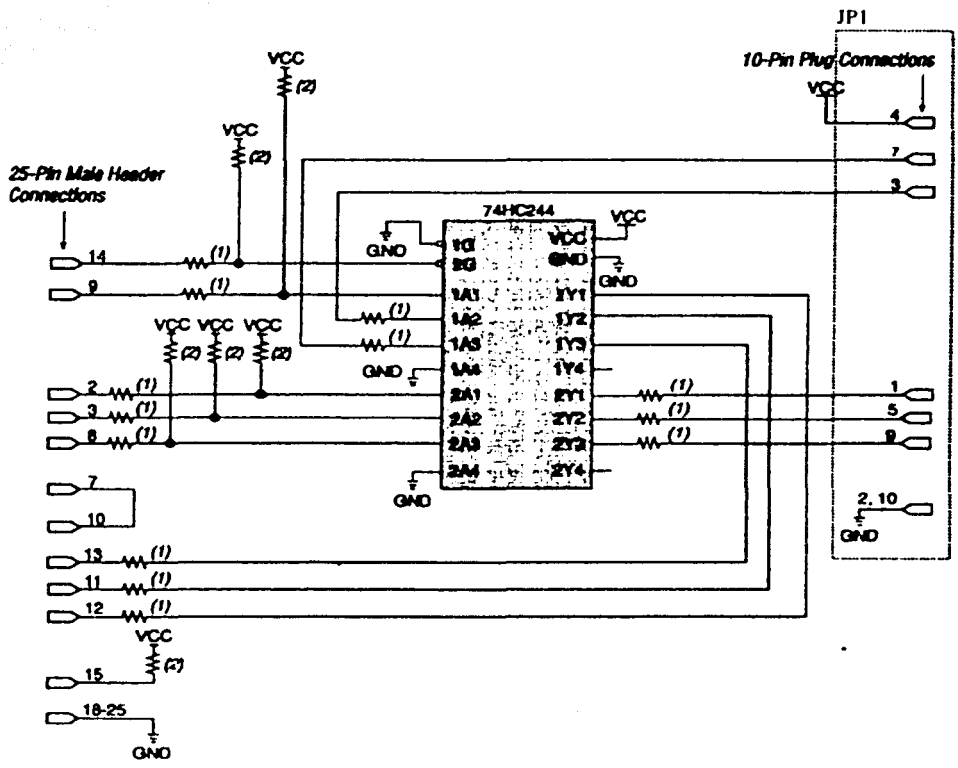
- MasterBlaster Serial.- Consiste en la interfaz entre la PC y el CPLD vía el puerto serial.
- ByteBlasterMV.- Consiste en la interfaz entre la PC y el CPLD vía el puerto paralelo.
- MasterBlaster USB.- Consiste en la interfaz entre la PC y el CPLD vía el puerto USB.

Los dispositivos que pueden programarse con cualquiera de estas interfazs son aquellos pertenecientes a las familias APEX 20K, ACEX 1K, FLEX 10K, FLEX 8000, FLEX 6000, MAX 9000, MAX 7000 y MAX 3000A. Estas interfazs operan tanto con el *software* Max+Plus II como con el *software* Quartus II.

En la figura 2.24 se muestra el diagrama esquemático de un circuito que realiza una interfaz equivalente al ByteBlasterMV. La interfaz DB-25 se conecta al puerto paralelo de la PC, mientras que el conector de 10 terminales, designado JP1 en el diagrama, es el que se conecta a la base donde se encuentra alojado el CPLD a programar.

De esta manera, disponiendo de unas terminales dedicadas en la interfaz electrónica cuyas terminales coincidan con los valores de la tabla 2.8, correspondientes asimismo a los contactos de JP1, el CPLD se puede programar desde la PC con el *software* de Max+Plus II (suponiendo, desde luego, que se conecten estas señales a los respectivas terminales del CPLD).

Cabe mencionar que el modo de interfaz para programación utilizado es el JTAG (del inglés: *Industry-standard Joint Test Action Group*), ya que es el empleado para configurar a los dispositivos APEXII, APEX20K, Mercury, ACEX 1K, Excalibur, FLEX 10K, MAX 9000, MAX 7000S, MAX7000A, MAX7000B y MAX3000A.



Notes:
(1) All series resistors are 100 Ω.
(2) All pull-up resistors are 2.2 kΩ.

Figura 2.24. Interfaz entre la PC y el CPLD para su programación[13].

[13] Circuito obtenido de *ByteBlasterMV Parallel Port Donload Cable Data Sheet*, Altera Corporation, http://www.altera.com/literature/ds/dsbytemv.pdf

	Programación en modo JTAG	
Terminal	***Nombre de la Señal***	***Descripción***
1	TCK	Señal de Reloj
2	GND	Tierra
3	TDO	Información desde el dispositivo
4	VCC	Fuente de tensión
5	TMS	Control JTAG
6	NC	No utilizado
7	NC	No utilizado
8	NC	No utilizado
9	TDI	Información hacia el dispositivo
10	GND	Tierra

Tabla. 2.8. Terminales de conexión entre la interfaz y el CPLD para su programación.

Una vez analizados los principales componentes que componen la interfaz electrónica, su funcionamiento y sus características de operación, se puede proceder a describir la forma específica en que estos componentes se relacionan, y la lógica detrás de ésta, para que la interfaz se lleve a cabo.

III
LÓGICA DEL DISEÑO DE LA INTERFAZ

La organización metódica de la lógica digital, con la cual la interfaz electrónica lleva a cabo su cometido, organizada en módulos específicos, permite entender el proceso de la transferencia de datos entre la micro computadora, la interfaz y los periféricos del telescopio e, igualmente, establece la lógica a programar en el CPLD que realiza las principales funciones de la interfaz.

3.1. LÓGICA DEL DISEÑO Y OPERACIÓN DE LA INTERFAZ

De acuerdo con el planteamiento desarrollado en la sección 1.2 y en la 1.3 de este trabajo, el objetivo principal de este proyecto es proponer un diseño que integre al máximo la electrónica de la tarjeta de interfaz MAGALI en un CPLD, para conseguir la mayor reducción posible del número de componentes y facilitar la implementación de otras tarjetas de interfaz. Así, la tarjeta MAGALI (fig. 3.1), compuesta de 34 circuitos integrados y demás componentes analógicos, representa el modelo lógico a embeber y optimizar, obteniendo de esta manera las ventajas que representan los sistemas embebidos, ventajas ya descritas en la sección 1.3.1. Es por esta razón que en esta sección se presentan optimizaciones de algunas secciones de la tarjeta MAGALI, optimizaciones cuyo fin están destinadas a conseguir una integración más eficaz, ya que es a partir de éstas que se basan los principios de programación del CPLD de la interfaz. El diagrama esquemático de la electrónica alambrada en la tarjeta MAGALI se puede consultar en el Apéndice A.

Igualmente, se presentarán algunas adaptaciones hechas a la tarjeta de desarrollo modelo PCL-750 de la Compañía Advantech, Ltd., sobre la cual se implementó la tarjeta MAGALI, con el fin de optimizarla para los fines de este trabajo. La tarjeta PCL-750 trae integradas varias funciones que permiten ahorrar el trabajo y tiempo de diseño e implementación de varios circuitos, presentando a la vez un gran área para el desarrollo del diseñador de *hardware* de 206.45 cm^2 con 3,290 perforaciones espaciadas específicamente para implementar sobre ella bases para *wire-wrap*. La lógica ya implementada se muestra en la fig. 3.1 en el rectángulo inferior derecho de la tarjeta, mientras que el diagrama esquemático de la tarjeta de desarrollo PCL-750 se puede consultar el Apéndice B.

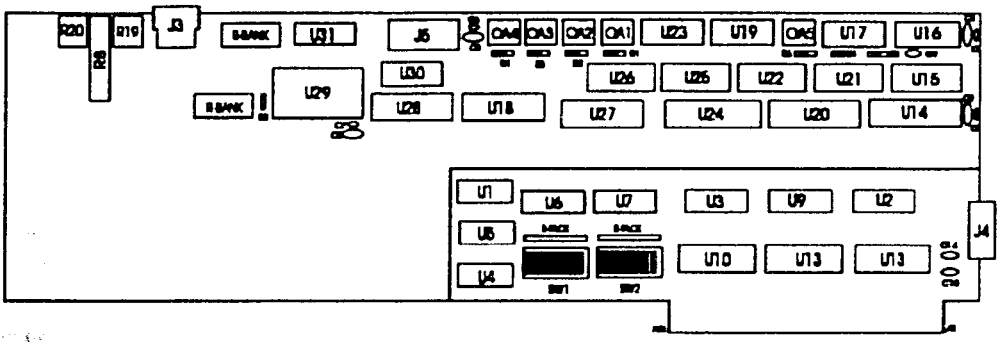


Figura 3.1. Aspecto físico de la tarjeta MAGALI.

La lógica adaptada de la tarjeta PCL-750 también será embebida dentro del CPLD. Las funciones que trae integrada la tarjeta PCL-750 son: la integración de un buffer (el CI 74LS245) que administra el *bus* de datos; la integración de dos buffers (el CI 74LS244) que administran al *bus* de direcciones; una electrónica, compuesta elementalmente por un comparador (el CI 74LS688) y un *dip-switch*, que decodifica la dirección de entrada/salida a memoria; una electrónica, compuesta elementalmente por un comparador (el CI 74LS688) y un dip-switch, que decodifica la dirección de entrada/salida a puerto; ocho señales de control de entrada/salida para escritura (control de escritura de registros *Read Write*) y ocho señales de control de entrada/salida para lectura (control de lectura de registros *Read Write*).

Con el fin de organizar metódicamente la explicación de la lógica del diseño de la interfaz, se han establecido módulos específicos, en los cuales se subdividen las funciones de la interfaz a desarrollar en este trabajo. Estos módulos son:

a) Módulo de los movimientos de AR y DEC.- En este módulo se trata la lógica que interpreta las señales provenientes del controlador manual del telescopio y su traslado al *bus* de datos.

b) Módulo del reloj sideral.- En este módulo se trata la lógica que adquiere la frecuencia del reloj sideral, su traslado al *bus* de datos, su habilitación o deshabilitación junto con el controlador manual para, igualmente, habilitar la señal del reloj sideral del controlador de motores del telescopio, *Galil*, y la indicación de la terminación de movimientos a *Galil*.

c) Módulo de los sensores.- En este módulo se trata la lógica que obtiene las lecturas analógicas de los sensores, su conversión a formato digital y su traslado al *bus* de datos.

d) Módulo del control de las habilitaciones.- En este módulo se trata la lógica de las habilitaciones de las señales de control del *bus* ISA y el control de las señales IOW e IOR.

e) Módulo de decodificación.- En este módulo se trata la lógica que realiza la decodificación de puertos de entrada y salida, vía el *bus* ISA, siguiendo el razonamiento desarrollado en la sección 3.1.5.

La lógica básica de diseño es que, a partir de los módulos anteriores, se implementa un *bus* de datos bidireccional, al cual se conectan todas las señales, ya procesadas, provenientes de los periféricos. Estas señales serán ingresadas a la computadora en el momento en que el microprocesador lo requiera por medio de las señales IOR ó IOW y el *bus* de direcciones.

Las denominaciones de las señales que ingresan a la interfaz, desde los periféricos hacia la computadora, son:

1) Indicadores de los cuatro movimientos del telescopio provenientes del controlador manual:
   - Dirección de ascensión recta (DIR AR).
   - Reloj de ascensión recta (CK AR).
   - Dirección de declinación (DIR DEC).
   - Reloj de declinación (CK DEC).
2) Frecuencia sideral (90 Hz SID y GND 90 Hz para sus terminales positivo y de tierra respectivamente).
3) Lectura del sensor de temperatura (TEMP).
4) Lectura del sensor de humedad (HUMEDAD).
5) Dos indicadores de la finalización de movimientos de parte de *Galil* (MOVIMIENTO TERMINADO X, MOVIMIENTO TERMINADO Y)

Las denominaciones de las señales que salen de la tarjeta, al controlador de motores *Galil*, son:

1) Habilitador del puerto de *Galil* para acreditarlo (HABILITA PUERTO GALIL).
2) Habilitador de la función de guiado que ingresa a *Galil* mediante la transmisión de la frecuencia sideral (R SID GALIL).

3.1.1 LÓGICA PARA LOS MOVIMIENTOS DEL TELESCOPIO

Para entender la lógica detrás de la adquisición de las señales que controlan los movimientos del telescopio, hay que mencionar que las señales provenientes del controlador manual del telescopio son un tren de pulsos, independientes para cada eje de movimiento (CK AR ó CK DEC). Este controlador manual tiene en su estructura interna un generador de trenes de pulsos, el CI 74HC4046, cuya frecuencia central depende de un arreglo de resistencias y de un capacitor, mientras que su capacidad de variar su

frecuencia alrededor de esa frecuencia central, se basa en el nivel de tensión que ingresa a una de sus terminales (la VCOIN). Así, el controlador manual opera en un rango entre 330 y 670 kHz en su modalidad de guiado lento, y entre 1.38 y 1.42 MHz aproximadamente en su modalidad de guiado rápido.

La modalidad lento/rápido se selecciona mediante un interruptor que cambia las resistencias que establecen la frecuencia central, mientras que la variación de frecuencia alrededor de la frecuencia central seleccionada (lenta o rápida), se realiza mediante un potenciómetro implementado en un divisor de tensión que ingresa a la terminal VCOIN del CI. El diagrama esquemático del controlador manual se muestra en el Apéndice C.

De esta forma, lo trenes de pulsos provenientes del controlador manual del telescopio tienen un ciclo de trabajo del 50% (señal cuadrada) y su frecuencia varía según la posición de un potenciómetro colocado para tal fin en el controlador manual, entre un mínimo de 330 kHz y un máximo de 1.41 MHz aproximadamente. Así, si el operador del telescopio desea incrementar la velocidad del movimiento, se aumenta la frecuencia del tren de pulsos y viceversa, si el operador desea movimientos más lentos, se baja la frecuencia del tren de pulsos.

La dirección del movimiento del eje se determina según un bit de dirección (DIR AR ó DIR DEC) que, cuando está activado, indica que el movimiento es negativo (-) y que, cuando está desactivado, el movimiento es positivo (+). Se presentan así cuatro casos posibles en los que las señales CK AR, DIR AR, CK DEC y DIR DEC se activan o desactivan según se resume en la tabla. 3.1.

Botón presionado en el controlador manual (*handset*)	Tren de pulsos CK AR	Bit DIR AR	Tren de pulsos CK DEC	Bit DIR DEC
Ascensión Recta Positiva (AR +)	Sí	No	No	No
Ascensión Recta Negativa (AR -)	Sí	Sí	No	No
Declinación Positiva (DEC +)	No	No	Sí	No
Declinación Negativa (DEC -)	No	No	Sí	No

Tabla 3.1. Estados posibles de las señales provenientes del controlador manual.

El circuito esquemático del módulo de la interfaz que realiza el procesamiento y la interpretación de los datos para los movimientos del telescopio, se muestra en la fig. 3.2.

En la siguiente explicación se detalla el funcionamiento de la electrónica para interpretar los movimientos de ascensión recta, siendo esta misma explicación aplicable a la interpretación de datos de los movimientos de declinación.

Al llegar las señales provenientes del controlador manual al circuito, éste realiza la cuenta del número de pulsos provenientes del controlador manual. Así, cuando el tren de pulsos ingresa a la interfaz, éste es ingresado al reloj de un contador bidireccional de 8 bits para que la cuenta del contador se incremente con cada pulso de reloj, para posteriormente enviar la suma de los pulsos, vía un buffer (el CI 74LS244) y el *bus* de datos interno del CPLD, el conteo que, procesado por el programa de guiado de la computadora, indica al controlador de los motores el inicio y finalización de cada movimiento y su velocidad. La tarjeta MAGALI, no teniendo un contador de 8 bits real, utiliza dos contadores de 4 bits, los CI 74LS191, conectados en cascada, para realizar conteos de 8 bits. El contador programado en el CPLD utiliza la macrofunción 8_COUNT, lo que ahorra el número de LABs utilizados.

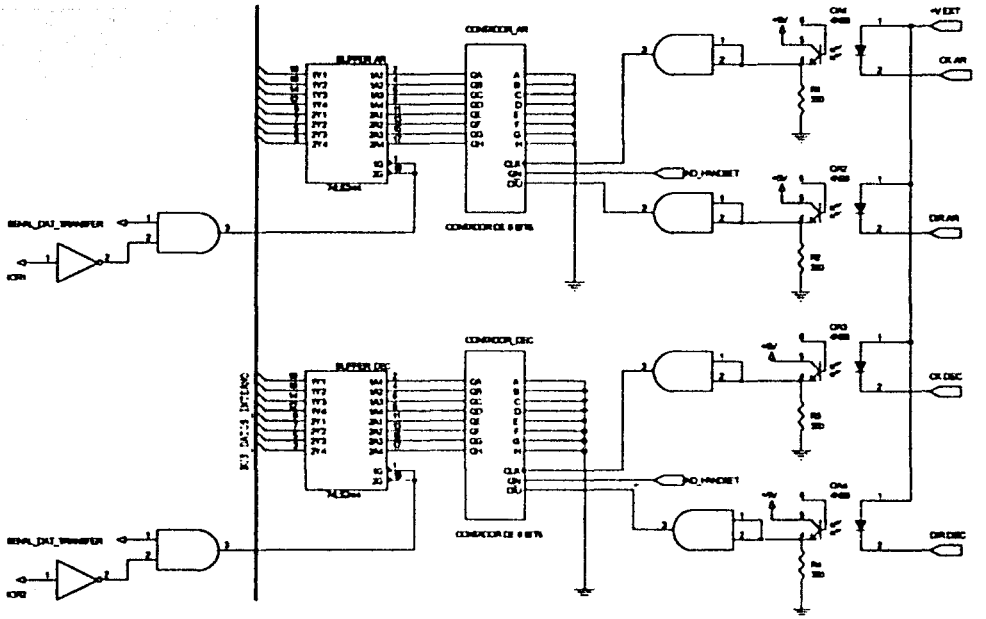


Figura 3.2. Lógica que interpreta los movimientos de AR mandados por el controlador manual.

Para determinar si el movimiento es positivo o negativo, el circuito realiza el conteo en forma ascendente o en forma descendente según sea el bit de dirección mandado desde el controlador manual, el cual ingresa a la terminal *Up/Down* del contador. El programa de guiado interpreta la dirección del movimiento según el signo de la suma entregado por el contador.

Así, a mayor cantidad de pulsos que se cuente en el tiempo fijo en que se realiza la lectura del movimiento, el programa de guiado interpretará que el operador solicita una mayor velocidad en los motores y viceversa.

Las señales IOR (IOR1 para el caso de la ascensión recta e IOR2 para el caso de la declinación) se conectan a una compuerta AND para que, sólo cuando coincidan con la señal denominada SENAL_DAT_TRANSFER, se dé la transferencia de datos. Esto se debe a que la señal resultante habilita la transferencia de datos dentro de la interfaz al activar a los habilitadores de los buffers (1GATE y 2GATE). Cabe hacer notar que, cuando se utiliza físicamente un 74LS244, la señal para habilitar a los buffers necesariamente debe ser de nivel bajo, pero, a nivel de programación del CPLD, este buffer se habilita con una señal de nivel alto. Igualmente, todas las señales IOR, en este caso IOR1 e IOR2, deben ser negadas ya que llegan invertidas desde el módulo del control de las habilitaciones.

Adicionalmente, los contadores son habilitados por la llegada de la señal NO HANDSET, al estar conectada a la terminal *GN* (*Gate*) de los contadores. La señal NO HANDSET es un deshabilitador del proceso de conteo de pulsos provenientes del controlador manual, que se utiliza cuando el programa de la consola de control está poniendo en posición el telescopio o cuando se desea bloquear el movimiento de la montura ecuatorial.

Con el fin de evitar indeterminaciones en los valores lógicos de la señales digitales provenientes de los periféricos, vía su respectivo optoacoplador, cada señal es transferida a una compuerta lógica AND, cuyas dos entradas se encuentran unidas. Estas indeterminaciones o tensiones no claramente definidas como de valor alto o de valor bajo, son debidas a las interferencias que se agregan a los trenes de pulsos provenientes del controlador manual del telescopio. De esta manera, al utilizar una compuerta AND cuyas salidas son siempre definidas, se logra que las señales provenientes del controlador

manual tengan siempre un valor definido de "1" lógico ó "0" lógico. Finalmente, cabe aclarar que, en la tarjeta MAGALI se utilizan compuertas NAND para este propósito, ya que, al utilizar el CI 74LS132 que contiene cuatro de estas compuertas, se pueden implementar otras funciones lógicas con compuertas restantes. Sin embargo, para la integración del CPLD es preferible utilizar compuertas AND, ya que éstas ocupan menos espacio en las macroceldas.

Cabe hacer notar que no existen resistencias limitadoras de corriente para las señales que provienen del controlador manual del telescopio al ingresar a los optoacopladores. Esto es debido a que el controlador manual está conectado a la interfaz electrónica con un cable de un poco más de 10 m., que llega hasta donde se encuentra ubicada la consola de control del Observatorio Dicho cable se comporta como una resistencia limitadora de corriente, volviendo innecesarias más resistencias ya que, en experimentos realizados en el laboratorio del IAUNAM, se comprobó que el efecto de añadir otra resistencia en la entrada del acoplador daba como resultado un decaimiento demasiado grande de los trenes de pulsos provenientes del controlador manual, lo que los volvía ilegibles. Además, esta longitud permite con holgura la manipulación del controlador manual en la base del telescopio por parte de los astrónomos, sin tropezarse con él.

3.1.2. LÓGICA PARA EL RELOJ SIDERAL

Esta interfaz cuenta con una entrada denominada 90_Hz_SID (ver fig. 3.3) por la que llega una frecuencia de 90 Hertz Siderales, que entra al bus de datos interno del CPLD a través de un contador de 4 bits (un CI 74LS393) y su respectivo manejador de buffer (un CI 74LS244). Esta frecuencia va de acuerdo a la frecuencia de guiado sideral necesaria para operar el telescopio, como se explicó en la sección 2.1.3.

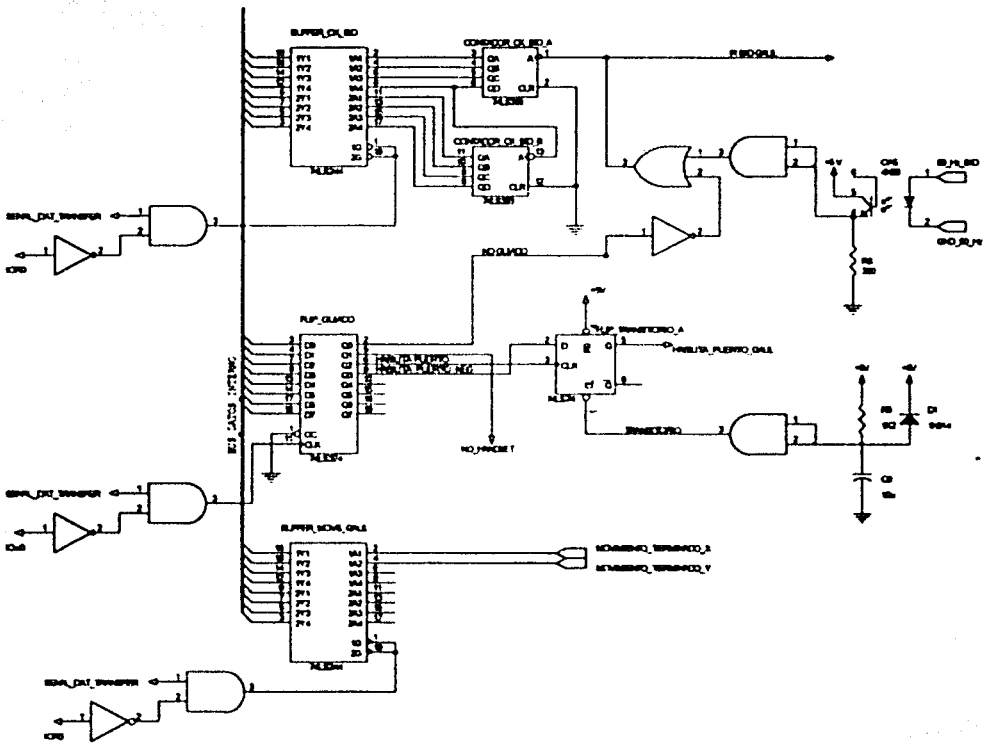


Figura 3.3. Lógica que trata a la frecuencia sideral y a *Galil*.

La transferencia de pulsos provenientes del reloj sideral a la computadora, es anulada al producirse una señal continua de valor "1" lógico cuando la señal NO GUIADO tiene valor de "0" lógico, ya que el sistema arroja la siguiente ecuación:

$$R\_SID\_GALIL = (NO\_GUIADO)' + (90\_HZ\_SID)$$

Debido al tipo de compuertas que utiliza, la electrónica del sistema la tarjeta MAGALI presenta la siguiente ecuación:

$$R\_SID\_GALIL = (NO\_GUIADO)'(90\_Hz\_SID)' + (NO\_GUIADO)'(90\_Hz\_SID)) + (NO\_GUIADO)(90\_Hz\_SID)$$

la cual es equivalente a la ecuación propuesta, con la ventaja de que para esta última se necesitan menos componentes para lograrla. Esto es posible ya que es indistinto que el tren de pulsos de la frecuencia sideral entren o no conjugados al contador.

El proceso de conteo de los pulsos provenientes del reloj sideral es equivalente al descrito en el módulo de movimientos del telescopio, salvo que en este módulo se utiliza el CI 74LS393, que es un contador dual de 4 bits, conectado en cascada para realizar un conteo de 8 bits. La razón de no utilizar un contador de 8 bits en este módulo es que se desperdiciarían varias funciones de éste (como, por ejemplo, la capacidad de contar ascendente o descendentemente), lo que provocaría un mayor consumo de macroceldas en la programación del CPLD.

El envío de las señales de control del guiado del telescopio está supeditado a un *flip-flop* octal de tipo D (el CI 74LS374). Es importante recordar que la diferencia entre el *flipflop* y el *latch* es que, a pesar de ser ambos elementos de memoria, los *flipflops* se cargan por flanco de disparo, mientras que los *latches* se cargan por nivel activo. La finalidad de utilizar el *flip-flop* de tipo D, llamado así debido a su capacidad de transferir "datos" dentro de sí mismo, es dado a su característica de transmitir a su salida el estado de las entradas que posee (siempre que se active la señal de reloj y se mantenga en ese nivel, independientemente de cualquier suceso externo) hasta la llegada de un nuevo

flanco de reloj. Las señales NO GUIADO, NO HANDSET y HABILITA PUERTO se mantienen en el nivel que les ordena el programa de la consola de control, y sólo se actualizan hasta que la señal proveniente de una compuerta AND, en la que se combinan la señal SENAL_DAT_TRANSFER e IOW0, conectada al Reloj del *flip-flop*, vuelva a activarse.

La señal NO GUIADO se utiliza cuando se encuentra operando el controlador manual y la señal NO HANDSET se utiliza cuando el telescopio se encuentra en estado de guiado automático.

La señal HABILITA PUERTO es la encargada de permitir la activación de *Galil* y consta en realidad de dos señales (Q2 y Q3 del *flip-flop*), sólo que se considera como una sola ya que Q3 es la misma señal que Q2. Esta señal tiene la finalidad de que, mientras se inicia el sistema operativo de la computadora, las condiciones iniciales del controlador de motores sean cero, y por ende que el telescopio se encuentre en reposo. De esta manera, se evitan movimientos involuntarios impulsivos de la montura debido a estados lógicos temporales no deseados. Así, *Galil*, y en consecuencia los motores, no son encendidos sino hasta que el programa de la consola lo indica.

Sin embargo, no sólo se debe considerar las condiciones iniciales de *Galil*, sino de la misma computadora, ya que los niveles de las tensiones dentro de la misma no se estabilizan sino hasta pasado un tiempo, transcurrido desde que la fuente de alimentación se ha encendido. Debido a que, por ningún motivo, el telescopio debe moverse cuando se enciende el sistema, se tiene que proteger a la misma computadora de que cree estados lógicos temporales durante su inicialización. Para evitar esto, se ha implementado que la señal HABILITA PUERTO (y su conjugada), activen un *flip-flop* tipo D (el CI 74LS74) cuya señal de borrado (terminal 1) sólo se retira cuando ha transcurrido un lapso de tiempo fijado por un circuito RC. Esta señal de borrado cumple con una función similar que la que suministra la señal RESET DRIVER del *bus* ISA. El circuito RC está

conformado por una resistencia de 1.2kΩ y un capacitor de 10μF, conectados en serie, cuya constante de tiempo es:

$$\tau = RC = (1.2k\Omega)(10\mu F) = 12mseg.$$

El tiempo para que el capacitor esté cargado en su totalidad (más del 98%) es:

$$t = 4\tau = 4(12mseg) = 48\ mseg.$$

Esto implica que, aproximadamente 50 milisegundos después de que se haya encendido la fuente, la señal de borrado (CLR) se desbloqueará permitiendo así que la salida Q_{negada} se utilice para habilitar o no a *Galil*, mediante la señal HABILITA PUERTO, una vez que los estados transitorios de la computadora han desaparecido.

Finalmente, las señales MOVIMIENTO_TERMINADO_X y MOVIMIENTO_TERMINADO_Y son los indicadores que Galil manda cuando los respectivos movimientos de la montura ecuatorial han terminado. Estas señales se denominan así ya que, aunque en la realidad los ejes son el de la Ascensión Recta y la Declinación, el controlador de motores sólo identifica los ejes X y Y. Esto no presenta alguna contradicción ya que, como se vio en la sección 2.1.3, una montura ecuatorial es una montura azimutal, con sus ejes X y Y, a la que se le ha inclinado su eje vertical hasta situarlo paralelo al eje de rotación de la Tierra. Estas señales ingresan al *bus* de datos interno del CPLD cuando son llamadas por la señal IOR3.

3.1.3. LÓGICA PARA LOS SENSORES

El análisis de la electrónica que conforma el módulo de adquisición de datos de los sensores se divide en dos (ver fig. 3.4): la amplificación de las señales provenientes de los sensores y la conversión de estos datos analógicos a un formato digital.

La razón de ser de los sensores es que, durante las observaciones astronómicas, es necesario operar el telescopio dentro de ciertas condiciones climáticas para no dañar los instrumentos. De esta manera, los sensores dentro de la computadora se hacen necesarios para poder vigilar las condiciones en las que opera el equipo. El operador puede entonces vigilar estas mediciones en la consola de control y tomar decisiones al respecto.

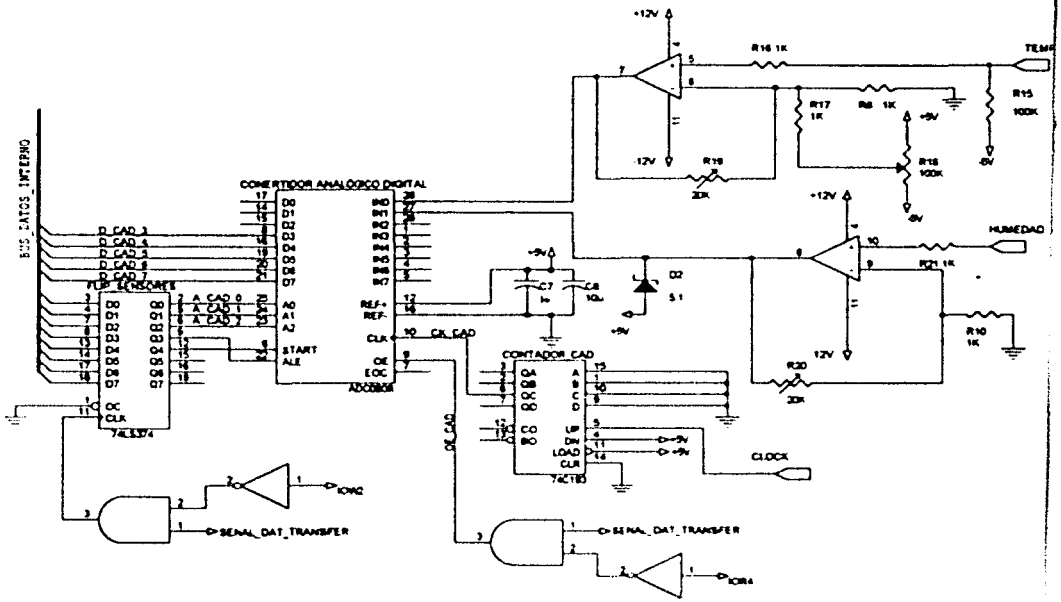


Figura 3.4. Lógica de amplificación (der.) y conversión (izq.) de las señales provenientes de los sensores.

Como las señales provenientes de los sensores son pequeñas señales analógicas, deben ser amplificadas antes de poder ser convertidas y enviadas al *bus* de datos. Esta amplificación se logra mediante amplificadores operacionales integrados en un CI TL074.

Éstos son amplificadores que generan poco ruido, presentando una alta impedancia de entrada y una muy baja impedancia de salida. Estas características hacen que, en estos aspectos, el CI TL074 represente una buena aproximación al amplificador operacional ideal.

La ganancia para la que están implementados estos amplificadores es de 10 y, para lograr este valor de manera exacta, se utilizan resistencias de retroalimentación ajustables (R19 y R20) de 20kΩ para que, mediante un ajuste preciso de sus resistencias, contrarresten el efecto de las tolerancias que pudieran tener las resistencias de 1kΩ (R8 y R10). Esto se consigue al buscar un valor exacto de una ganancia A_o para un amplificador operacional en configuración de amplificador no inversor; es decir, que la configuración en la que la señal de salida esté en fase con la de entrada, pero incrementada. El objetivo de colocar una resistencia ajustable para contrarrestar la tolerancia inherente de las resistencia se observa en la siguiente ecuación:

$$A_o = \frac{V_{salida}}{V_{entrada}} = 1 + \frac{R_{retroalimentación}}{R_{10}} = 1 + \frac{R_{20}}{1k\Omega} = 10$$

Debido a que R_{10} nunca tiene un valor preciso de 1kΩ, y se busca que el cociente del término fraccionario sea exactamente 9, R_{20} debe ajustarse para que sea exactamente 9 veces el valor de R_{10}.

Con objeto de presentar una impedancia de entrada más constante a los sensores, se cuenta con R16 y R21 de 1kΩ conectadas en serie a las terminales negativas de cada amplificador operacional, ya que la impedancia natural de salida del amplificador operacional varía entre los 85Ω y los 300Ω aproximadamente.

Adicionalmente, el amplificador operacional que se encuentra conectado al sensor de temperatura (un LM35 de *National Semiconductor*), cuenta con un divisor de tensión conectado a su terminal negativa, conformado por una resistencia ajustable de 100 kΩ (R18), para provocar una compensación, mediante una tensión de *offset* que se suma a la señal proveniente del sensor. Esto permite al programa de la consola de guiado leer, tanto lecturas positivas, como negativas, del sensor de temperatura, ya que el convertidor analógico digital (el CI ADC0808) sólo trabaja con tensiones positivas. Debido a la compensación, las tensiones negativas que pudieran provenir del sensor de temperatura, son elevadas a tensiones positivas que el convertidor analógico digital puede convertir. El programa de la consola de guiado del telescopio es el que realiza la interpretación de los valores provenientes del convertidor, restándole al valor leído esta tensión de *offset*. Igualmente, la R15 conectada al amplificador operacional asociado al sensor de temperatura, se conecta a una tensión de -5V debido a los requisitos de la configuración indicada por la hoja de especificaciones del LM35 (disponible en *www.national.com*) para obtener un rango completo de temperaturas, teóricamente de -55 a +50°C.

Una vez amplificadas las señales, éstas ingresan al Convertidor Analógico/Digital (el CI ADC0808). Este convertidor de 8 bits, que utiliza el método de aproximaciones sucesivas, cuenta con 8 canales para las distintas conversiones. El método de aproximaciones sucesivas realiza *n* iteraciones para un convertidor de *n* bits. En este caso, el ADC realiza 8 iteraciones para aproximarse a la tensión de entrada. Este método está diseñado para sistemas de conversión radiométricos, también llamados sistemas de conversión de doble pendiente, en el cual el ADC compara la entrada analógica en cada iteración, cuyas separaciones están determinadas por una escala fijada por dos tensiones de referencia (0V y +5V en este caso), mientras que la variable a medir se expresa como un porcentaje de esta escala.

El canal se selecciona mediante un multiplexor que trae integrado el ADC. Sus entradas (A0, A1 y A2) están conectadas a un *flip-flop* octal 74LS374, controlado por la señales IOW2 y SENAL_DAT_TRANSFER, que cumple con la misma finalidad que el *flip-flop* del módulo de control de las habilitaciones. Las salidas del multiplexor (D_CAD_3 al D_CAD_7, que son los bits más significativos del CAD), son del tipo *Tri-state*, significando por esto que sus posibles salidas son alto, bajo y alta impedancia, por lo que pueden conectarse directamente al *bus* de datos interno del CPLD, controladas por la señales IOR4 y SENAL_DAT_TRANSFER.

Adicionalmente, se utiliza un CI 74C192, que es un contador bidireccional síncrono y programable, que se utiliza para dividir la frecuencia de reloj proveniente del *bus* ISA entre 8. La razón de esto es debido a que la frecuencia máxima a la que puede operar el ADC, por sus características inherentes, es de 1.28 MHz, mientras que la frecuencia del reloj del *bus* ISA es de 4.77 MHz. Mediante una división entre 8 de esta frecuencia, se tiene una frecuencia de 596.3 kHz, que es muy cercana al valor típico de 640 kHz que sugiere la hoja de especificaciones del convertidor. La división entre 8 se logra insertando la señal de reloj del *bus* ISA en la terminal 5 (UP) del CI 74C193, lo que además genera un conteo ascendente, utilizando como salida el tercer bit más significativo del contador (QC). Así, en la terminal 6 del contador se obtiene un reloj cuyos estados lógicos alto o bajo tienen una duración de 4 ciclos del reloj del *bus* cada uno, señal que se utiliza como reloj en el ADC.

Cabe hacer notar que el ADC será el único circuito de la electrónica de la tarjeta MAGALI que no será integrada dentro del CPLD. Esto debido, principalmente, al gran desempeño que se obtiene con el uso de un ADC especialmente desarrollado y especializado en sus funciones, como es el caso del ADC0808, y a la imposibilidad de ingresar señales analógicas en el EPM7064LC68.

3.1.4 LÓGICA PARA EL CONTROL DE HABILITACIONES

La lógica para el control de las habilitaciones tiene que ver con el manejo de las señales de control que provienen del *bus* ISA y es una adaptación de la tarjeta de desarrollo PCL-750. Una de las principales diferencias con la lógica de la tarjeta de desarrollo PCL-750 es que, en la lógica que demanda este trabajo, sólo se opera con las señales IOW e IOR para dirigir la información entre la computadora y la interfaz. En el Apéndice D se muestra la programación para el CPLD en el que es posible, adicionalmente, realizar el *direccionamiento* a memoria mediante las señales MEMW y MEMR.

Las señales de control que provienen del *bus* ISA, como se explicó en la sección 3.1.2, son IOR e IOW y se muestran en la fig. 3.5. Como todas las señales que se dirigen al *bus* ISA, éstas deben pasar por un buffer para mejorar el rendimiento del *bus*. Esto se debe a que una interfaz que no utilizara buffers para la comunicación entre puertos dependería para su éxito, principalmente, de las capacidades de transferencia de información del puerto con el que se comunica. Una vez que las señales han pasado por el *bus*, con fines de notación, a estas señales se les denomina el prefijo BUFFER_, por lo que las señales se convierten en: BUFFER_IO_READ y BUFFER_IO_WRITE.

Las primeras tres señales del *bus* de direcciones A0, A1 y A2, son las utilizadas para seleccionar las señales IOR0 hasta IOR7 (aunque en este interfaz sólo se trabaja con las señales IOR0 hasta IOR4 y con las señales IOW0 e IOW1) mediante el demultiplexor DEMUX_IOR, que es activado por la señal HABILITA_DECOD_DIR que proviene del módulo de las decodificaciones y por la señal BUFFER_IO_READ. Las señales IOR0 hasta IOR4 se utilizan para habilitar y deshabilitar los buffers que permiten el acceso a los datos que buscan ingresar al *bus* de datos interno del CPLD.

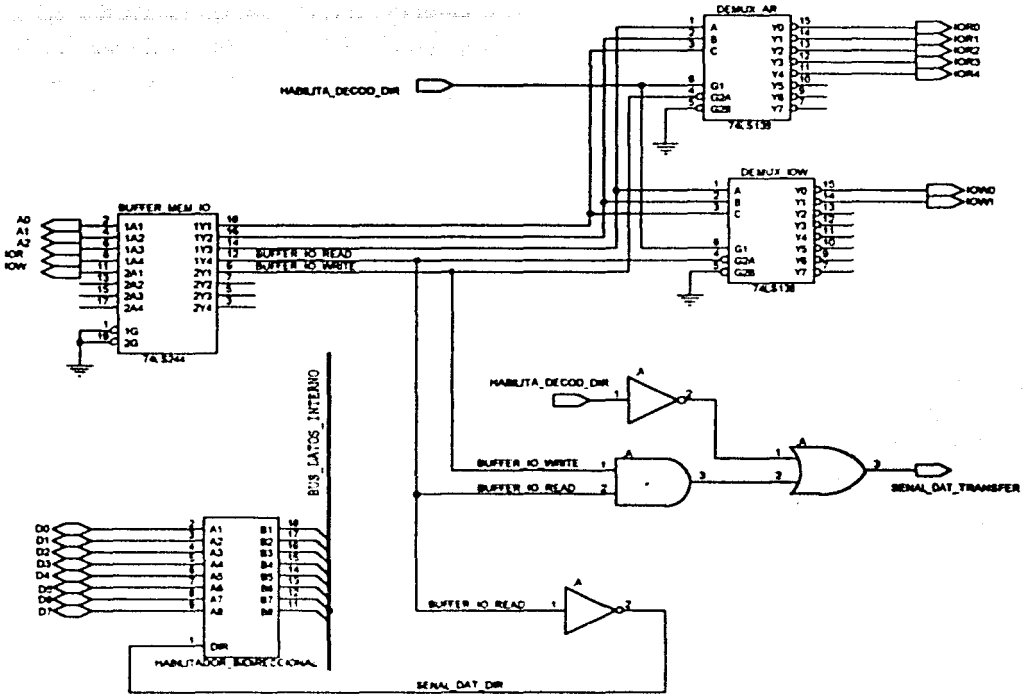


Figura 3.5. Lógica utilizada para manejar las señales de control que provienen del *bus* ISA.

Igualmente, las mismas tres primeras señales del *bus* de direcciones se utilizan para seleccionar las señales IOW0 hasta IOW7 mediante el demultiplexor DEMUX_IOW, que es activado por la señal HABILITA_DECOD_DIR y por la señal BUFFER_IO_WRITE. Las señales IOW0 e IOW1 se utilizan para habilitar y deshabilitar a los buffers, que a su vez permiten el envío de datos desde el *bus* de datos interno del CPLD.

La habilitación del *bus* de datos interno del CPLD (BUS_DATOS_INTERNO), es controlado para permitir o no la transferencia datos entre el *bus* ISA y la interfaz, independientemente de cual sea su dirección. Esto se logra conectando, como se vio en los módulos anteriores, las habilitaciones de los buffers a la salida de una compuerta AND que combina a la respectiva señal de habilitación del buffer (IOWx o IORx) y la señal SENAL_DATA_TRANSFER, que es a la vez función de las señales BUFFER_IO_WRITE, BUFFER_IO_READ y HABILITA_DECOD_DIR, (1 habilita y 0 deshabilita) se muestra en la tabla 3.2.

BUFFER IO WRITE	BUFFER IO READ	HABILITA DECOD DIR	SENAL DAT TRANSFER
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Tabla 3.2. Habilitación del *bus* de datos interno del CPLD.

Para los fines de la programación del CPLD, se programará la tabla de verdad anterior, en vez de programar el arreglo de compuertas lógicas, para que, al ser éste un arreglo de distintos tipos de compuertas, el compilador Max+Plus II tenga la opción de adaptar mejor la lógica de la tabla a la arquitectura interna de las macroceldas del CPLD.

La dirección del *bus* de datos interno es controlado para transferir datos desde el *bus* ISA hacia la interfaz, o bien, desde la interfaz al *bus* ISA mediante las señal negada de BUFFER_IO_READ. De esta manera, cuando la señal SENAL_DAT_DIR es 1, se envía información desde la interfaz hacia la computadora mediante el *bus* de datos (lectura de datos) y, cuando es 0, se envía información desde la computadora hacia la interfaz. La tarjeta de desarrollo PCL-750 utiliza para este fin el buffer bidireccional 74LS245,

mientras que, en este trabajo, la lógica de control antes desarrollada será programada en el CPLD.

3.1.5. LÓGICA PARA LAS DECODIFICACIONES

Los principios para la lógica de las decodificaciones de puertos de entrada/salida se explicó en la sección 2.1.5. Siguiendo los mismos conceptos, y retomando el razonamiento de decodificación de la fig. 2.16, donde la dirección a decodificar es seleccionada por interruptores, se puede estructurar la electrónica que se muestra en la fig.3.6:

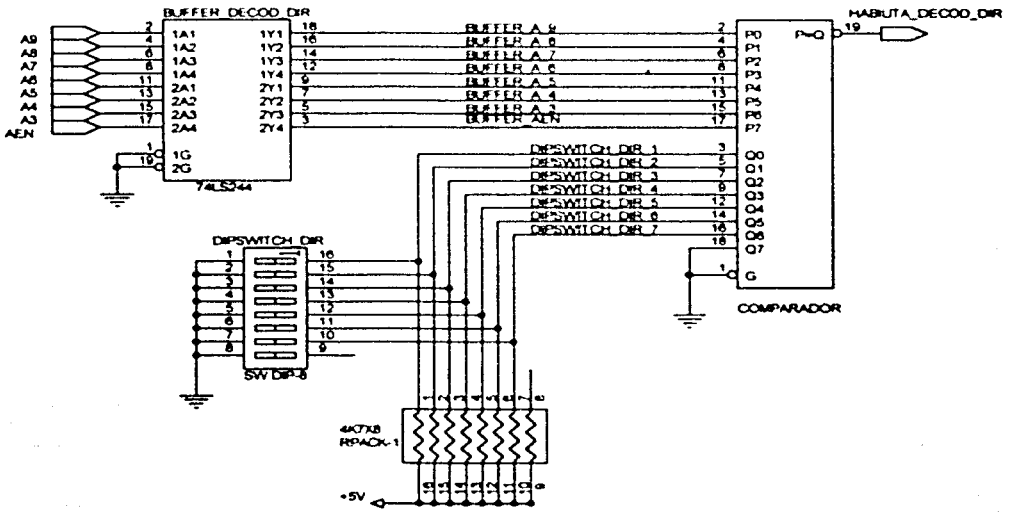


Figura 3.6. Lógica empleada para la decodificación de puertos.

Sin embargo, para fines de programación resulta más eficiente programar una sentencia "*if*", que compare la igualdad entre las señales provenientes del dipswitch y las señales provenientes del buffer con las señales para la decodificación, que utilizar una macrofunción de comparador. Por ejemplo, para la decodificación de direcciones, es preferible programar el diagrama de flujo mostrado en la fig. 3.7. Cabe mencionar que, para implementar físicamente el circuito de la fig. 3.6, se utilizaría un comparador 74LS688 en la tarjeta PCL-750, el cual debería ser habilitado con una señal de nivel bajo. Sin embargo, a nivel de programación del CPLD, esta señal se habilita con un nivel alto, como se muestra en la fig. 3.7.

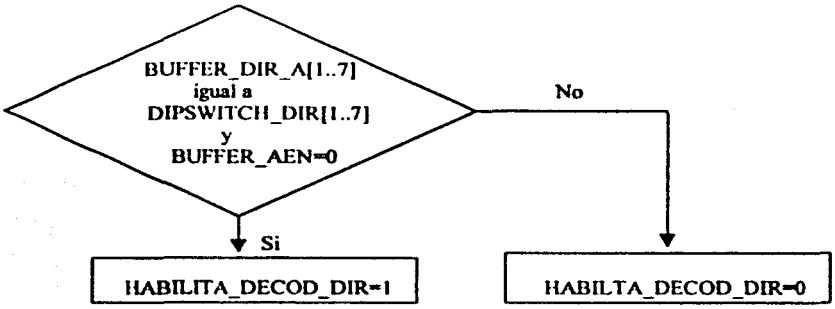


Figura 3.7. Sentencia "*if*" para la decodificación de puertos.

De esta manera, la lógica de este módulo se enfoca más a la programación de sentencias lógicas, que a la implementación de macrofunciones en el CPLD, como fue el caso de la lógica de los módulos anteriores.

Una vez analizada la lógica y los procedimientos con que se realiza la interfaz, entre la computadora de la consola del telescopio y sus periféricos, se puede proceder a

integrar todos estos puntos en la programación del CPLD de la tarjeta, junto con aquellos dispositivos electrónicos con los que operará para realizar la interfaz.

IV
DISEÑO DE LA INTERFAZ

De acuerdo con lo planteado en los capítulos anteriores, se implementa la programación del CPLD utilizado para este proyecto mediante el desarrollo de distintos módulos de programación, que son posteriormente integrados en un solo programa. Su simulación, individual y en conjunto, garantiza su correcto funcionamiento ante las distintas condiciones de operación que puedan acontecer. Finalmente, mediante el tratamiento de las especificaciones de los distintos componentes que integran la interfaz, se expone el diseño de un posible circuito impreso de la interfaz electrónica embebida.

4.1. PROGRAMACIÓN DEL CPLD

Siguiendo la lógica de diseño de la interfaz desarrollada en la sección 3, la programación del CPLD se realiza en cinco módulos, los cuales son probados y analizados individualmente, antes de ser conjuntados en una sola lógica para el microcontrolador.

De esta manera, la programación del CPLD se fragmenta en la siguiente forma:

a) Módulo de los movimientos de AR y DEC. En este módulo se programa la lógica desarrollada en la sección 3.1.1.
b) Módulo del reloj sideral. En este módulo se programa la lógica desarrollada en la sección 3.1.2.
c) Módulo de los sensores. En este módulo se programa la lógica desarrollada en la sección 3.1.3.
d) Módulo de las decodificaciones. En este módulo se programa la lógica desarrollada en la sección 3.1.4.
e) Módulo del control de habilitaciones. En este módulo se programa la lógica desarrollada en la sección 3.1.5.

Todas estos módulos interactúan entre sí cuando se integran en un solo programa. La forma en que se relacionan estos módulos se presenta en la fig. 4.1.

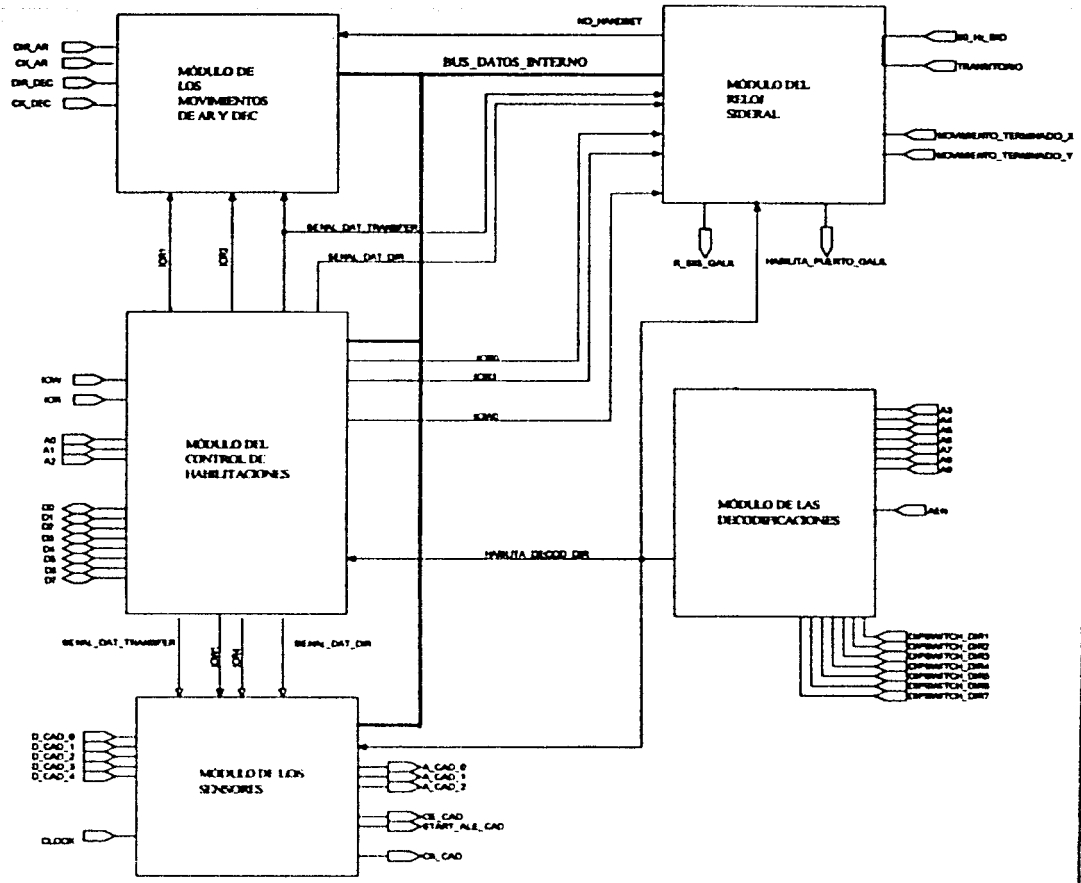


Figura 4.1. Relación entre los módulos programados en el CPLD integrados en la interfaz.

Evidentemente, varias señales de entrada/salida que ingresan o proceden de los distintos módulos, no son salidas globales del sistema. Estas señales son: HABILITA_DECOD_DIR, NO_HANDSET, IOR0, IOR1, IOR2, IOR3, IOR4, IOW0, IOW1, SENAL_DAT_DIR, SENAL_DAT_TRANSFER y BUS_DATOS_INTERNO (0 al 7). Estas señales se convierten en nodos en el programa que integra a los módulos. Esto es debido a que un nodo, en la programación de Max+Plus II, representa a un conductor que transporta a una señal que viaja entre distintos componentes o macroceldas.

Los programas están compilados en la versión 9.4 de Max+Plus II de Altera. El lenguaje de programación utilizado es AHDL (*Altera Hardware Description Language*) y los comentarios a la programación son aquellos que están delimitados por el símbolo "%". Cabe señalar que sólo se presentan las simulaciones más representativas del funcionamiento de los módulos, para demostrar su buen funcionamiento, siendo que se realizaron las simulaciones de todos los casos posibles antes de dar por terminada la programación de un módulo.

Para fines de notación, las señales que entran o salen del microcontrolador, es decir, aquellas que tienen una terminal asignada en el CPLD, se identifican al estar escritas en **negritas**, mientras que, las señales que operan como nodos entre los diferentes módulos se encuentran escritas en *cursiva*.

4.1.1. MÓDULO DE LOS MOVIMIENTOS DE AR Y DEC

Entradas: **DIR_AR, CK_AR, DIR_DEC, CK DEC**
NO_HANDSET,
SENAL_DAT_TRANSFER, IOR1, IOR2

Salidas: *BUS_DATOS_INTERNO* **(0 al 7)**

- *DESARROLLO EN AHDL*

```
%MODULO DE LOS MOVIMIENTOS DEL TELESCOPIO%

OPTIONS BIT0 = ANY;

INCLUDE "8COUNT";
INCLUDE "74244";
INCLUDE "7408";
SUBDESIGN MODULO_MOVIMIENTOS
(
%DEL MODULO%
DIR_AR,CK_AR:INPUT;
DIR_DEC,CK_DEC:INPUT;

%EXTERNAS%
NO_HANDSET:INPUT;                          %EN EL GLOBAL ES SEÑAL INTERNA
(NODOS)%
SENAL_DAT_TRANSFER,IOR1,IOR2:INPUT;        %EN EL GLOBAL ES SEÑAL INTERNA
(NODOS)%

D[7..0]:OUTPUT;                            %ESTA SEÑAL ES BIDIRECCIONAL EN EL
GLOBAL%
)

VARIABLE
CONTADOR_AR,CONTADOR_DEC:8COUNT;
BUFFER_AR,BUFFER_DEC:74244;
BUS_DATOS_INTERNO[7..0]:TRI_STATE_NODE;
AND_CK_AR,AND_DIR_AR:7408;
AND_CK_DEC,AND_DIR_DEC:7408;

BEGIN

%PARTE DE AR%

AND_CK_AR.2=CK_AR;
AND_CK_AR.3=CK_AR;
AND_DIR_AR.2=DIR_AR;
AND_DIR_AR.3=DIR_AR;

CONTADOR_AR.CLK=AND_CK_AR.1;
CONTADOR_AR.DNUP=AND_DIR_AR.1;
CONTADOR_AR.GN=NO_HANDSET;

BUFFER_AR.1A[1]=CONTADOR_AR.QA;
BUFFER_AR.1A[2]=CONTADOR_AR.QB;
BUFFER_AR.1A[3]=CONTADOR_AR.QC;
BUFFER_AR.1A[4]=CONTADOR_AR.QD;
```

```
BUFFER_AR.2A[1]=CONTADOR_AR.QE;
BUFFER_AR.2A[2]=CONTADOR_AR.QF;
BUFFER_AR.2A[3]=CONTADOR_AR.QG;
BUFFER_AR.2A[4]=CONTADOR_AR.QH;

%PARTE DE DEC%
AND_CK_DEC.2=CK_DEC;
AND_CK_DEC.3=CK_DEC;
AND_DIR_DEC.2=DIR_DEC;
AND_DIR_DEC.3=DIR_DEC;

CONTADOR_DEC.CLK=AND_CK_DEC.1;
CONTADOR_DEC.UP=AND_DIR_DEC.1;
CONTADOR_DEC.GN=NO_HANDSET;

BUFFER_DEC.1A[1]=CONTADOR_DEC.QA;
BUFFER_DEC.1A[2]=CONTADOR_DEC.QB;
BUFFER_DEC.1A[3]=CONTADOR_DEC.QC;
BUFFER_DEC.1A[4]=CONTADOR_DEC.QD;
BUFFER_DEC.2A[1]=CONTADOR_DEC.QE;
BUFFER_DEC.2A[2]=CONTADOR_DEC.QF;
BUFFER_DEC.2A[3]=CONTADOR_DEC.QG;
BUFFER_DEC.2A[4]=CONTADOR_DEC.QH;

%PARTE DEL BUS%

BUS_DATOS_INTERNO[0]=TRI(BUFFER_DEC.1Y[1],(SENAL_DAT_TRANSFER & !IOR2));
BUS_DATOS_INTERNO[1]=TRI(BUFFER_DEC.1Y[2],(SENAL_DAT_TRANSFER & !IOR2));
BUS_DATOS_INTERNO[2]=TRI(BUFFER_DEC.1Y[3],(SENAL_DAT_TRANSFER& ! IOR2));
BUS_DATOS_INTERNO[3]=TRI(BUFFER_DEC.1Y[4],(SENAL_DAT_TRANSFER & !IOR2));
BUS_DATOS_INTERNO[4]=TRI(BUFFER_DEC.2Y[1],(SENAL_DAT_TRANSFER & !IOR2));
BUS_DATOS_INTERNO[5]=TRI(BUFFER_DEC.2Y[2],(SENAL_DAT_TRANSFER & !IOR2));
BUS_DATOS_INTERNO[6]=TRI(BUFFER_DEC.2Y[3],(SENAL_DAT_TRANSFER & !IOR2));
BUS_DATOS_INTERNO[7]=TRI(BUFFER_DEC.2Y[4],(SENAL_DAT_TRANSFER & !IOR2));

BUS_DATOS_INTERNO[0]=TRI(BUFFER_AR.1Y[1],(SENAL_DAT_TRANSFER & !IOR1));
BUS_DATOS_INTERNO[1]=TRI(BUFFER_AR.1Y[2],(SENAL_DAT_TRANSFER & !IOR1));
BUS_DATOS_INTERNO[2]=TRI(BUFFER_AR.1Y[3],(SENAL_DAT_TRANSFER & !IOR1));
BUS_DATOS_INTERNO[3]=TRI(BUFFER_AR.1Y[4],(SENAL_DAT_TRANSFER & !IOR1));
BUS_DATOS_INTERNO[4]=TRI(BUFFER_AR.2Y[1],(SENAL_DAT_TRANSFER & !IOR1));
BUS_DATOS_INTERNO[5]=TRI(BUFFER_AR.2Y[2],(SENAL_DAT_TRANSFER & !IOR1));
BUS_DATOS_INTERNO[6]=TRI(BUFFER_AR.2Y[3],(SENAL_DAT_TRANSFER & !IOR1));
BUS_DATOS_INTERNO[7]=TRI(BUFFER_AR.2Y[4],(SENAL_DAT_TRANSFER & !IOR1));

D[]=BUS_DATOS_INTERNO[];  %PARTE PARA COMPROBAR LAS SALIDAS SOLAMENTE.
                          NO VA EN EL GLOBAL%
END;
```

- *SIMULACIÓN Y ANÁLISIS*

Las pruebas realizadas al módulo de los movimientos del telescopio consisten, básicamente, en comprobar que el conteo ascendente y descendente de los pulsos provenientes del controlador manual del telescopio sea correcto, y de que la señal que ingresa al *bus* de datos interno del CPLD esté correctamente habilitada por la señal IOR1, para los movimientos de ascensión recta, y por la señal IOR2, para los movimientos de declinación de la montura ecuatorial.

En la fig. 4.2 se muestra la primera simulación, en la que se indica el caso cuando se desea leer el movimiento de ascensión recta. Esto se logra ya que IOR1 se habilita al tener un valor de IOR1=0, mientras que se permite la transferencia de datos con SENAL_DAT_TRANSFER=1. La señal de NO_HANDSET no está activada, y el conteo de pulsos que se obtienen en el *bus* de datos es ascendente ya que la señal que manda el controlador manual del telescopio es DIR_AR=0.

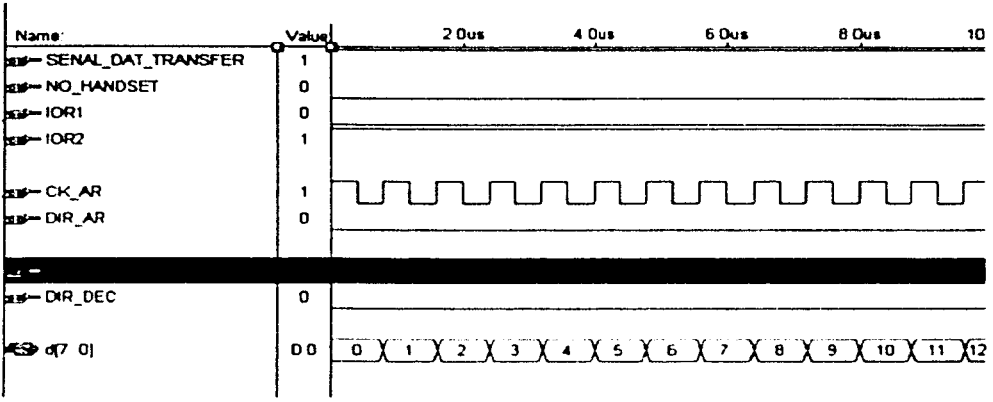


Figura 4.2. Simulación 1 del módulo de los movimientos.

En la fig. 4.3 se muestra la segunda simulación, en la que se indica el mismo caso cuando se desea leer el movimiento de ascensión recta, pero ahora cuando la señal proveniente del controlador manual del telescopio indica que el conteo es descendente al ser DIR_AR=1. Esto se logra ya que IOR1 sigue teniendo un valor de IOR1=0, lo que implica que continúa habilitada, y SENAL_DAT_TRANSFER permanece en "1".

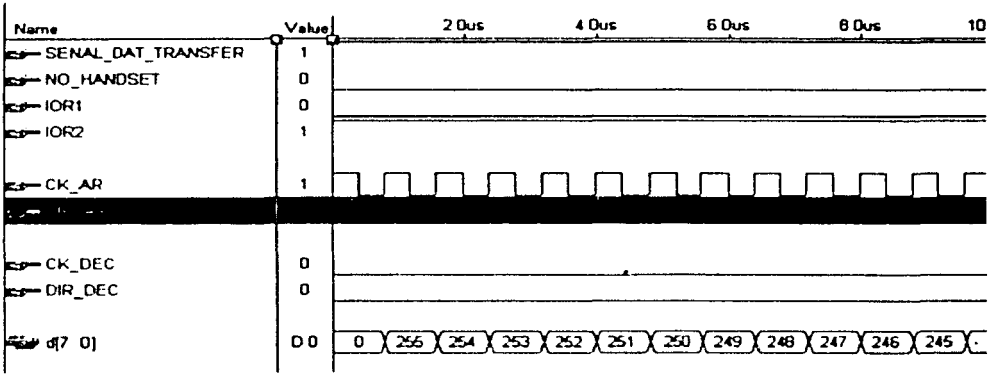


Figura 4.3. Simulación 2 del módulo de los movimientos.

En la fig. 4.4 se muestra la tercera simulación, en la que se indica el caso cuando se desea leer el movimiento de declinación. Esto se logra ya que IOR2 se habilita al tener un valor de IOR2=0 (IOR1 cambia de valor a IOR1=1), mientras que se permite la transferencia de datos con SENAL_DAT_TRANSFER=1. La señal de NO_HANDSET no está activada, y el conteo de pulsos que se obtienen en el *bus* de datos es ascendente ya que la señal que manda el controlador manual del telescopio es DIR_DEC=0.

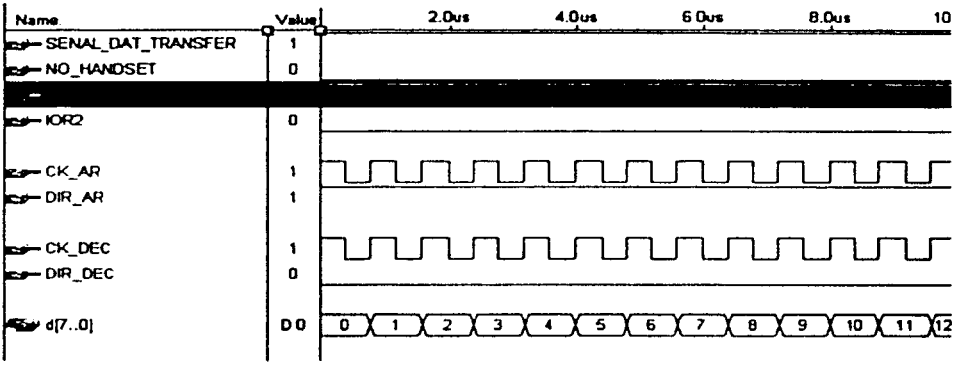


Figura 4.4. Simulación 3 del módulo de los movimientos.

Finalmente, en la fig. 4.5, se muestra la quinta simulación, en la que se indica el caso cuando la señal de NO_HANDSET se activa al tomar el valor de "1". Esto provoca que, a pesar de que estén ingresando pulsos en el movimiento de declinación, y que éste canal se esté leyendo al estar IOR2=0, la salida por el *bus* de datos es cero ya que el conteo de pulsos está deshabilitado.

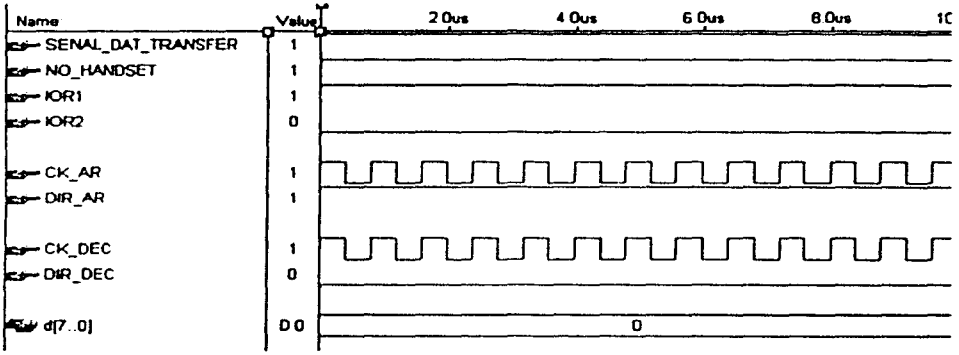


Figura 4.5. Simulación 4 del módulo de los movimientos.

4.1.2. MÓDULO DEL RELOJ SIDERAL

Entradas: **90_Hz_SID**
TRANSITORIO
MOVIMIENTO_TERMINADO_X, MOVIMIENTO_TERMINADO_Y
SENAL_DAT_TRANSFER , *SENAL_DAT_DIR_ HABILITA_DECOD_DIR*
IOR0, IOR3
IOW0

Salidas: **HABILITA_PUERTO_GALIL**
R_SID_GALIL
NO_HANDSET

Entradas/Salidas: *BUS_DATOS_INTERNO*

- *DESARROLLO EN AHDL*

```
%MODULO DEL RELOJ SIDERAL%

OPTIONS BIT0=ANY;
INCLUDE "74244";
INCLUDE "74393";
INCLUDE "7408";
INCLUDE "7474";
INCLUDE "74374";

SUBDESIGN MODULO_RELOJ_SIDERAL
(
90_HZ_SID :INPUT;
R_SID_GALIL :OUTPUT;

IOR0,IOR3 :INPUT;                          %EN EL GLOBAL SON NODOS%
SENAL_DAT_TRANSFER,  HABILITA_DECOD_DIR :INPUT;    %EN EL GLOBAL ES NODO%

TRANSITORIO :INPUT;
HABILITA_PUERTO_GALIL :OUTPUT;
MOVIMIENTO_TERMINADO_X,MOVIMIENTO_TERMINADO_Y :INPUT;

%PARTE DEL LATCH%

IOW0 :INPUT;                               %NODO EN EL GLOBAL%
SENAL_DAT_DIR :INPUT;                      %NODO EN EL GLOBAL%
NO_HANDSET :OUTPUT;                        %NODO EN EL GLOBAL%

D[7..0] :BIDIR;                    %COMUN EN EL GLOBAL%
)
VARIABLE

BUFFER_CK_SID :74244;
CONTADOR_CK_SID :74393;
AND_90_HZ_SID :7408;
AND_TRANSITORIO :7408;
BUS_DATOS_INTERNO[7..0] :TRI_STATE_NODE;
FLIP_TRANSITORIO :7474;
BUFFER_MOVS_GALIL :74244;

NO_GUIADO :NODE;
HABILITA_PUERTO,HABILITA_PUERTO_NEG :NODE;
PERMITE_TRANSFERENCIA :NODE;
FLIP_GUIADO :74374;
PRE_NO_GUIADO,PRE_NO_HANDSET,PRE_HABILITA_PUERTO,
PRE_HABILITA_PUERTO_NEG :TRI_STATE_NODE;
```

```
BEGIN

AND_90_HZ_SID.2=90_HZ_SID;
AND_90_HZ_SID.3=90_HZ_SID;
AND_TRANSITORIO.2=TRANSITORIO;
AND_TRANSITORIO.3=TRANSITORIO;

%PARTE DE MOVS TERMINADOS%

BUFFER_MOVS_GALIL.1A[1]=MOVIMIENTO_TERMINADO_X;
BUFFER_MOVS_GALIL.1A[2]=MOVIMIENTO_TERMINADO_Y;
BUFFER_MOVS_GALIL.1A[3]=GND;
BUFFER_MOVS_GALIL.1A[4]=GND;
BUFFER_MOVS_GALIL.2A[1]=GND;
BUFFER_MOVS_GALIL.2A[2]=GND;
BUFFER_MOVS_GALIL.2A[3]=GND;
BUFFER_MOVS_GALIL.2A[4]=GND;

BUS_DATOS_INTERNO[0]=TRI(BUFFER_MOVS_GALIL.1Y[1],(SENAL_DAT_TRANSFER &
!IOR3));
BUS_DATOS_INTERNO[1]=TRI(BUFFER_MOVS_GALIL.1Y[2],(SENAL_DAT_TRANSFER &
!IOR3));
BUS_DATOS_INTERNO[2]=TRI(BUFFER_MOVS_GALIL.1Y[3],(SENAL_DAT_TRANSFER &
!IOR3));
BUS_DATOS_INTERNO[3]=TRI(BUFFER_MOVS_GALIL.1Y[4],(SENAL_DAT_TRANSFER &
!IOR3));
BUS_DATOS_INTERNO[4]=TRI(BUFFER_MOVS_GALIL.2Y[1],(SENAL_DAT_TRANSFER &
!IOR3));
BUS_DATOS_INTERNO[5]=TRI(BUFFER_MOVS_GALIL.2Y[2],(SENAL_DAT_TRANSFER &
!IOR3));
BUS_DATOS_INTERNO[6]=TRI(BUFFER_MOVS_GALIL.2Y[3],(SENAL_DAT_TRANSFER &
!IOR3));
BUS_DATOS_INTERNO[7]=TRI(BUFFER_MOVS_GALIL.2Y[4],(SENAL_DAT_TRANSFER &
!IOR3));

%PARTE DEL TRANSITORIO%

HABILITA_PUERTO_GALIL=FLIP_TRANSITORIO.1Q;
FLIP_TRANSITORIO.1CLRN=AND_TRANSITORIO.1;
FLIP_TRANSITORIO.1CLK=HABILITA_PUERTO;
FLIP_TRANSITORIO.1D=HABILITA_PUERTO_NEG;

%PARTE DEL RELOJ SIDERAL%

R_SID_GALIL=(AND_90_HZ_SID.1 OR !NO_GUIADO);

CONTADOR_CK_SID.A1=R_SID_GALIL;
CONTADOR_CK_SID.A2=CONTADOR_CK_SID.Q1D;

BUFFER_CK_SID.1A[1]=CONTADOR_CK_SID.Q1A;
```

```
BUFFER_CK_SID.1A[2]=CONTADOR_CK_SID.Q1B;
BUFFER_CK_SID.1A[3]=CONTADOR_CK_SID.Q1C;
BUFFER_CK_SID.1A[4]=CONTADOR_CK_SID.Q1D;
BUFFER_CK_SID.2A[1]=CONTADOR_CK_SID.Q2A;
BUFFER_CK_SID.2A[2]=CONTADOR_CK_SID.Q2B;
BUFFER_CK_SID.2A[3]=CONTADOR_CK_SID.Q2C;
BUFFER_CK_SID.2A[4]=CONTADOR_CK_SID.Q2D;

BUS_DATOS_INTERNO[0]=TRI(BUFFER_CK_SID.1Y[1],(SENAL_DAT_TRANSFER & !IOR0));
BUS_DATOS_INTERNO[1]=TRI(BUFFER_CK_SID.1Y[2],(SENAL_DAT_TRANSFER & !IOR0));
BUS_DATOS_INTERNO[2]=TRI(BUFFER_CK_SID.1Y[3],(SENAL_DAT_TRANSFER & !IOR0));
BUS_DATOS_INTERNO[3]=TRI(BUFFER_CK_SID.1Y[4],(SENAL_DAT_TRANSFER & !IOR0));
BUS_DATOS_INTERNO[4]=TRI(BUFFER_CK_SID.2Y[1],(SENAL_DAT_TRANSFER & !IOR0));
BUS_DATOS_INTERNO[5]=TRI(BUFFER_CK_SID.2Y[2],(SENAL_DAT_TRANSFER & !IOR0));
BUS_DATOS_INTERNO[6]=TRI(BUFFER_CK_SID.2Y[3],(SENAL_DAT_TRANSFER & !IOR0));
BUS_DATOS_INTERNO[7]=TRI(BUFFER_CK_SID.2Y[4],(SENAL_DAT_TRANSFER & !IOR0));

%PARTE DEL FLIP%

PERMITE_TRANSFERENCIA= (!SENAL_DAT_DIR & !IOW0);

PRE_NO_GUIADO=TRI(D[0],PERMITE_TRANSFERENCIA);
PRE_NO_HANDSET=TRI(D[1],PERMITE_TRANSFERENCIA);
PRE_HABILITA_PUERTO=TRI(D[2],PERMITE_TRANSFERENCIA);
PRE_HABILITA_PUERTO_NEG=TRI(D[3],PERMITE_TRANSFERENCIA);

FLIP_GUIADO.D[1]=PRE_NO_GUIADO;
FLIP_GUIADO.D[2]=PRE_NO_HANDSET;
FLIP_GUIADO.D[3]=PRE_HABILITA_PUERTO;
FLIP_GUIADO.D[4]=PRE_HABILITA_PUERTO_NEG;
FLIP_GUIADO.D[5]=GND;
FLIP_GUIADO.D[6]=GND;
FLIP_GUIADO.D[7]=GND;
FLIP_GUIADO.D[8]=GND;

NO_GUIADO=FLIP_GUIADO.Q[1];
NO_HANDSET=FLIP_GUIADO.Q[2];
HABILITA_PUERTO=FLIP_GUIADO.Q[3];
HABILITA_PUERTO_NEG=FLIP_GUIADO.Q[4];

FLIP_GUIADO.CLK= (SENAL_DAT_TRANSFER & !IOW0);

%PARTE COMUN%

D[0]=TRI(BUS_DATOS_INTERNO[0],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[1]=TRI(BUS_DATOS_INTERNO[1],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[2]=TRI(BUS_DATOS_INTERNO[2],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[3]=TRI(BUS_DATOS_INTERNO[3],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[4]=TRI(BUS_DATOS_INTERNO[4],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[5]=TRI(BUS_DATOS_INTERNO[5],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
```

```
D[6]=TRI(BUS_DATOS_INTERNO[6],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[7]=TRI(BUS_DATOS_INTERNO[7],SENAL_DAT_DIR & HABILITA_DECOD_DIR);

END;
```

- *SIMULACIÓN Y ANÁLISIS*

Las pruebas realizadas al módulo del reloj sideral consisten, básicamente, en comprobar que ingresen al *bus* de datos, cuando son llamados, el conteo del reloj sideral (con la activación de la señal IOR=0) y los indicadores de terminación de movimientos provenientes del controlador de motores *Galil* (con la activación de la señal IOR3=0). Igualmente, se comprueba que, cuando se escribe desde el *bus* y la señal IOW0 se habilita, se activen las señales NO_HANDSET y HABILITA_PUERTO_GALIL. Las entradas 90_HZ_SID y TRANSITORIO se consideran fijas para todas las simulaciones.

En la fig. 4.6 se muestra la primera simulación, en la que se indica el caso cuando se desea leer si existen indicadores de terminación de movimientos provenientes de *Galil*. En esta simulación, MOVIMIENTO_TERMINADO_X=0 y MOVIMIENTO_TERMINADO_Y=1. Estas señales ingresan al *bus* de datos cuando IOR3=0 y cuando SENAL_DAT_DIR=1, lo que indica que la información se dirige desde la computadora hacia la interfaz. SENAL_DAT_TRANSFER=1 por lo que se permite la transferencia de datos. En el *bus* de datos aparece el valor binario 00000010, ya que D[0]=0 (MOVIMIENTO_TERMINADO_X) y D[1]=1 (MOVIMIENTO_TERMINADO_Y). Las entradas 90_HZ_SID y TRANSITORIO se consideran fijas para todas las simulaciones.

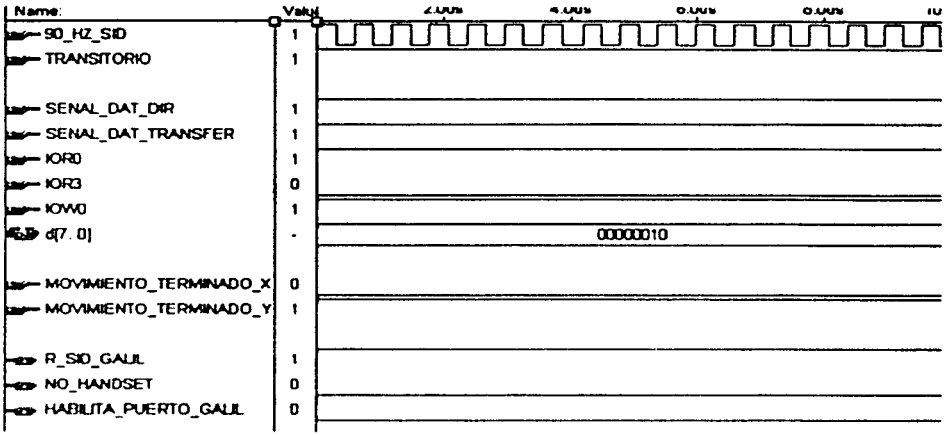


Figura 4.6. Simulación 1 del módulo del reloj sideral.

En la fig. 4.7 se muestra la segunda simulación, en la que se indica el caso cuando se desea leer el conteo del reloj sideral. Para lograr esto, la señal que activa el ingreso de esta señal al *bus* de datos es IOR=0. Igualmente, la señal N0_GUIADO (no mostrada en la simulación ya que es un nodo), había sido activada en "1" mediante la escritura de D[0]=1, con IOR0=0, para habilitar la transferencia de los pulsos provenientes del reloj sideral.

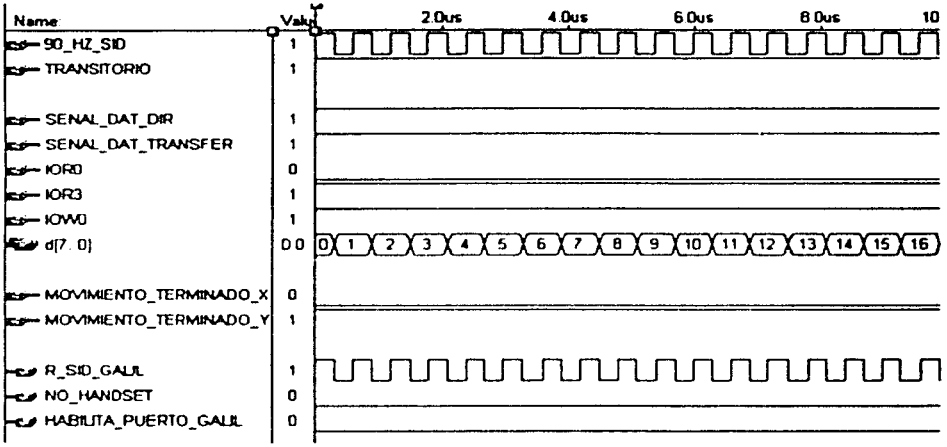


Figura 4.7. Simulación 2 del módulo del reloj sideral.

En la fig. 4.8 se muestra la tercera simulación, en la que se indica el caso cuando se desea encender las señales NO_HANDSET y HABILITA_PUERTO_GALIL. Para lograr esto, IOW0 se activa en valor bajo (en la simulación se muestra como pulsos, ya que el flip-flop que activa esta señal debe activarse por flancos) y se escribe el número binario 00001110 en el *bus* de datos. De esta manera, se tiene D[1]=1 (NO_HANDSET) y D[2]=D[3]=1 (HABILITA_PUERTO); SENAL_DAT_DIR=0 para permitir que ingresen datos desde la computadora hacia la interfaz y SENAL_DAT_TRANSFER=1 para permitir la transferencia de datos.

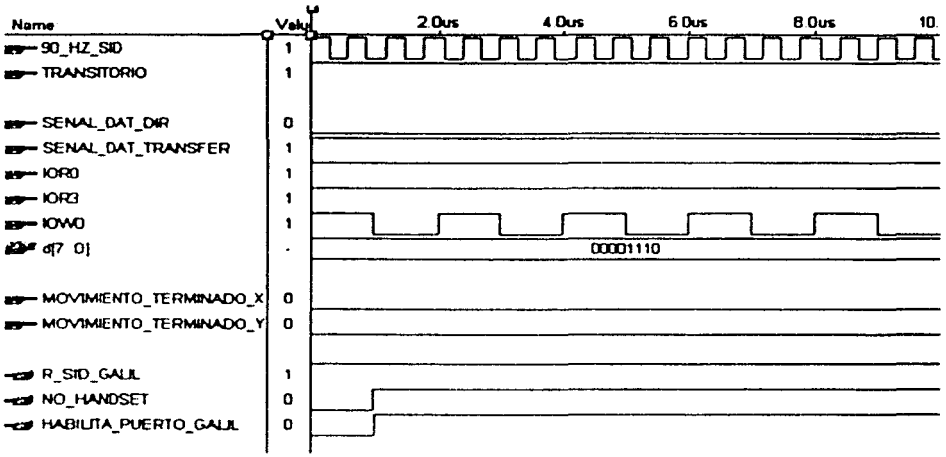


Figura 4.8. Simulación 3 del módulo del reloj sideral.

En la fig. 4.9 se muestra la cuarta simulación, que no es más que un complemento de la simulación anterior. En esta se comprueba que la señale NO_HANDSET vuelve a ser desactivada cuando la señal que se le escribe desde el *bus* de datos cambia de valor, mientras que la señal HABILITA_PUERTO_GALIL ya no vuelve a desactivarse.

Finalmente, en la fig. 4.10 se muestra la quinta simulación, en la que se indica el caso cuando SENAL_DAT_TRANSFER=0, por lo que no se permite la transferencia de datos, aunque IOR3=0, lo que provoca que el *bus* de datos presente un estado de alta impedancia.

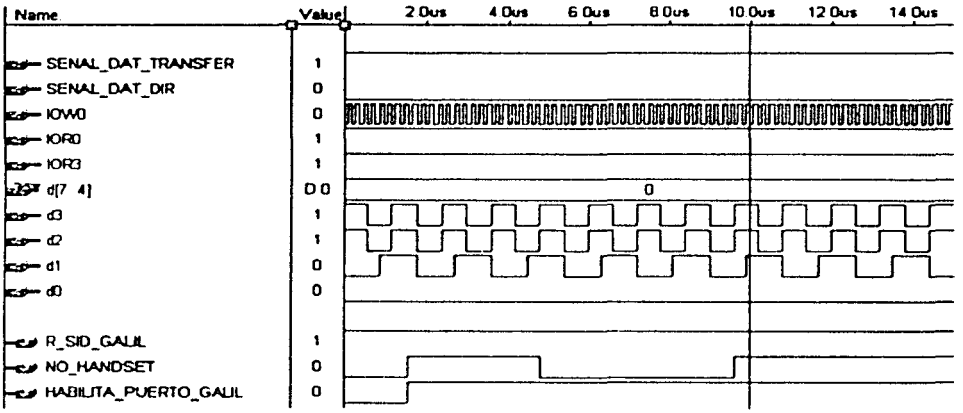


Figura 4.9. Simulación 4 del módulo del reloj sideral.

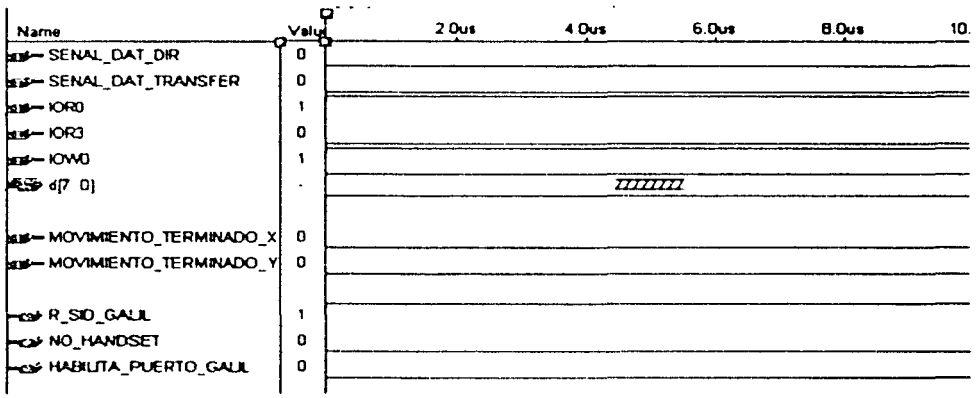


Figura 4.10. Simulación 5 del módulo del reloj sideral

4.1.3. MÓDULO DE LOS SENSORES

Entradas: **D_CAD (de 3 a 7)**
CLOCK
SENAL_DAT_TRANSFER , SENAL_DAT_DIR, HABILITA_DECOD_DIR
IOR4, IOW1

Salidas: **A_CAD (de 0 a 2)**
OE_CAD, START_ALE_CAD, CK_CAD

Entradas/Salidas: BUS_DATOS_INTERNO

- *DESARROLLO EN AHDL*

```
%MODULO DE LOS SENSORES%

OPTIONS BIT0=ANY;

INCLUDE "74374";
INCLUDE "74193";
SUBDESIGN MODULO__SENSORES
(
D_CAD[7..3] INPUT;
CLOCK INPUT;          %RELOJ DEL BUS ISA%
A_CAD[2..0] OUTPUT;
START_ALE_CAD OUTPUT;
CK_CAD,OE_CAD OUTPUT;

SENAL_DAT_DIR,SENAL_DAT_TRANSFER, HABILITA_DECOD_DIR:INPUT; %NODO EN EL
GLOBAL%
IOR4, IOW1 INPUT;              %NODOS EN EL GLOBAL%
D[7..0] BIDIR;                 %COMUN EN EL GLOBAL%

)
VARIABLE

FLIP_SENSORES 74374;
CONTADOR_CAD 74193;
BUS_DATOS_INTERNO[7..0] TRI_STATE_NODE;
PERMITE_TRANSFERENCIA_SENSORES NODE;
PRE_A_CAD[2..0],PRE_START_ALE_CAD TRI_STATE_NODE;
```

```
BEGIN

%PARTE DEL FLIP%

PERMITE_TRANSFERENCIA_SENSORES= (!SENAL_DAT_DIR & !IOW1);

PRE_A_CAD[0]=TRI(D[0],PERMITE_TRANSFERENCIA_SENSORES);
PRE_A_CAD[1]=TRI(D[1],PERMITE_TRANSFERENCIA_SENSORES);
PRE_A_CAD[2]=TRI(D[2],PERMITE_TRANSFERENCIA_SENSORES);
PRE_START_ALE_CAD=TRI(D[3],PERMITE_TRANSFERENCIA_SENSORES);

FLIP_SENSORES.D[1]=PRE_A_CAD[0].
FLIP_SENSORES.D[2]=PRE_A_CAD[1].
FLIP_SENSORES.D[3]=PRE_A_CAD[2].
FLIP_SENSORES.D[4]=PRE_START_ALE_CAD;
FLIP_SENSORES.D[5]=GND;
FLIP_SENSORES.D[6]=GND.
FLIP_SENSORES.D[7]=GND.
FLIP_SENSORES.D[8]=GND.

A_CAD[0]=FLIP_SENSORES.Q[1].
A_CAD[1]=FLIP_SENSORES.Q[2].
A_CAD[2]=FLIP_SENSORES.Q[3].
START_ALE_CAD=FLIP_SENSORES.Q[4].

FLIP_SENSORES.CLK= (SENAL_DAT_TRANSFER & !IOW1);

%PARTE DEL DIVISOR DE FRECUENCIA%

CONTADOR_CAD.UP=CLOCK.
CK_CAD=CONTADOR_CAD.QC.

OE_CAD=!IOR4.

BUS_DATOS_INTERNO[0]=TRI(GND,(SENAL_DAT_TRANSFER & !IOR4));
BUS_DATOS_INTERNO[1]=TRI(GND,(SENAL_DAT_TRANSFER & !IOR4));
BUS_DATOS_INTERNO[2]=TRI(GND,(SENAL_DAT_TRANSFER & !IOR4));
BUS_DATOS_INTERNO[3]=TRI(D_CAD[3],(SENAL_DAT_TRANSFER & !IOR4));
BUS_DATOS_INTERNO[4]=TRI(D_CAD[4],(SENAL_DAT_TRANSFER & !IOR4));
BUS_DATOS_INTERNO[5]=TRI(D_CAD[5],(SENAL_DAT_TRANSFER & !IOR4));
BUS_DATOS_INTERNO[6]=TRI(D_CAD[6],(SENAL_DAT_TRANSFER & !IOR4));
BUS_DATOS_INTERNO[7]=TRI(D_CAD[7],(SENAL_DAT_TRANSFER & !IOR4));

%PARTE COMUN%

D[0]=TRI(BUS_DATOS_INTERNO[0],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[1]=TRI(BUS_DATOS_INTERNO[1],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[2]=TRI(BUS_DATOS_INTERNO[2],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[3]=TRI(BUS_DATOS_INTERNO[3],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
```

```
D[4]=TRI(BUS_DATOS_INTERNO[4],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[5]=TRI(BUS_DATOS_INTERNO[5],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[6]=TRI(BUS_DATOS_INTERNO[6],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[7]=TRI(BUS_DATOS_INTERNO[7],SENAL_DAT_DIR & HABILITA_DECOD_DIR);

END;
```

- *SIMULACIÓN Y ANÁLISIS*

Las pruebas realizadas al módulo de los sensores consisten, básicamente, en comprobar que ingresen al *bus* de datos, cuando son llamados, los datos provenientes del convertidor analógico- digital, con la activación de la señal IOR4=0. Para lograr esto, se debe seleccionar el canal a leer del CAD, mediante la escritura de las señales A_CAD[0] a A_CAD[2], que se habilitan con la señal IOW1=0. Igualmente, se comprueba que, cuando se escribe desde el *bus* y la señal IOW1 se habilita, se activen las señales OE_CAD, START_ALE_CAD y que CK_CAD sea la octava parte de la señal CLOCK del *bus* ISA.

En la fig. 4.11 se muestra la primera simulación, en la que se indica el caso cuando se desea leer la lectura del CAD. En esta simulación, el valor que arroja el CAD es el número binario 01110xxx. Sólo se utilizan los 5 bits más significativos del CAD ya que no es necesario demasiada precisión de las medidas provenientes de los sensores (la temperatura se lee hasta una décima de grado, mientras que la humedad se lee como un porcentaje). Estas señales ingresan al *bus* de datos cuando IOR4=0 y cuando SENAL_DAT_DIR=1 (igualmente SENAL_DAT_TRANSFER=1, por lo que se permite la transferencia de datos), apareciendo en el *bus* de datos el valor binario 01110000. También es posible comprobar que el divisor de frecuencia opera correctamente comparando las señales CLOCK y CK_CAD y que OE_CAD se habilita al cuando la señal IOR4=0.

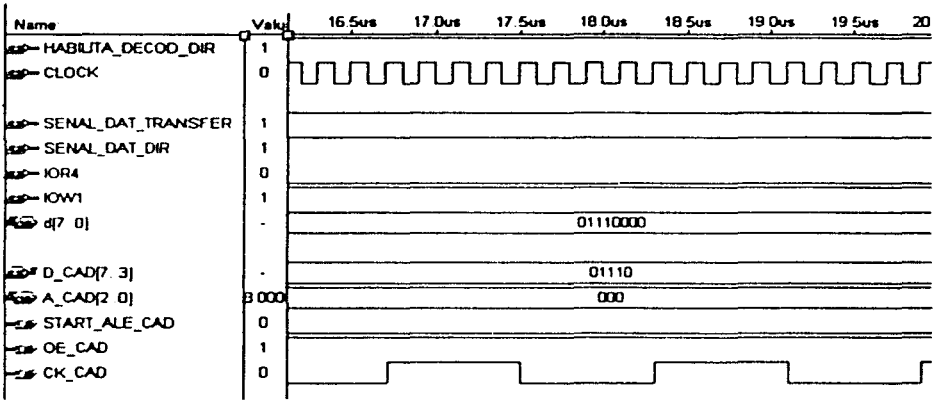


Figura 4.11. Simulación 1 del módulo de los sensores.

En la fig. 4.12 se muestra la segunda simulación, en la que se indica el caso cuando se desea escribir al CAD el canal que será posteriormente leído. En esta simulación, el canal elegido es el 1 por lo que D[1]=1. Así, como resultado de estas señales, A_CAD[2..0] toma el valor de "1". También se comprueba la habilitación de la señal START_ALE_CAD mediante la escritura de D[3]=1. Las señales anteriores son activadas cuando IOW1=0 y cuando SENAL_DAT_DIR=0. Igualmente, se verifica que OE_CAD sea desactivada al ser un ciclo de escritura.

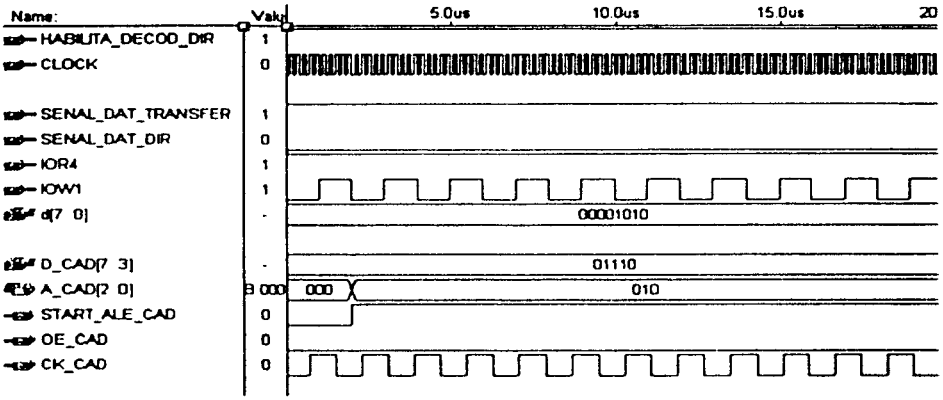


Figura 4.12. Simulación 2 del módulo de los sensores.

4.1.4. MÓDULO DEL CONTROL DE HABILITACIONES

Entradas: **A0, A1, A2**
IOW, IOR
HABILITA_DECOD_DIR, HABILITA_DECOD_MEM

Salidas: *IOR0, IOR1, IOR2, IOR3, IOR4*
IOW0, IOW1
SENAL_DAT_TRANSFER, SENAL_DAT_DIR
D0, D1, D2, D3, D4, D5, D6, D7

Entradas/Salidas: *BUS_DATOS_INTERNO*

- *DESARROLLO EN AHDL*

```
%MÓDULO DEL CONTROL DE HABILITACIONES%

INCLUDE "74244";
INCLUDE "74138";
OPTIONS BIT0=ANY;

subdesign MODULO__HABILITACIONES
(
A[2..0]:INPUT;
IOW_,IOR_:INPUT;
D[7..0]:BIDIR;
HABILITA_DECOD_DIR:INPUT;              %EN EL GLOBAL ES UN NODO%
IOR[4..0]:OUTPUT;                      %EN EL GLOBAL SON NODOS%
IOW[1..0]:OUTPUT;                      %EN EL GLOBAL SON NODOS%
SENAL_DAT_TRANSFER:OUTPUT;             %EN EL GLOBAL SON NODOS%
SENAL_DAT_DIR:OUTPUT;                  %EN EL GLOBAL ES NODO%
ENTRADA_PRUEBA[7..0]:INPUT;            %VARIABLE DE PRUEBA. NO VA EN EL
GLOBAL%
HABILITA_SALIDA_PRUEBA:INPUT;          %VARIABLE DE PRUEBA. NO VA EN EL
GLOBAL%

)

VARIABLE
BUFFER_MEM_IO:74244;
DEMUX_IOR,DEMUX_IOW:74138;
BUS_DATOS_INTERNO[7..0]:TRI_STATE_NODE;
BUFFER_IO_WRITE,BUFFER_IO_READ:NODE;
SENAL_DAT_DIR:NODE;

BUFFER_DAT_IN:74244;          %SOLO PARA PRUEBA, NO VA EN EL GLOBAL%


BEGIN

%PARTE DEL CONTROL DE IOW E IOR%

BUFFER_MEM_IO.1A[1]=A[2];
BUFFER_MEM_IO.1A[2]=A[1];
BUFFER_MEM_IO.1A[3]=A[0];
BUFFER_MEM_IO.1A[4]=IOR_;
BUFFER_MEM_IO.2A[1]=IOW_;
BUFFER_MEM_IO.2A[2]=GND;
BUFFER_MEM_IO.2A[3]=GND;

BUFFER_MEM_IO.1GN=GND;
BUFFER_MEM_IO.2GN=GND;
```

```
BUFFER_IO_READ=BUFFER_MEM_IO.1Y[4];
BUFFER_IO_WRITE=BUFFER_MEM_IO.2Y[1];

DEMUX_IOR.A=BUFFER_MEM_IO.1Y[3];
DEMUX_IOR.B=BUFFER_MEM_IO.1Y[2];
DEMUX_IOR.C=BUFFER_MEM_IO.1Y[1];
DEMUX_IOR.G2AN=BUFFER_IO_READ;
DEMUX_IOR.G2BN=GND;
DEMUX_IOR.G1=HABILITA_DECOD_DIR;

DEMUX_IOW.A=BUFFER_MEM_IO.1Y[3];
DEMUX_IOW.B=BUFFER_MEM_IO.1Y[2];
DEMUX_IOW.C=BUFFER_MEM_IO.1Y[1];
DEMUX_IOW.G2AN=BUFFER_IO_WRITE;
DEMUX_IOW.G2BN=GND;
DEMUX_IOW.G1=HABILITA_DECOD_DIR;

IOR0=DEMUX_IOR.Y0N;
IOR1=DEMUX_IOR.Y1N;
IOR2=DEMUX_IOR.Y2N;
IOR3=DEMUX_IOR.Y3N;
IOR4=DEMUX_IOR.Y4N;

IOW0=DEMUX_IOW.Y0N;
IOW1=DEMUX_IOW.Y1N;

%PARTE DE LAS COMPUERTAS%

SENAL_DAT_DIR='BUFFER_IO_READ;

TABLE
BUFFER_IO_WRITE,BUFFER_IO_READ,HABILITA_DECOD_DIR=>SENAL_DAT_TRANSFER;

0,0,0=>0;
0,0,1=>1;
0,1,0=>0;
0,1,1=>1;
1,0,0=>0;
1,0,1=>1;
1,1,0=>0;
1,1,1=>0;
END TABLE;


%PARTE DE CONTROL DEL BUS DE DATOS INTERNO%

D[0]=TRI(BUS_DATOS_INTERNO[0],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[1]=TRI(BUS_DATOS_INTERNO[1],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[2]=TRI(BUS_DATOS_INTERNO[2],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[3]=TRI(BUS_DATOS_INTERNO[3],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
```

```
D[4]=TRI(BUS_DATOS_INTERNO[4],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[5]=TRI(BUS_DATOS_INTERNO[5],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[6]=TRI(BUS_DATOS_INTERNO[6],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[7]=TRI(BUS_DATOS_INTERNO[7],SENAL_DAT_DIR & HABILITA_DECOD_DIR);


%PARTE DE BUS_DATOS_INTERNO PARA PRUEBA, NO VA EN EL GLOBAL%

BUFFER_DAT_IN.1A[1]=ENTRADA_PRUEBA[0],
BUFFER_DAT_IN.1A[2]=ENTRADA_PRUEBA[1],
BUFFER_DAT_IN.1A[3]=ENTRADA_PRUEBA[2],
BUFFER_DAT_IN.1A[4]=ENTRADA_PRUEBA[3],
BUFFER_DAT_IN.2A[1]=ENTRADA_PRUEBA[4],
BUFFER_DAT_IN.2A[2]=ENTRADA_PRUEBA[5],
BUFFER_DAT_IN.2A[3]=ENTRADA_PRUEBA[6],
BUFFER_DAT_IN.2A[4]=ENTRADA_PRUEBA[7],

BUS_DATOS_INTERNO[0]=TRI(BUFFER_DAT_IN.1Y[1],HABILITA_SALIDA_PRUEBA);
BUS_DATOS_INTERNO[1]=TRI(BUFFER_DAT_IN.1Y[2],HABILITA_SALIDA_PRUEBA);
BUS_DATOS_INTERNO[2]=TRI(BUFFER_DAT_IN.1Y[3],HABILITA_SALIDA_PRUEBA);
BUS_DATOS_INTERNO[3]=TRI(BUFFER_DAT_IN.1Y[4],HABILITA_SALIDA_PRUEBA);
BUS_DATOS_INTERNO[4]=TRI(BUFFER_DAT_IN.2Y[1],HABILITA_SALIDA_PRUEBA);
BUS_DATOS_INTERNO[5]=TRI(BUFFER_DAT_IN.2Y[2],HABILITA_SALIDA_PRUEBA);
BUS_DATOS_INTERNO[6]=TRI(BUFFER_DAT_IN.2Y[3],HABILITA_SALIDA_PRUEBA);
BUS_DATOS_INTERNO[7]=TRI(BUFFER_DAT_IN.2Y[4],HABILITA_SALIDA_PRUEBA);

END;
```

- *SIMULACIÓN Y ANÁLISIS*

Las pruebas realizadas al módulo del control de las habilitaciones consisten, básicamente, en comprobar que, según el valor que se escribe en el *bus* de direcciones A0 a A2 y, según la selección de IOR ó de IOW, se seleccionen correctamente las señales de habilitación IOR0 a IOR4 o IOW0 e IOW1.

En la fig. 4.13 se muestra la primera simulación, en la que se indica el caso cuando se selecciona una señal IOR, en este caso IOR4. Para lograr esto, se envía por A[2..0] el número binario 100, o sea 4 decimal, y se activa IOR=0, lo que provoca que IOR4=0 (evidentemente, SENAL_DAT_TRANSFER=1 y HABILITA_DECOD_DIR=1). En este módulo también se comprueba la correcta conexión entre el *bus* de datos interno del

CPLD, cuando es habilitado, y el *bus* de datos D[7..0]. Esto se consigue utilizando una entrada de prueba cuyo valor decimal es 145 en esta simulación, habilitada mediante la señal HABILITA_ENTRADA_PRUEBA=1, por lo que en el *bus* de datos D[7..0] aparece el mismo valor de 145 decimal.

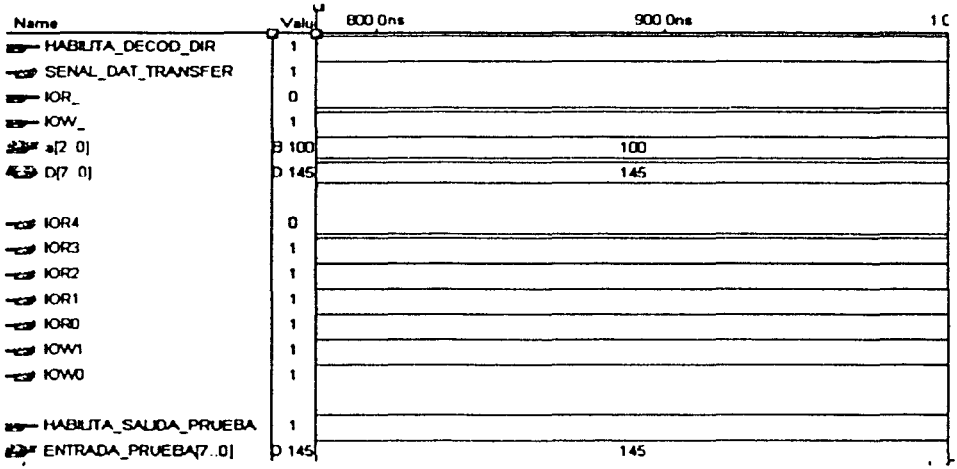


Figura 4.13. Simulación 1 del módulo del control de las habilitaciones.

En la fig. 4.14 se muestra la segunda simulación, en la que se indica el caso cuando se selecciona una señal IOW, en este caso IOW1. Para lograr esto, se envía por A[2..0] el número binario 001, o sea 1 decimal, y se activa IOW=0, lo que provoca que IOW1=0.

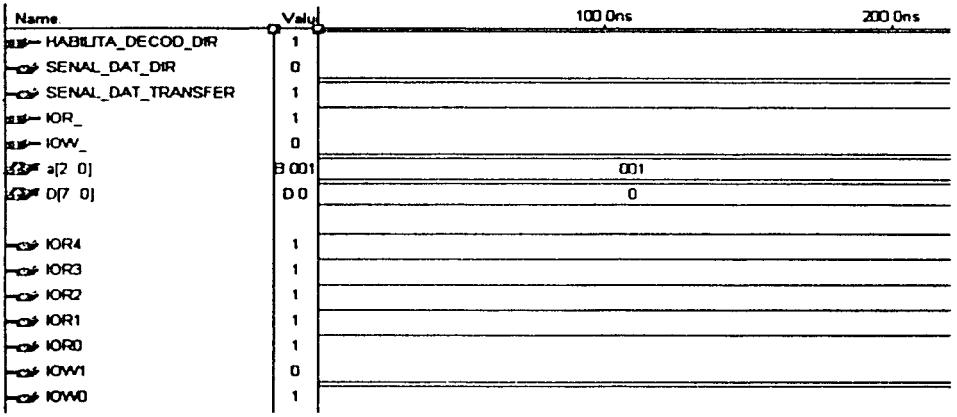


Figura 4.14. Simulación 2 del módulo del control de las habilitaciones.

4.1.5. MÓDULO DE LAS DECODIFICACIONES

Entradas: **A3, A4, A5, A6, A7, A8, A9**
DIPSWITCH_DIR (1 al 7)

Salidas: *HABILITA_DECOD_DIR*

- *DESARROLLO EN AHDL*

```
%MODULO DE LAS DECODIFICACIONES%

INCLUDE"74244";
OPTIONS BIT0=ANY.

SUBDESIGN MODULO_DECODIFICACIONES
(
A[9..3],AEN:INPUT;                    %BUS DE DIRS DEL BUS ISA%
DIPSWITCH_DIR[1..7]:INPUT;
```

```
HABILITA_DECOD_DIR:OUTPUT;              %EN EL GLOBAL ES UN NODO%
)

VARIABLE
BUFFER_DECOD_DIR:74244;
BUFFER_A[9..3]:NODE;
BUFFER_AEN:NODE;

BEGIN

BUFFER_DECOD_DIR.1A[1]=A[9];
BUFFER_DECOD_DIR.1A[2]=A[8];
BUFFER_DECOD_DIR.1A[3]=A[7];
BUFFER_DECOD_DIR.1A[4]=A[6];
BUFFER_DECOD_DIR.2A[1]=A[5];
BUFFER_DECOD_DIR.2A[2]=A[4];
BUFFER_DECOD_DIR.2A[3]=A[3];
BUFFER_DECOD_DIR.2A[4]=AEN;

BUFFER_A[3]=BUFFER_DECOD_DIR.1Y[1];
BUFFER_A[4]=BUFFER_DECOD_DIR.1Y[2];
BUFFER_A[5]=BUFFER_DECOD_DIR.1Y[3];
BUFFER_A[6]=BUFFER_DECOD_DIR.1Y[4];
BUFFER_A[7]=BUFFER_DECOD_DIR.2Y[1];
BUFFER_A[8]=BUFFER_DECOD_DIR.2Y[2];
BUFFER_A[9]=BUFFER_DECOD_DIR.2Y[3];

BUFFER_AEN=BUFFER_DECOD_DIR.2Y[4];

IF ((DIPSWITCH_DIR[1..7] == BUFFER_A[9..3]) & BUFFER_AEN==0) THEN
        HABILITA_DECOD_DIR=B"1";
        ELSE
        HABILITA_DECOD_DIR=B"0";
        END IF;

END;
```

- *SIMULACIÓN Y ANÁLISIS*

Las pruebas realizadas al módulo de las decodificaciones consisten, básicamente, en comprobar que, si la dirección escrita en el *bus* A[3..9] es igual a la establecida en el dipswitch, la señal HABILITA_DECOD_DIR se active en nivel alto. Para lograr esto, la señal del *bus* ISA AEN, que permite la habilitación para el DMA, debe estar desactivada.

En la fig. 4.15 se muestra la primera simulación, en la que se indica el caso cuando la señal HABILITA_DECOD_DIR se activa al darse los criterios anteriores. Es en este momento que la tarjeta de interfaz sabe que es a ella a la que se está comunicando la computadora. En la simulación se utiliza la dirección 1100001 binario, que es la dirección de memoria 308 hexadecimal, que es la que utiliza la tarjeta en su operación normal.

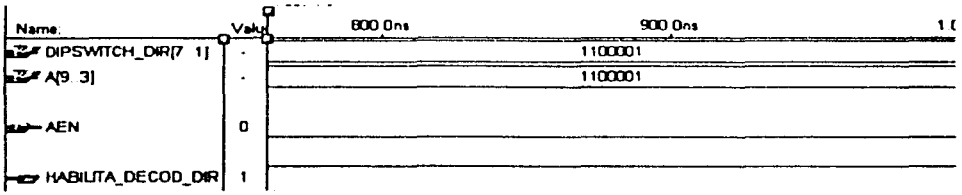


Figura 4.15. Simulación 1 del módulo de las decodificaciones.

En la fig. 4.16 se muestra la segunda simulación, en la que se indica el caso cuando la señal HABILITA_DECOD_DIR no se activa, ya que la dirección del *bus* de direcciones A[9..3] no coincide con la del *dipswitch*. En este caso, no es a la tarjeta de interfaz a la que la computadora se desea comunicar.

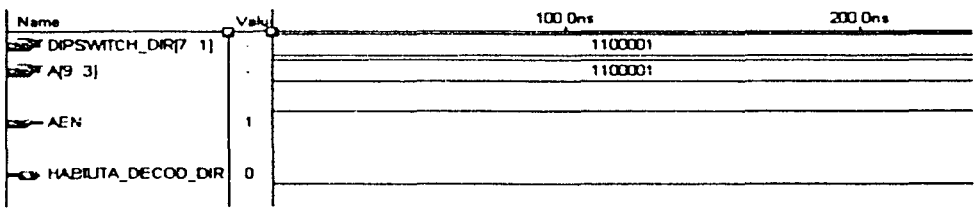


Figura 4.16. Simulación 2 del módulo de las decodificaciones.

Finalmente, en la fig. 4.17, se muestra la tercera simulación, en la que se indica el caso cuando la señal AEN=1, por lo que HABILITA_DECOD_DIR no se activa, ya que esto indica que se realizará un acceso directo a memoria (DMA).

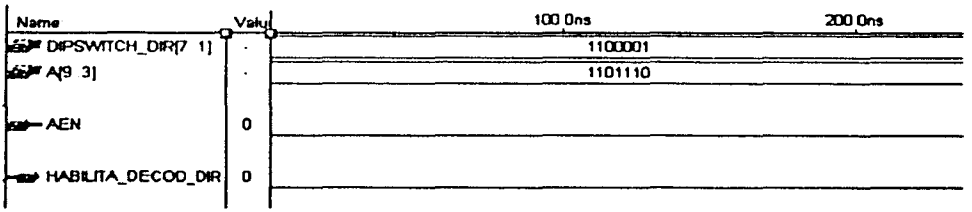


Figura 4.17. Simulación 3 del módulo de las decodificaciones.

4.2. INTEGRACIÓN DE LOS MÓDULOS EN UN SOLO PROGRAMA

Los cinco módulos anteriores, al integrarse de la forma como se expone en la sección 4.1, crean el programa final que debe programarse en el CPLD EPM7064LC68-7. En este programa, las señales de entrada/salida que relacionaban a los distintos módulos han sido convertidos en nodos y todas las señales de prueba han sido removidas. El resultado de esto es que las entradas y salidas que presenta el CPLD son para aquellas señales del *bus* ISA con las que se realiza la interfaz y las para las señales pertenecientes a los periféricos con que se comunica la interfaz como se indica a continuación:

Entradas: **A (3 a 9)**
DIPSWITCH_DIR (1 al 7)
A_ (0 a 2)
IOW, IOR

DIR_AR, CK_AR, DIR_DEC, CK DEC
90_Hz_SID, TRANSITORIO
MOVIMIENTO_TERMINADO_X
MOVIMIENTO_TERMINADO_Y
D_CAD (de 0 a 6)
CLOCK

Salidas: **HABILITA_PUERTO_GALIL**
R_SID_GALIL
A_CAD (de 0 a 2)
OE_CAD, START_ALE_CAD, CK_CAD

Entradas/Salidas: D (0 a 7)

Cabe mencionar que la única modificación en esta integración de las rutinas de los módulos anteriores, es la siguiente instrucción del módulo de las decodificaciones:

```
IF (SOFT(DIPSWITCH_DIR[7..4] == BUFFER_A[9..6]) &
(SOFT(DIPSWITCH_DIR[3..1] == BUFFER_A[5..3])) &
SOFT(BUFFER_AEN==0))
THEN
HABILITA_DECOD_DIR=B"1";
ELSE
HABILITA_DECOD_DIR=B"0";
END IF;
```

Como se vio en la sección 3.1.5, se prefirió utilizar la sentencia "if" para programar la lógica de las decodificaciones, en vez de utilizar una macrofunción de un comparador. El utilizar este recurso permite la utilización de los denominados "*soft buffers*" para los dispositivos de Altera. Si no se utilizaran en la instrucción anterior, el programa no cabría en el CPLD EPM7064LC68-7, ya que, en el proceso de encajado o *fitting* del programa al dispositivo, la lógica de esta instrucción se intentaría embeber en una sola macrocelda (específicamente, la macrocelda B). Debido a que el enunciado anterior sobrepasa el tamaño necesario de una macrocelda en el EPM7064LC68-7, aún cuando existiera espacio restante en las macroceldas A y D, se necesitaría otro CPLD de

mayor capacidad para que contuviera el programa, pero se desperdiciarían demasiados recursos, ya que el siguiente CPLD MAX7000 en capacidad es el EPM7096, con 6 macroceldas. Para solucionar esto, y utilizar correctamente el espacio sobrante en las otras macroceldas, se parte la instrucción mediante la indicación SOFT, para que se distribuya cada parte en distintas macroceldas y así no sobrepasar la capacidad de éstas.

4.2.1 DESARROLLO EN AHDL

```
%PROGRAMA QUE CONJUNTA TODOS LOS MÓDULOS  PARA REALIZAR LA INTERFAZ%

INCLUDE "74244";
INCLUDE "74138";
INCLUDE "8COUNT";
INCLUDE "7408";
INCLUDE "74393";
INCLUDE "7474";
INCLUDE "74374";
INCLUDE "74193";

OPTIONS BIT0=ANY;

subdesign INTERFAZ_LIGHT_3

(
%VARIABLES DEL MÓDULO DEL CONTROL DE LAS HABILITACIONES%
A__[2..0] INPUT;
IOW_,IOR_ INPUT;
D[7..0] BIDIR;

%VARIABLES DEL MÓDULO DE LAS DECODIFICACIONES%
A[9..3],AEN INPUT;
DIPSWITCH_DIR[7..1] INPUT;

%VARIABLES DEL MÓDULO DE MOVIMIENTOS%
DIR_AR,CK_AR INPUT;
DIR_DEC,CK_DEC INPUT;

%VARIABLES DEL MÓDULO DEL RELOJ SIDERAL%
90_HZ_SID INPUT;
TRANSITORIO INPUT;
MOVIMIENTO_TERMINADO_X,MOVIMIENTO_TERMINADO_Y:INPUT;
R_SID_GALIL OUTPUT;
HABILITA_PUERTO_GALIL OUTPUT;
```

```
%VARIABLES DEL MODULO DE LOS SENSORES%
D_CAD[7..3]:INPUT;
CLOCK:INPUT;
A_CAD[2..0]:OUTPUT;
START_ALE_CAD:OUTPUT;
CK_CAD,OE_CAD:OUTPUT;
)

VARIABLE

%CI DEL MODULO DE LAS HABILITACACIONES%
BUFFER_MEM_IO:74244;
DEMUX_IOR,DEMUX_IOW:74138;
BUS_DATOS_INTERNO[7..0]:TRI_STATE_NODE;
BUFFER_IO_WRITE,BUFFER_IO_READ:NODE;
IOR[4..0] NODE;
IOW[1..0] NODE;
SENAL_DAT_TRANSFER:NODE;
SENAL_DAT_DIR NODE;

%CI DEL MODULO DE CODIFICACIONES%
BUFFER_DECOD_DIR:74244;
BUFFER_A[9..3] NODE;
BUFFER_AEN NODE;
HABILITA_DECOD_DIR NODE;

%CI DEL MODULO DE MOVIMIENTOS%
CONTADOR_AR,CONTADOR_DEC:8COUNT;
BUFFER_AR,BUFFER_DEC:74244;
AND_CK_AR,AND_DIR_AR:7408;
AND_CK_DEC,AND_DIR_DEC:7408;

%CI DEL MODULO DEL RELOJ SIDERAL%
BUFFER_CK_SID 74244;
CONTADOR_CK_SID 74393;
AND_90_HZ_SID 7408;
AND_TRANSITORIO 7408;
FLIP_TRANSITORIO 7474;
BUFFER_MOVS_GALIL 74244;
FLIP_GUIADO 74374;
NO_GUIADO NODE;
HABILITA_PUERTO,HABILITA_PUERTO_NEG NODE;
PERMITE_TRANSFERENCIA NODE;
PRE_NO_GUIADO,PRE_NO_HANDSET TRI_STATE_NODE;
PRE_HABILITA_PUERTO,PRE_HABILITA_PUERTO_NEG:TRI_STATE_NODE;
NO_HANDSET NODE;

%CI DEL MODULO DE LOS SENSORES%
FLIP_SENSORES 74374;
```

```
CONTADOR_CAD:74193;
PERMITE_TRANSFERENCIA_SENSORES:NODE;
PRE_A_CAD[2..0],PRE_START_ALE_CAD:TRI_STATE_NODE;

BEGIN

%MODULO DEL CONTROL DE HABILITACIONES%

%PARTE DEL CONTROL DE IOW E IOR%

BUFFER_MEM_IO.1A[1]=A__[2].
BUFFER_MEM_IO.1A[2]=A__[1].
BUFFER_MEM_IO 1A[3]=A__[0].
BUFFER_MEM_IO 1A[4]=IOR_.
BUFFER_MEM_IO 2A[1]=IOW_.
BUFFER_MEM_IO 2A[2]=GND.
BUFFER_MEM_IO 2A[3]=GND.

BUFFER_MEM_IO 1GN=GND.
BUFFER_MEM_IO 2GN=GND.

BUFFER_IO_READ=BUFFER_MEM_IO 1Y[4];
BUFFER_IO_WRITE=BUFFER_MEM_IO 2Y[1].

DEMUX_IOR A=BUFFER_MEM_IO 1Y[3].
DEMUX_IOR B=BUFFER_MEM_IO 1Y[2].
DEMUX_IOR C=BUFFER_MEM_IO 1Y[1].
DEMUX_IOR G2AN=BUFFER_IO_READ.
DEMUX_IOR G2BN=GND.
DEMUX_IOR G1=HABILITA_DECOD_DIR.

DEMUX_IOW A=BUFFER_MEM_IO 1Y[3].
DEMUX_IOW B=BUFFER_MEM_IO 1Y[2].
DEMUX_IOW C=BUFFER_MEM_IO 1Y[1].
DEMUX_IOW G2AN=BUFFER_IO_WRITE;
DEMUX_IOW G2BN=GND.
DEMUX_IOW G1=HABILITA_DECOD_DIR;

IOR0=DEMUX_IOR Y0N.
IOR1=DEMUX_IOR Y1N.
IOR2=DEMUX_IOR Y2N.
IOR3=DEMUX_IOR Y3N.
IOR4=DEMUX_IOR Y4N.

IOW0=DEMUX_IOW Y0N.
IOW1=DEMUX_IOW Y1N;

%PARTE DE LAS COMPUERTAS%
SENAL_DAT_DIR='!BUFFER_IO_READ.
```

```
TABLE
BUFFER_IO_WRITE,BUFFER_IO_READ,HABILITA_DECOD_DIR=>SENAL_DAT_TRANSFER;

0,0,0=>0;
0,0,1=>1;
0,1,0=>0;
0,1,1=>1;
1,0,0=>0;
1,0,1=>1;
1,1,0=>0;
1,1,1=>0;
END TABLE;

%PARTE DE CONTROL DEL BUS%

D[0]=TRI(BUS_DATOS_INTERNO[0],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[1]=TRI(BUS_DATOS_INTERNO[1],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[2]=TRI(BUS_DATOS_INTERNO[2],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[3]=TRI(BUS_DATOS_INTERNO[3],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[4]=TRI(BUS_DATOS_INTERNO[4],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[5]=TRI(BUS_DATOS_INTERNO[5],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[6]=TRI(BUS_DATOS_INTERNO[6],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[7]=TRI(BUS_DATOS_INTERNO[7],SENAL_DAT_DIR & HABILITA_DECOD_DIR);

%MODULO DE LAS CODIFICACIONES%

%PARTE DE DECODIFICACIÒN DE DIRECCIÒN%

BUFFER_DECOD_DIR 1A[1]=A[9].
BUFFER_DECOD_DIR 1A[2]=A[8].
BUFFER_DECOD_DIR 1A[3]=A[7].
BUFFER_DECOD_DIR 1A[4]=A[6].
BUFFER_DECOD_DIR 2A[1]=A[5].
BUFFER_DECOD_DIR 2A[2]=A[4].
BUFFER_DECOD_DIR 2A[3]=A[3].
BUFFER_DECOD_DIR 2A[4]=AEN.

BUFFER_A[3]=BUFFER_DECOD_DIR 1Y[1].
BUFFER_A[4]=BUFFER_DECOD_DIR 1Y[2].
BUFFER_A[5]=BUFFER_DECOD_DIR 1Y[3].
BUFFER_A[6]=BUFFER_DECOD_DIR 1Y[4].
BUFFER_A[7]=BUFFER_DECOD_DIR 2Y[1].
BUFFER_A[8]=BUFFER_DECOD_DIR 2Y[2].
BUFFER_A[9]=BUFFER_DECOD_DIR 2Y[3].

BUFFER_AEN=BUFFER_DECOD_DIR 2Y[4].


IF (SOFT(DIPSWITCH_DIR[7..4] == BUFFER_A[9..6]) & (SOFT(DIPSWITCH_DIR[3..1] ==
BUFFER_A[5..3])) & SOFT(BUFFER_AEN==0)) THEN
```

```
        HABILITA_DECOD_DIR=B"1";
        ELSE
        HABILITA_DECOD_DIR=B"0";
        END IF;

%MODULO DE LOS MOVIMIENTOS%

%PARTE DE AR%

AND_CK_AR.2=CK_AR;
AND_CK_AR.3=CK_AR;
AND_DIR_AR.2=DIR_AR;
AND_DIR_AR.3=DIR_AR;

CONTADOR_AR.CLK=AND_CK_AR.1;
CONTADOR_AR.DNUP=AND_DIR_AR.1;
CONTADOR_AR.GN=NO_HANDSET;

BUFFER_AR.1A[1]=CONTADOR_AR.QA;
BUFFER_AR.1A[2]=CONTADOR_AR.QB;
BUFFER_AR.1A[3]=CONTADOR_AR.QC;
BUFFER_AR.1A[4]=CONTADOR_AR.QD;
BUFFER_AR.2A[1]=CONTADOR_AR.QE;
BUFFER_AR.2A[2]=CONTADOR_AR.QF;
BUFFER_AR.2A[3]=CONTADOR_AR.QG;
BUFFER_AR.2A[4]=CONTADOR_AR.QH;

%PARTE DE DEC%
AND_CK_DEC.2=CK_DEC;
AND_CK_DEC.3=CK_DEC;
AND_DIR_DEC.2=DIR_DEC;
AND_DIR_DEC.3=DIR_DEC;

CONTADOR_DEC.CLK=AND_CK_DEC.1;
CONTADOR_DEC.DNUP=AND_DIR_DEC.1;
CONTADOR_DEC.GN=NO_HANDSET;

BUFFER_DEC.1A[1]=CONTADOR_DEC.QA;
BUFFER_DEC.1A[2]=CONTADOR_DEC.QB;
BUFFER_DEC.1A[3]=CONTADOR_DEC.QC;
BUFFER_DEC.1A[4]=CONTADOR_DEC.QD;
BUFFER_DEC.2A[1]=CONTADOR_DEC.QE;
BUFFER_DEC.2A[2]=CONTADOR_DEC.QF;
BUFFER_DEC.2A[3]=CONTADOR_DEC.QG;
BUFFER_DEC.2A[4]=CONTADOR_DEC.QH;

%PARTE DEL BUS%

BUS_DATOS_INTERNO[0]=TRI(BUFFER_DEC.1Y[1],(SENAL_DAT_TRANSFER & !IOR2));
BUS_DATOS_INTERNO[1]=TRI(BUFFER_DEC.1Y[2],(SENAL_DAT_TRANSFER & !IOR2));
```

```
BUS_DATOS_INTERNO[2]=TRI(BUFFER_DEC.1Y[3],(SENAL_DAT_TRANSFER& ! !OR2));
BUS_DATOS_INTERNO[3]=TRI(BUFFER_DEC.1Y[4],(SENAL_DAT_TRANSFER & !IOR2));
BUS_DATOS_INTERNO[4]=TRI(BUFFER_DEC.2Y[1],(SENAL_DAT_TRANSFER & !IOR2));
BUS_DATOS_INTERNO[5]=TRI(BUFFER_DEC.2Y[2],(SENAL_DAT_TRANSFER & !IOR2));
BUS_DATOS_INTERNO[6]=TRI(BUFFER_DEC.2Y[3],(SENAL_DAT_TRANSFER & !IOR2));
BUS_DATOS_INTERNO[7]=TRI(BUFFER_DEC.2Y[4],(SENAL_DAT_TRANSFER & !IOR2));

BUS_DATOS_INTERNO[0]=TRI(BUFFER_AR.1Y[1],(SENAL_DAT_TRANSFER & !IOR1));
BUS_DATOS_INTERNO[1]=TRI(BUFFER_AR.1Y[2],(SENAL_DAT_TRANSFER & !IOR1));
BUS_DATOS_INTERNO[2]=TRI(BUFFER_AR.1Y[3],(SENAL_DAT_TRANSFER & !IOR1));
BUS_DATOS_INTERNO[3]=TRI(BUFFER_AR.1Y[4],(SENAL_DAT_TRANSFER & !IOR1));
BUS_DATOS_INTERNO[4]=TRI(BUFFER_AR.2Y[1],(SENAL_DAT_TRANSFER & !IOR1));
BUS_DATOS_INTERNO[5]=TRI(BUFFER_AR.2Y[2],(SENAL_DAT_TRANSFER & !IOR1));
BUS_DATOS_INTERNO[6]=TRI(BUFFER_AR.2Y[3],(SENAL_DAT_TRANSFER & !IOR1));
BUS_DATOS_INTERNO[7]=TRI(BUFFER_AR.2Y[4],(SENAL_DAT_TRANSFER & !IOR1));

%MODULO DEL RELOJ SIDERAL%

AND_90_HZ_SID.2=90_HZ_SID.
AND_90_HZ_SID.3=90_HZ_SID.
AND_TRANSITORIO.2=TRANSITORIO.
AND_TRANSITORIO.3=TRANSITORIO.

%PARTE DE MOVS TERMINADOS%
BUFFER_MOVS_GALIL.1A[1]=MOVIMIENTO_TERMINADO_X;
BUFFER_MOVS_GALIL.1A[2]=MOVIMIENTO_TERMINADO_Y;
BUFFER_MOVS_GALIL.1A[3]=GND.
BUFFER_MOVS_GALIL.1A[4]=GND.
BUFFER_MOVS_GALIL.2A[1]=GND.
BUFFER_MOVS_GALIL.2A[2]=GND.
BUFFER_MOVS_GALIL.2A[3]=GND.
BUFFER_MOVS_GALIL.2A[4]=GND.

BUS_DATOS_INTERNO[0]=TRI(BUFFER_MOVS_GALIL.1Y[1],(SENAL_DAT_TRANSFER &
!IOR3)).
BUS_DATOS_INTERNO[1]=TRI(BUFFER_MOVS_GALIL.1Y[2],(SENAL_DAT_TRANSFER &
!IOR3)).
BUS_DATOS_INTERNO[2]=TRI(BUFFER_MOVS_GALIL.1Y[3],(SENAL_DAT_TRANSFER &
!IOR3)).
BUS_DATOS_INTERNO[3]=TRI(BUFFER_MOVS_GALIL.1Y[4],(SENAL_DAT_TRANSFER &
!IOR3)).
BUS_DATOS_INTERNO[4]=TRI(BUFFER_MOVS_GALIL.2Y[1],(SENAL_DAT_TRANSFER &
!IOR3)).
BUS_DATOS_INTERNO[5]=TRI(BUFFER_MOVS_GALIL.2Y[2],(SENAL_DAT_TRANSFER &
!IOR3)).
BUS_DATOS_INTERNO[6]=TRI(BUFFER_MOVS_GALIL.2Y[3],(SENAL_DAT_TRANSFER &
!IOR3)).
BUS_DATOS_INTERNO[7]=TRI(BUFFER_MOVS_GALIL.2Y[4],(SENAL_DAT_TRANSFER &
!IOR3)).
```

```
%PARTE DEL TRANSITORIO%
HABILITA_PUERTO_GALIL=FLIP_TRANSITORIO.1Q;
FLIP_TRANSITORIO.1CLRN=AND_TRANSITORIO.1;
FLIP_TRANSITORIO.1CLK=HABILITA_PUERTO;
FLIP_TRANSITORIO.1D=HABILITA_PUERTO_NEG;

%PARTE DEL RELOJ SIDERAL%
R_SID_GALIL=(AND_90_HZ_SID.1 OR !NO_GUIADO);

CONTADOR_CK_SID.A1=R_SID_GALIL;
CONTADOR_CK_SID.A2=CONTADOR_CK_SID.Q1D;

BUFFER_CK_SID.1A[1]=CONTADOR_CK_SID.Q1A;
BUFFER_CK_SID.1A[2]=CONTADOR_CK_SID.Q1B;
BUFFER_CK_SID.1A[3]=CONTADOR_CK_SID.Q1C;
BUFFER_CK_SID 1A[4]=CONTADOR_CK_SID Q1D;
BUFFER_CK_SID 2A[1]=CONTADOR_CK_SID Q2A;
BUFFER_CK_SID 2A[2]=CONTADOR_CK_SID Q2B;
BUFFER_CK_SID 2A[3]=CONTADOR_CK_SID Q2C;
BUFFER_CK_SID 2A[4]=CONTADOR_CK_SID Q2D;

BUS_DATOS_INTERNO[0]=TRI(BUFFER_CK_SID 1Y[1],(SENAL_DAT_TRANSFER & !IOR0));
BUS_DATOS_INTERNO[1]=TRI(BUFFER_CK_SID 1Y[2],(SENAL_DAT_TRANSFER & !IOR0));
BUS_DATOS_INTERNO[2]=TRI(BUFFER_CK_SID 1Y[3],(SENAL_DAT_TRANSFER & !IOR0));
BUS_DATOS_INTERNO[3]=TRI(BUFFER_CK_SID 1Y[4],(SENAL_DAT_TRANSFER & !IOR0));
BUS_DATOS_INTERNO[4]=TRI(BUFFER_CK_SID 2Y[1],(SENAL_DAT_TRANSFER & !IOR0));
BUS_DATOS_INTERNO[5]=TRI(BUFFER_CK_SID 2Y[2],(SENAL_DAT_TRANSFER & !IOR0));
BUS_DATOS_INTERNO[6]=TRI(BUFFER_CK_SID 2Y[3],(SENAL_DAT_TRANSFER & !IOR0));
BUS_DATOS_INTERNO[7]=TRI(BUFFER_CK_SID 2Y[4],(SENAL_DAT_TRANSFER & !IOR0));

%PARTE DEL FLIP%

PERMITE_TRANSFERENCIA= (!SENAL_DAT_DIR & !IOW0);

PRE_NO_GUIADO=TRI(D[0],PERMITE_TRANSFERENCIA);
PRE_NO_HANDSET=TRI(D[1],PERMITE_TRANSFERENCIA);
PRE_HABILITA_PUERTO=TRI(D[2],PERMITE_TRANSFERENCIA);
PRE_HABILITA_PUERTO_NEG=TRI(D[3],PERMITE_TRANSFERENCIA);

FLIP_GUIADO.D[1]=PRE_NO_GUIADO;
FLIP_GUIADO.D[2]=PRE_NO_HANDSET;
FLIP_GUIADO.D[3]=PRE_HABILITA_PUERTO;
FLIP_GUIADO.D[4]=PRE_HABILITA_PUERTO_NEG;
FLIP_GUIADO.D[5]=GND;
FLIP_GUIADO.D[6]=GND;
FLIP_GUIADO.D[7]=GND;
FLIP_GUIADO.D[8]=GND;
NO_GUIADO=FLIP_GUIADO.Q[1];
NO_HANDSET=FLIP_GUIADO.Q[2];
HABILITA_PUERTO=FLIP_GUIADO.Q[3];
```

```
HABILITA_PUERTO_NEG=FLIP_GUIADO.Q[4];

FLIP_GUIADO.CLK= (SENAL_DAT_TRANSFER & !IOW0);

%MODULO DE LOS SENSORES%

%PARTE DEL FLIP%

PERMITE_TRANSFERENCIA_SENSORES= (!SENAL_DAT_DIR & !IOW1);

PRE_A_CAD[0]=TRI(D[0],PERMITE_TRANSFERENCIA_SENSORES);
PRE_A_CAD[1]=TRI(D[1],PERMITE_TRANSFERENCIA_SENSORES);
PRE_A_CAD[2]=TRI(D[2],PERMITE_TRANSFERENCIA_SENSORES);
PRE_START_ALE_CAD=TRI(D[3],PERMITE_TRANSFERENCIA_SENSORES);

FLIP_SENSORES.D[1]=PRE_A_CAD[0].
FLIP_SENSORES.D[2]=PRE_A_CAD[1].
FLIP_SENSORES.D[3]=PRE_A_CAD[2].
FLIP_SENSORES.D[4]=PRE_START_ALE_CAD.
FLIP_SENSORES.D[5]=GND.
FLIP_SENSORES.D[6]=GND.
FLIP_SENSORES.D[7]=GND.
FLIP_SENSORES.D[8]=GND.

A_CAD[0]=FLIP_SENSORES.Q[1].
A_CAD[1]=FLIP_SENSORES.Q[2].
A_CAD[2]=FLIP_SENSORES.Q[3].
START_ALE_CAD=FLIP_SENSORES.Q[4].

FLIP_SENSORES.CLK= (SENAL_DAT_TRANSFER & !IOW1);

%PARTE DEL DIVISOR DE FRECUENCIA%

CONTADOR_CAD.UP=CLOCK.
CK_CAD=CONTADOR_CAD.QC.

OE_CAD=!IOR4.

BUS_DATOS_INTERNO[0]=TRI(GND,(SENAL_DAT_TRANSFER & !IOR4)).
BUS_DATOS_INTERNO[1]=TRI(GND,(SENAL_DAT_TRANSFER & !IOR4)).
BUS_DATOS_INTERNO[2]=TRI(GND,(SENAL_DAT_TRANSFER & !IOR4)).
BUS_DATOS_INTERNO[3]=TRI(D_CAD[3],(SENAL_DAT_TRANSFER & !IOR4)).
BUS_DATOS_INTERNO[4]=TRI(D_CAD[4],(SENAL_DAT_TRANSFER & !IOR4)).
BUS_DATOS_INTERNO[5]=TRI(D_CAD[5],(SENAL_DAT_TRANSFER & !IOR4)).
BUS_DATOS_INTERNO[6]=TRI(D_CAD[6],(SENAL_DAT_TRANSFER & !IOR4)).
BUS_DATOS_INTERNO[7]=TRI(D_CAD[7],(SENAL_DAT_TRANSFER & !IOR4)).

END.
```

4.2.2. SIMULACIÓN Y ANÁLISIS

Las pruebas realizadas al programa que contiene todos los módulos consisten, básicamente, en comprobar que, cuando se hace una lectura o escritura a puerto con la instrucción IOR e IOW respectivamente, se realice correctamente. En las siete primeras simulaciones se supone que es con la interfaz con quien se comunica la computadora, por lo que la señal HABILITA_DECOD_DIR=1.

En la fig. 4.18 se muestra la primera simulación, en la que se indica el caso cuando se leen las señales de ascensión recta provenientes del controlador del telescopio. La señales que se activan son IOR=0 y $A[2..0]=1_{dec}$, por lo que IOR1=0. La cuenta es ascendente.

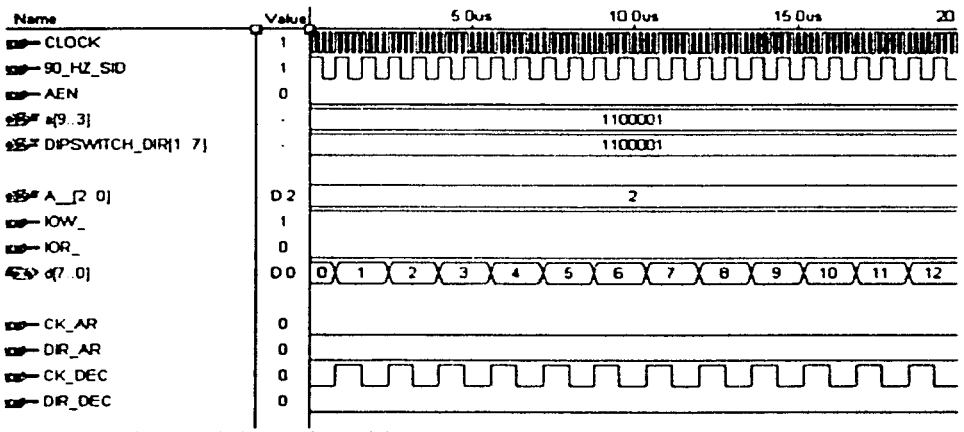


Figura 4.18. Simulación 1 del programa que a su vez se programa en el CPLD.

En la fig. 4.19 se muestra la segunda simulación, en la que se indica el caso cuando se leen las señales de declinación provenientes del controlador del telescopio. La señales que se activan son IOR=0 y A[2..0]=2_{dec}, por lo que IOR2=0. La cuenta es descendente.

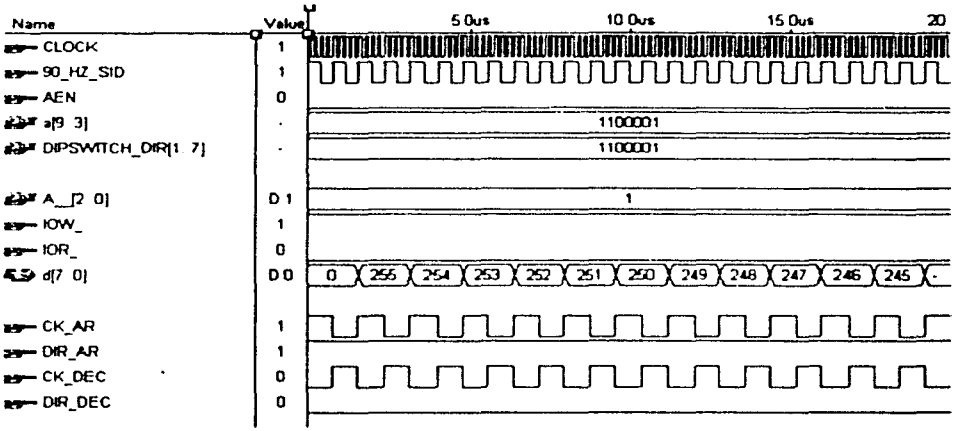


Figura 4.19. Simulación 2 del programa que a su vez se programa en el CPLD.

En la fig. 4.20 se muestra la tercera simulación, en la que se indica el caso cuando se leen las señales que indican la finalización de los movimientos del telescopio desde el controlador de motores, *Galil*. La señales que se activan son IOR=0 y A[2..0]=3 $_{dec}$, por lo que IOR3=0. De esta manera, las señales MOVIMIENTO_TERMINADO_X y MOVIMIENTO_TERMINADO_Y llegan al *bus* de datos.

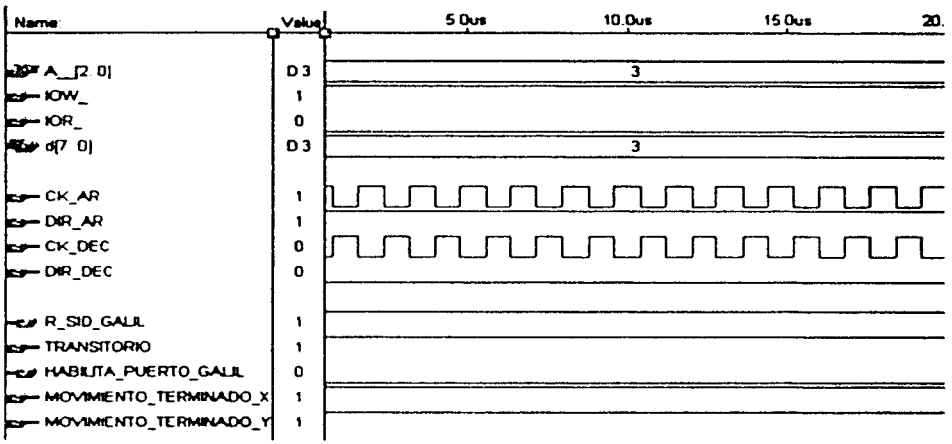


Figura 4.20. Simulación 3 del programa que a su vez se programa en el CPLD.

En la fig. 4.21 se muestra la cuarta simulación, en la que se indica el caso cuando se lee la señal que proviene del reloj sideral. La señales que se activan son IOR=0 y A[2..0]=0, por lo que IOR0=0. De esta manera, las señale R_SID_GALIL llega al *bus* de datos.

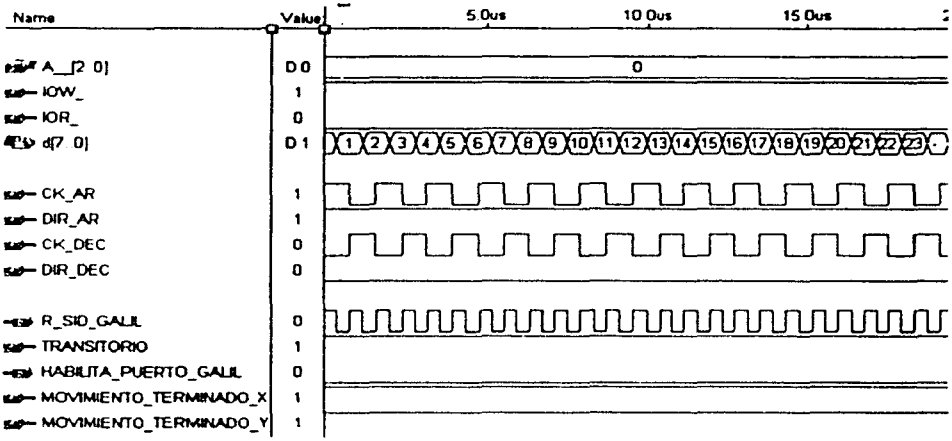


Figura 4.21. Simulación 4 del programa que a su vez se programa en el CPLD.

En la fig. 4.22 se muestra la quinta simulación, en la que se indica el caso cuando se lee las señales que provienen del convertidor analógico digital. La señales que se activan son IOR=0 y A[2..0]=4_{dcc}, por lo que IOR4=0. De esta manera, las señales D_CAD[7..3] llegan al *bus* de datos. Igualmente, la señal OE_CAD es encendida para permitir la transferencia de datos desde el CAD.

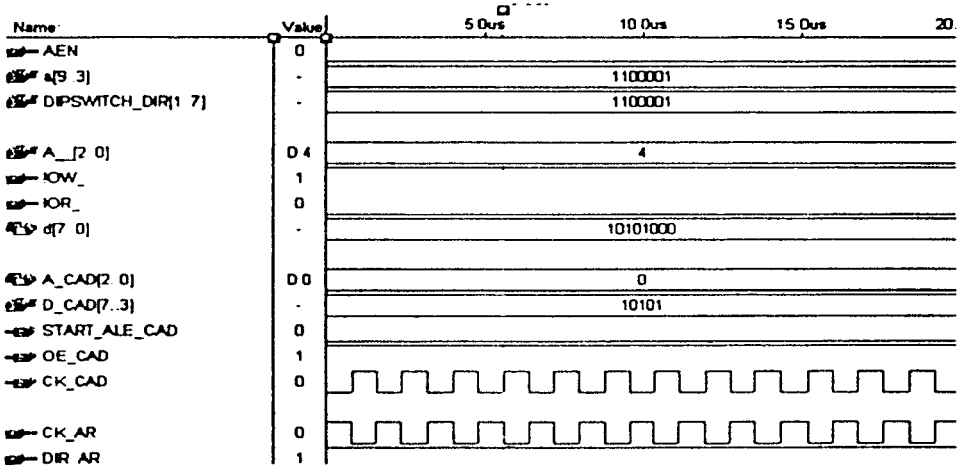


Figura 4.22. Simulación 5 del programa que a su vez se programa en el CPLD.

En la fig. 4.23 se muestra la sexta simulación, en la que se indica el caso cuando se escriben las señales habilitan la señal HABILITA_PUERTO_GALIL. Las señales que se activan son IOW=0, y A[2..0]=0, por lo que IOR0=0, y D[7..0]=00001100$_{bin}$, por lo que HABILITA_PUERTO = HABILITA_PUERTO_NEG=1. De esta manera, HABILITA_PUERTO_GALIL se enciende cuando el flanco de IOW cambia.

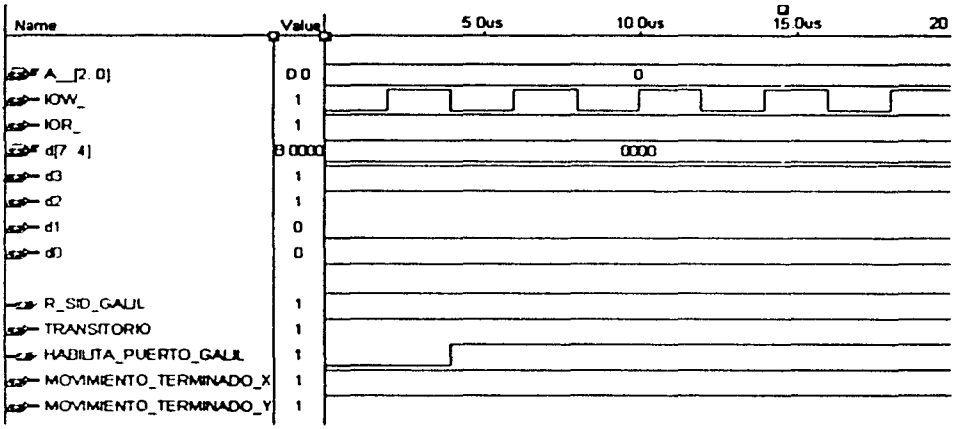


Figura 4.23. Simulación 6 del programa que a su vez se programa en el CPLD.

En la fig. 4.24 se muestra la séptima simulación, en la que se indica el caso cuando se escriben las señales habilitan la señal START_ALE_CAD y seleccionan el canal dos del CAD. La señales que se activan son IOW=0, y A[2..0]=1, por lo que IOR0=1, y D[7..0]=00001010$_{bin}$, por lo que A_CAD[2..0]=010 $_{bin}$ y START_OE_CAD se enciende cuando el flanco de IOW cambia.

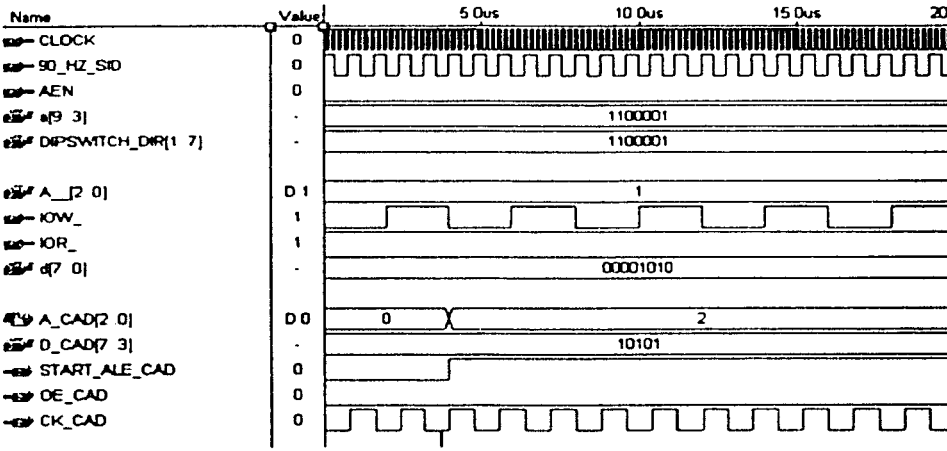


Figura 4.24. Simulación 7 del programa que a su vez se programa en el CPLD.

Finalmente, en la fig. 4.25 se muestra la octava simulación, en la que se indica el caso cuando las señales de A[9..3] no coinciden con las del *dipswitch*. Como DIPSWITCH_DIR[1..7]=1100111_{bin} y A[9..3]=1100001_{bin}, la computadora no se está comunicando con la tarjeta de interfaz, por lo que la señal HABILITA_DECOD_DIR no se activa, y el *bus* de datos de la interfaz presenta un estado de alta impedancia.

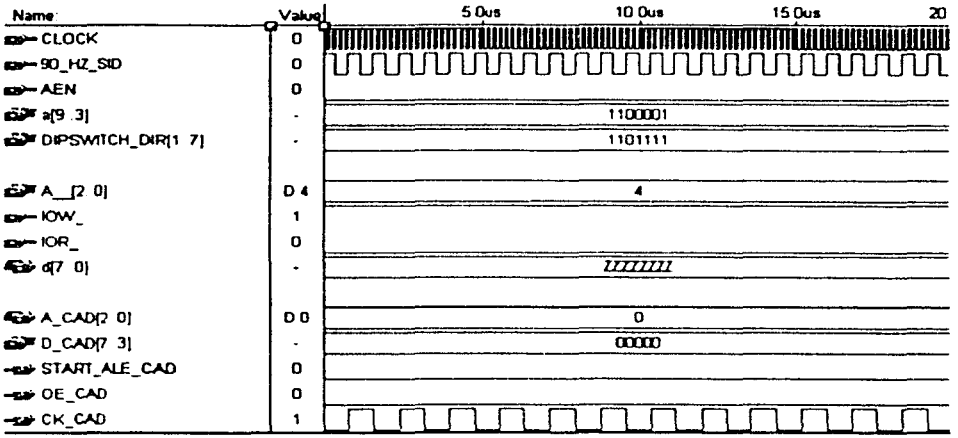


Figura 4.25. Simulación 8 del programa que a su vez se programa en el CPLD.

4.3. DISEÑO DEL CIRCUITO IMPRESO

El diseño del circuito impreso de la interfaz se basa, principalmente, en las necesidades del CPLD EPM7064LC68-7 y de la lógica del módulo de amplificación de las señales provenientes de los sensores. Como parte de la programación del CPLD, se le asignaron manualmente las señales de entrada/salida a distintas terminales, buscando implementar una configuración tal que el diseño del circuito impreso se facilitara. Esta

asignación manual debe ser realizada cuando se desactiva, en el *software* de Max+Plus II, la parte del proceso de compilación en el que se asignan automáticamente las entradas/salidas de las terminales del CPLD. De esta manera, en vez de que el *software* asigne las terminales según sus propios criterios internos de eficiencia, el programador puede asignar las señales de entrada/salida a las terminales que mejor le convenga para acomodar físicamente, según su diseño, las terminales del CPLD asociadas a las señales de entrada/salida en el circuito impreso.

Respetando las restricciones en la configuración de terminales del EPM7064LC68-7, analizadas en la sección 2.2.2, y a las limitaciones de interconexión entre los distintos LABs y el PIA que impone Max+Plus II durante la compilación de sus programas, la asignación más óptima de las señales a las terminales del CPLD se muestra en la tabla 4.1.

Cabe señalar que existen dos señales, ~PIN001 y ~PIN002, que no están definidas en ninguno de los módulos de programación ni en las terminales reservadas del CPLD y, sin embargo, deben considerarse en la implementación final del mismo. Estas dos señales son resultado de que, en los dispositivos MAX7000, la señal que habilita a un *bus* de datos bidireccional de tres estados, debe forzosamente provenir de una entrada física al CPLD, que esté conectada directamente a una compuerta que niegue a esta señal. Como la señal que activa el habilitador del *bus* de datos interno del CPLD está necesariamente programado como la combinación de dos señales, que provienen de nodos para juntarse en la instrucción (SENAL_DAT_DIR AND HABILITA_DECOD_DIR), Max+Plus II crea una señal asociada a una terminal de salida, la señal ~PIN001, que contiene el resultado de la instrucción anterior, negado, para que éste sea utilizada como señal de entrada en ~PIN002, que es la que finalmente se utiliza como habilitador del *bus* de datos interno. Es por esta razón que existen éstas dos señales y que deban ser unidas físicamente en el circuito impreso.

Terminal	*Señal*
1	IOR
2	DIPSWITCH_DIR7
3	VCCINT
4	DIPSWITCH_DIR6
5	DIPSWITCH_DIR5
6	GND
7	DIPSWITCH_DIR4
8	DIPSWITCH_DIR3
9	DIPSWITCH_DIR2
10	DIPSWITCH_DIR1
11	VCCIO
12	~PIN0002
13	DIR_AR
14	CK_AR
15	CK_CAD
16	GND
17	OE_CAD
18	START_ALE_CAD
19	A_CAD2
20	A_CAD1
21	VCCIO
22	A_CAD0
23	D_CAD7
24	D_CAD6
25	D_CAD5
26	GND
27	D_CAD4
28	D_CAD3
29	90_HZ_SID
30	HABILITA PUERTO GALIL
31	VCCIO
32	IOW
33	R_SID_GALIL
34	GND
35	VCCINT

Terminal	*Señal*
36	TRANSITORIO
37	A0
38	GND
39	A1
40	A2
41	A3
42	A4
43	VCCIO
44	A5
45	A6
46	A7
47	A8
48	GND
49	A9
50	AEN
51	D0
52	D1
53	VCCIO
54	D2
55	D3
56	D4
57	D5
58	GND
59	D6
60	D7
61	MOVIMIENTO TERMINADO_X
62	MOVIMIENTO TERMINADO_Y
63	VCCIO
64	DIR_DEC
65	CK_DEC
66	GND
67	CLOCK
68	~PIN001

Tabla 4.1. Asignación de señales a las terminales del EPM7064LC68-7.

Igualmente, con la asignación de señales y terminales mostrada anteriormente, se busca reducir en lo posible el tamaño de las pistas en el circuito impreso y sus trayectorias, tanto si van o provienen de un conector de entrada o de salida, agrupando, en lo posible, aquellas señales que tienen un interés común.

4.3.1. TRATAMIENTO DE LAS ESPECIFICACIONES

Las características de la mayoría de los elementos que integran la interfaz han sido desarrollados a lo largo de este trabajo, particularmente en la sección 4. Sin embargo, existen otras consideraciones que se exponen a continuación, ya que aportan criterios que sirvieron para la implementación del circuito impreso de la tarjeta de interfaz.

La tarjeta de interfaz se conecta a la ranura o *slot* para 62 terminales del *bus* ISA de la computadora. Debido a esta característica, la interfaz sirve como enlace con la fuente de alimentación de la computadora, al presentar terminales para las tensiones +5V, -5V, tierra, +12V y -12V.

Por otra parte, todas las señales digitales que ingresan a la tarjeta desde los periféricos lo hacen, cada una, vía un optoacoplador: el OA4N28. Se utiliza un optoacoplador del tipo LED-fototransistor ya que, en éstos, la terminal común para el circuito de entrada en el LED no es el mismo que la terminal común para el circuito de salida del fototransistor. Esto no sólo implica que no exista un camino de conducción entre los circuitos sino que, además, al llevar a tierra uno de los dos circuitos, la configuración sigue operando, aún cuando se dejara flotante el otro en caso accidental. Esta medida de seguridad no sólo protege a la electrónica de la interfaz, sino también de la computadora en general, de alguna descarga que pudiera presentarse en estas entradas.

El utilizar optoacopladores también minimiza los lazos de corriente entre las distintas fuentes del sistema.

Igualmente, se toma la precaución de colocar pares de capacitores de tantalio de 0.01 μF y de 100 nF, denominados de *by-pass*, entre las terminales de alimentación de los circuitos integrados. Estos capacitores anulan el ruido que se genera en un ambiente de *switcheo* de alta velocidad, ocasionado por la carga y descarga de los capacitores internos de los circuitos integrados. Las corrientes generadas por los cambios de estado de las terminales de salida de los circuitos integrados causa un efecto de resonancia en la línea de alimentación. Este comportamiento puede generar señales falsas generando errores en el funcionamiento de la lógica electrónica. El capacitor de *by-pass* almacena una carga eléctrica que es liberada a la línea de alimentación cuando ocurre un pico de tensión transitorio, actuando como una fuente de baja impedancia, minimizando el ruido en la línea.

El valor de los capacitores de *by-pass* de 0.01 μF y de 100 nF es porque, al conectarse en paralelo, presentan una capacitancia de 110 nF. Este valor se obtiene de la siguiente ecuación:

$$C = \frac{I_{CC} \times N \times \Delta t}{\Delta V_{CC}}$$

donde:

I_{CC}= Corriente necesaria para que una terminal de salida cambie de estado bajo a alto.

N= Número de terminales de salida que cambian de estado.

Δt= Tiempo requerido para que el capacitor cargue la línea.

ΔV_{CC}= Caída de tensión que puede ser tolerada.

Los datos a sustituir en la fórmula anterior se obtienen de la manera explicada a continuación. De la figura 14 de las hojas de especificaciones del CPLD EPM7064 (*I_{CC} vs. Frequency for MAX7000 Devices*), se observa que a una frecuencia de 5 MHz, similar a la máxima frecuencia con la que operará la interfaz electrónica, ya que la frecuencia del bus ISA es de 4.77 MHz, se tiene que I_{CC} es aproximadamente 120 mA. El número de terminales de salidas digitales que cambian en la interfaz son 32. Siguiendo los parámetros sugeridos en la publicación *The By-pass Capacitor in High Speed Enviroments* de Texas Instruments se fija: Δt=3 ns y ΔV_{CC}=0.1 V. Sustituyendo estos valores se obtiene lo siguiente:

$$C = \frac{(120\times 10^{-3})(32)(3\times 10^{-9})}{0.1} \approx 110nF$$

Estos capacitores son de tantalio ya que estos presentan una inductancia muy pequeña. Esto es importante ya que las armónicas de alta frecuencia, presentes en el ruido de la línea de alimentación, ocasionan que los capacitores que cuentan con una alta inductancia, operen como circuito abierto. De la misma manera, debido a que la impedancia de línea, provocada por el alto consumo de corriente de los CI, genera una caída de tensión (los CI se van comportando como divisores de tensión), los capacitores de *by-pass* operan como fuentes de tensión suplementarias para contrarrestar este efecto.

Igualmente, estos capacitores de *by-pass* ayudan a atenuar el ruido ocasionado por la fuente de alimentación de la computadora, que es una fuente conmutada. Las fuentes de alimentación conmutadas o *switcheadas*, basan su funcionamiento en un circuito de gran velocidad que detecta constantemente el nivel de alguna de las salidas de tensión de la fuente para que, al momento en que se registre alguna variación, se envíen las órdenes adecuadas a un conmutador para que éste se abra y así el nivel de la tensión vuelva rápidamente a la normalidad. En las computadoras, el dispositivo que se encarga de controlar la anchura de los pulsos de encendido del conmutador es un controlador basado en la modulación de ancho de pulsos o PWM (*Pulse Width Modulation*), el cual es un

sistema económico y a la vez preciso, pero que suministra mucho ruido, el cual se busca eliminar con los capacitores de *by-pass*.

De esta manera, los elementos que conforman físicamente a la interfaz pueden listarse de la siguiente manera:

- El CPLD EPM7064LC68-7 (U1).
- Conector de cinco terminales para el ingreso de las señales de los movimientos provenientes del controlador manual del telescopio (HANDSET).
- Conector de dos terminales para el ingreso de la frecuencia sideral (FRECSID).
- Conector de 4 terminales para las señales que se dirigen y proceden del controlador de motores (GALIL)
- Un *dipswitch* de ocho interruptores para la decodificación de dirección (DIPSWITCH).
- El convertidor analógico digital ADC0808 (U2).
- Dos capacitores para establecer la tensión de referencia del CAD (C2 y C3).
- Un conector de 6 terminales para las señales que provienen de los sensores (SENSORES).
- Un CI TL074 (U8) que contiene 4 amplificadores operacionales (U8A, U8B, U8C y U8D) para amplificar las señales que provienen de los sensores.
- Ocho resistencias, un diodo y cinco resistencias ajustables para polarizar los amplificadores operacionales.
- Tres juegos de capacitores de *bypass* (CBY).
- Un diodo, un capacitor y una resistencia para crear la señal TRANSITORIO.
- Conector para el *bus* ISA (J1).
- Cinco optoacopladores (U3, U4, U5, U6 y U7) con sus cinco respectivas resistencias.
- Conector de 10 terminales para la programación del CPLD (PROGRAMACIÓN).

4.3.2. DESARROLLO EN PROTEL

La conjunción de todos los elementos anteriormente mencionados se muestra primero en el circuito esquemático de la fig. 4.26 y, posteriormente, en el diagrama del circuito impreso correspondiente, figuras 4.27 y 4.28. La localización de los componentes en el circuito impreso se muestra en la fig. 4.29. Estos diseños han sido desarrollados mediante la versión EDA/CLIENT 98 de PROTEL.

El circuito esquemático está realizado a partir de los elementos de las librerías *Device, Texas Instruments, National Semiconductors* y *Altera*. Para los componentes como las resistencias, capacitores, diodos y potenciómetros se emplean las bases de los tamaños adecuados a los componentes normalmente ofrecidos en el mercado. Las terminales con las que se alimentan de tensión tanto el CAD, como el EPM7064LC68, no están indicados, pero se toman en cuenta para el circuito impreso.

El diagrama esquemático presenta el circuito impreso de la interfaz realizado con dos pistas sobre el circuito impreso; estas son la pista superior y la pista inferior. El diámetro mínimo posible de las pistas es de 20 *mils* y la distancia mínima posible entre ellas es de 10 *mils*. El diámetro de las perforaciones es de 28 *mils*, mientras que el anillo conductor alrededor de ellas es de 50 *mils*. El CPLD se fija al circuito impreso mediante una base de 68 terminales que se suelda al circuito impreso.

Cabe señalar que el circuito impreso no incluye el soporte mecánico con el que se atornilla la tarjeta a la armadura de la computadora, para que pueda utilizarse cualquier modelo de éste, con los conectores adecuados para las entradas particulares de cada uno de los periféricos.

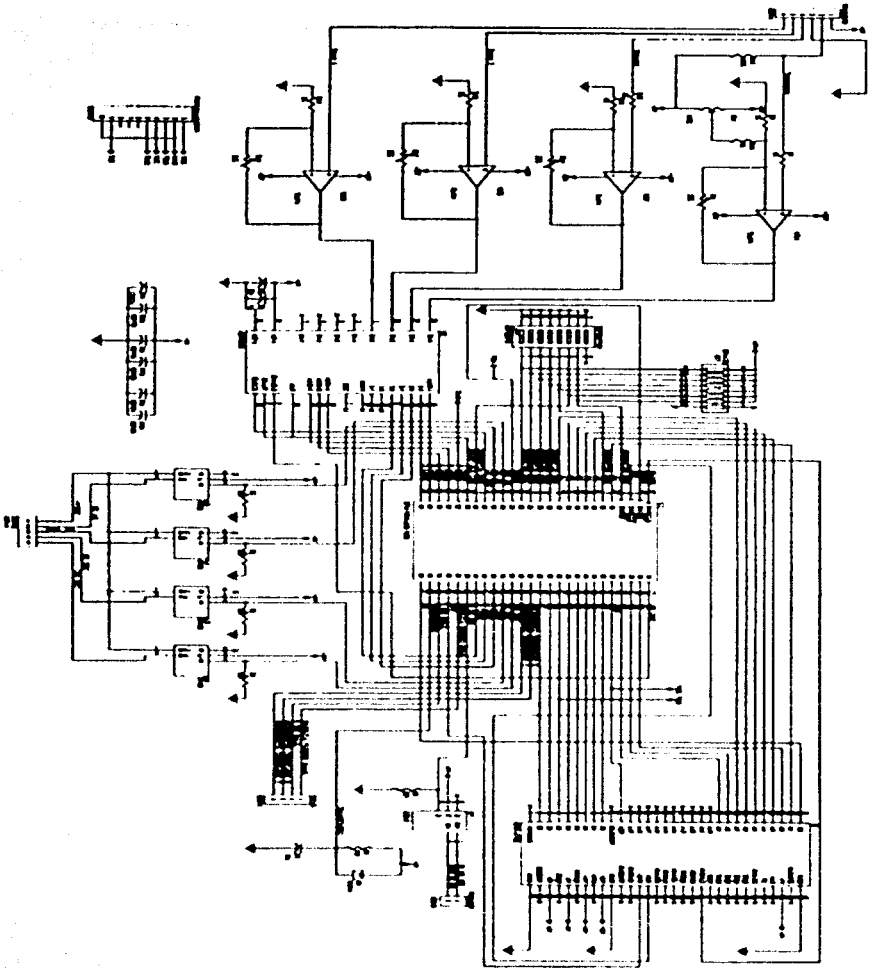


Figura 4.26. Circuito esquemático de la interface electrónica

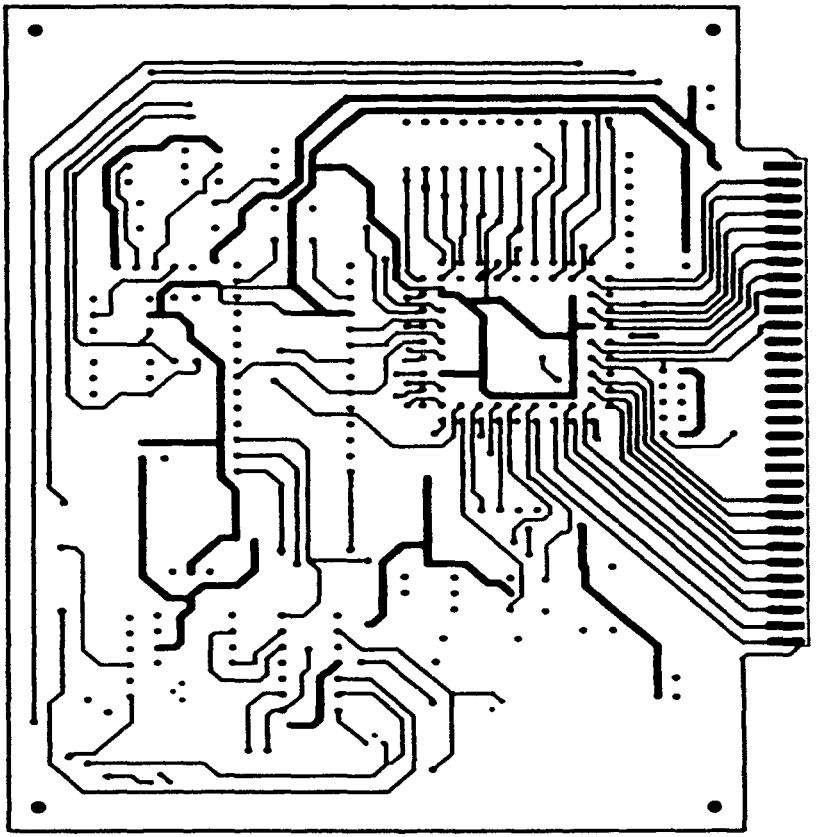


Figura 4.27. Circuito impreso de la interfaz electrónica (pista superior).

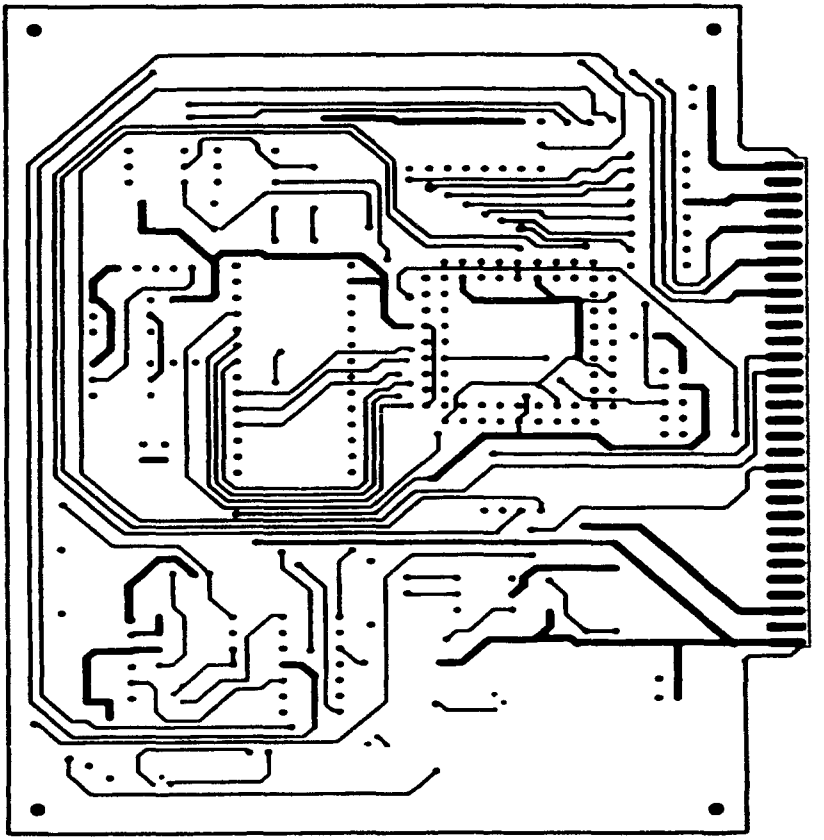


Figura 4.28. Circuito impreso de la interfaz electrónica (pista inferior).

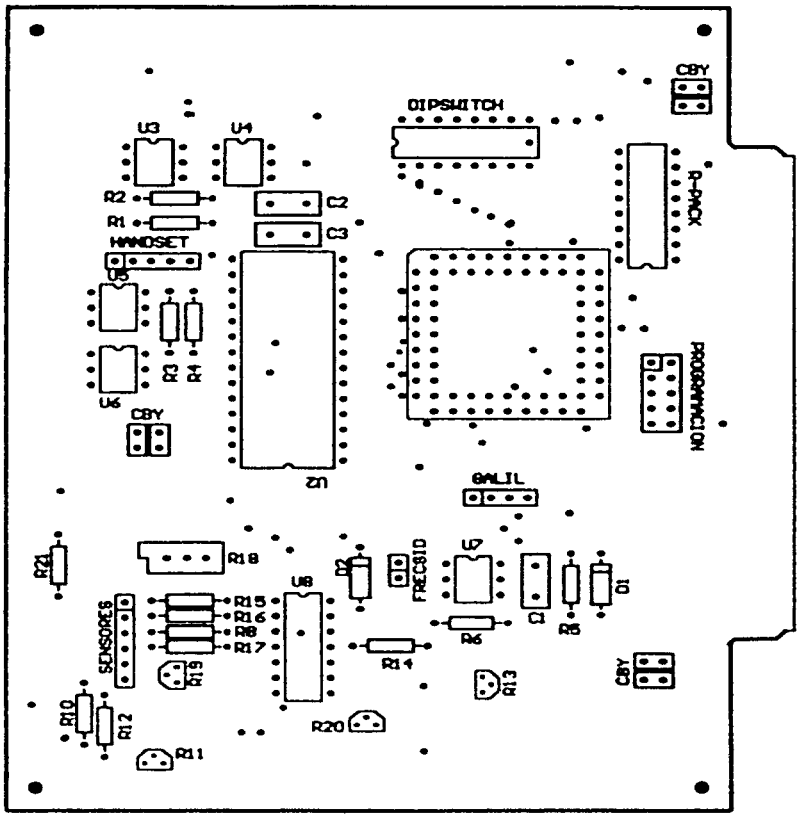


Figura 4.29. Localización de los componentes en el circuito impreso.

V

EVALUACIÓN DE LA OPERACIÓN DE LA INTERFAZ

Durante la compilación del programa listado en la sección 4.3.1, Max+Plus II genera un archivo de extensión .pof (*Programmer Objet File*). Conectando la terminal del puerto paralelo de la computadora a la interfaz Byteblaster (mostrada en la sección 2.2.3) y a su vez, conectando la terminal hembra de esta interfaz, a la interfaz electrónica, se procede a programar el CPLD (en el comando de configuración del *Hardware* de Max+Plus II se especifica la utilización del Byteblaster como interfaz de programación, se indica el puerto LPT de la computadora a utilizar, el tipo de dispositivo a programar, el EPM7064LC68, y se indica el programa de extensión .pof a programar).

Las características de operación que muestra el CPLD programado, en conjunto con los demás elementos de la interfaz se resume a continuación.

Los tiempos que se tarda el CPLD en procesar las señales de entrada a la interfaz, cuando son llamadas, para convertirlas en señales de salida, se muestran en la tabla 5.1.

	D0	D1	D2	D3	D4	D5	D6	D7
CK AR	17.5 ns	12.5 ns	12.5 ns	12.5 ns	12.5 ns	12.5 ns	12.5 ns	12.5 ns
CK DEC	17.5 ns	12.5 ns	12.5 ns	12.5 ns	12.5 ns	12.5 ns	12.5 ns	12.5 ns
MOV. TERMINADO X	12.5 ns							
MOV. TERMINADO Y		7.5 ns						
90 HZ SID	17.5 ns	12.5 ns	12.5 ns	12.5 ns	17.5 ns	17.5 ns	17.5 ns	17.5 ns
D CAD 3				7.5 ns				
D CAD 4					7.5 ns			
D CAD 5						7.5 ns		
D CAD 6							7.5 ns	
D CAD 7								7.5 ns

Tabla 5.1. Tiempos de retardo en el proceso de señales en el CPLD.

La máxima frecuencia de los trenes de pulsos que pueden ingresarse en las terminales CK AR y CK DEC, sin que los contadores pierdan su capacidad de conteo correcta, es aproximadamente 125 MHz, que es inferior al valor teórico de especificaciones de 151.5 MHz.

Los tiempos que transcurren entre que se activa la señal IOW o IOR, y se coloca información en el *bus* de datos, se muestran en la tabla 5.2 (estando las señales A0, A1 y A2 del *bus* de direcciones activadas).

	D0	D1	D2	D3	D4	D5	D6	D7
IOW	22.5 ns	17.5 ns	17.5 ns	17.5 ns	22.5 ns	22.5 ns	22.5 ns	22.5 ns
IOR	13 ns	13 ns	13 ns	13 ns	13 ns	13 ns	13 ns	13 ns

Tabla 5.2. Tiempos de colocación de la información en el *bus*.

De aquí se ve que los tiempos de respuesta de las señales del *bus* de datos para la señal IOW son diferentes, pero esto no representa problema alguno ya que, al ser realizada la lectura desde el programa de control en intervalos de tiempo mayores a 22.5 ns, las lecturas que se obtienen pertenecen a un mismo evento.

El tiempo que se retrasa la señal CLOCK del *bus* ISA para emerger dividida entre 8 como CK CAD es de 25.5 ns.

Los tiempos que se tardan las señales IOW e IOR en habilitar las señales de salida, que en su proceso atraviesan alguno de los *flipflops* (el de guiado o el de los sensores), se muestran en la tabla 5.3.

	HABILITA PUERTO GALIL	OE CAD	R SID GALIL	START ALE CAD	A CAD 0	A CAD 1	A CAD 2
IOW	12.5 ns		12.5 ns	7.5 ns	7.5 ns	7.5 ns	7.5 ns
IOR		7.5 ns					

Tabla 5.3. Tiempos de retraso en los *flipflops* del CPLD.

En las señales involucradas con el uso de los *flipflops*, el intervalo mínimo posible entre la aplicación de una señal entrada y una transición de nivel bajo a nivel alto que alimenta el reloj de entrada del *flipflop*, llamado tiempo de activación (*setup time*) es de 3 ns. Esto es aplicable para las señales de los movimientos del telescopio: DIR AR y DIR DEC que ingresan a los contadores; las señales involucradas con los *flipflops* de guiado y de los sensores: A0, A1, A2 y las señales involucradas con la decodificación de direcciones: A3, A4, A5, A6, A7, A8, A9. Igualmente, el periodo de tiempo mínimo por el que una señal debe ser mantenida en una terminal de entrada después de una transición en la terminal de entrada que alimenta el reloj entrada del *flipflop*, llamado tiempo de mantenimiento (*hold time*) es de 2 ns para las señales anteriores. Los tiempos mínimos de activación y de mantenimiento para las señales DIPSWITCH DIR 1 al 8 son los mismos: 3ns y 2 ns, respectivamente. Sin embargo, debido a que las señales provenientes del *dipswitch* no son desactivadas nunca, no tiene sentido aplicarles estos parámetros.

En la práctica, estos tiempos mínimos son superados tranquilamente por las señales provenientes del controlador manual del telescopio y las de la consola de guiado. Debido a que el tiempo máximo de levantamiento o de caída de una señal para que sea interpretada correctamente por el CPLD es, según su hoja de características, de 40 ns, basta con que las señales anteriores tengan una duración mayor a este lapso, en estado alto o bajo, para evitar indeterminaciones lógicas o fallas en los *flipflops*.

El sensor de temperatura utilizado para las mediciones es un LM35CH, de presentación metálica, ya que su tensión de salida es proporcional a la temperatura ambiente. Comparadas las mediciones obtenidas a la salida del módulo de los sensores, explicado en la sección 3.1.3, y las mediciones simultáneas de un multímetro digital, se tiene que, para las lecturas que oscilan desde los 11° C en adelante, la medición coincide perfectamente con la lectura del multímetro, mientras que, las lecturas inferiores a esta temperatura (hasta -12°C, que fue la temperatura mínima que se logró conseguir mediante la utilización de un extinguidor de dióxido carbono a bajas temperaturas), tienden a desviarse de ½° a 1°C.

Igualmente, cabe señalar que las lecturas obtenidas en el monitor de la computadora de la consola de control del telescopio, no cambian tan rápidamente comparadas a las del multímetro digital. Esto se debe a que el proceso de la lectura proveniente del sensor se realiza a intervalos de un segundo de separación, a instrucción del programa de la consola de control del telescopio, mientras que, el despliegue de la lectura obtenido en el multímetro digital, es prácticamente instantánea. Esta falta de despliegues en tiempo real se justifica ya que al operador del telescopio sólo le basta observar la tendencia de los cambios de temperatura en el ambiente para la toma de decisiones, por lo que la lectura desplegada sólo incluye una décima de grado de exactitud, mediante un proceso de redondeo. Cabe señalar que los observatorios cuentan con una estación de meteorología que les suministra datos precisos de las condiciones ambientales. Como la computadora de la consola de guiado de estos observatorios se encuentra en el mismo habitación, llamado también cúpula, del telescopio, las lecturas de los sensores que ingresan a la tarjeta de interfaz son similares a los de la estación meteorológica y cumplen solamente una finalidad de redundancia.

De la misma manera, el sensor de humedad utilizado es el obtenido de una antigua videocasetera *Beta*, debido a su disponibilidad y fácil manejo, ya que opera como una

resistencia variable que cambia su resistividad en función de la humedad que se encuentra en el ambiente (a mayor humedad mayor resistencia). Cabe señalar que la elección de este sensor recayó en los investigadores del IAUNAM y el principal factor que influyó en su decisión fue el factor económico, buscando así ahorrar costos en el proyecto. La configuración del sensor de humedad es la de un divisor de tensión. Debido a que el encargado de mover el telescopio sólo necesita observar las tendencias de las mediciones de este sensor para tomar decisiones, el sensor de humedad está polarizado para operar correctamente sólo cuando se está cerca del punto de saturación de humedad, que es cuando existe el cien por ciento de humedad en el ambiente. Esto es debido a que el sensor de humedad no es lineal, sino que es logarítmico. Este compromiso se justifica ya que es muy importante prever la lluvia en los Observatorios, para no dañar la óptica de los telescopios. Las lecturas del sensor de humedad que se despliegan en el monitor de la computadora de la consola de control del telescopio, se expresan en porcentaje.

VI
RESULTADOS Y CONCLUSIONES

En este trabajo se ha presentado una propuesta de diseño de una interfaz electrónica, como parte del control de un telescopio con montura ecuatorial, que puede ser utilizada en algunos de los observatorios operados por el Instituto de Astronomía de la UNAM. Cabe mencionar que este trabajo se desarrolló en estrecha colaboración con dicho Instituto y siempre de acuerdo con sus requerimientos. El diseño presentado está desarrollado para las especificaciones de la consola de control del Observatorio Astronómico Nacional de Tonantzintla y la del Observatorio de la Luz de la Universidad de Guanajuato, por lo que puede encontrar cabida en ellos.

Este proyecto se desarrolló aprovechando una de las tecnologías más recientes en la programación de CPLDs, para proporcionar una alternativa de producción de hardware de fácil entendimiento y acceso, tanto en la planeación como en la rápida obtención de resultados. La realización de una interfaz electrónica de estas características requiere del conocimiento y estudio detallado del comportamiento de los diferentes sistemas y medios que la integran.

El principal objetivo cumplido con esta propuesta de diseño ha sido el de facilitar el incremento del tiempo de utilización de otros controladores de telescopios con los que operan estos observatorios actualmente, concretamente los controladores de los motores de los ejes de los telescopios y sus controladores manuales, mediante una tarjeta que realice satisfactoriamente la interfaz entre el hardware anteriormente instalado y una microcomputadora reciente. De esta manera, estos observatorios podrán seguir operando con los sistemas actuales de control durante otro periodo de tiempo, indefinido, sin realizar mayores inversiones en sus equipos. Cuando los observatorios antes mencionados consideren necesario, el IAUNAM puede instrumentar esta interfaz o una implementación mejorada de ésta, ya que el trabajo desarrollado en este proyecto contempla la primera versión del prototipo.

Otra finalidad que se cumple con el desarrollo de este trabajo, es el asentamiento de las bases para que el Laboratorio de Electrónica de la Sección de Instrumentación del IAUNAM, comience a considerar, a mayor escala, el uso de los CPLDs para la implementación de sus proyectos. Debido a la flexibilidad del medio, la generación de alternativas de diseño es muy libre, esto permite proponer conceptos distintos a los de la cotidianeidad, sin estar sujetos a patrones comunes. La presentación de las distintas variaciones en la programación del CPLD en este proyecto es un ejemplo de ello.

El costo del desarrollo de esta interfaz es de menos de la mitad del valor de la anterior tarjeta, con las ventajas añadidas que ofrecen los sistemas embebidos, principalmente en lo que se refiere a la fiabilidad en el proceso de verificación de las distintas etapas de desarrollo y prueba. Igualmente, el costo de esta interfaz es de menos de la tercera parte del valor de un sistema comercial de características similares, como puede ser la tarjeta de interfaz AT-GPIB/TNT de *National Instruments*, con la ventaja de que la interfaz desarrollada en este trabajo está específicamente diseñada para las funciones de los observatorios.

Dentro de las mejoras posteriores que se pueden realizar al sistema presentado en este trabajo, cabría considerar el desarrollo de esta interfaz utilizando un bus PCI, cuyo uso se encuentra más extendido entre las microcomputadoras actuales, aunque esta posibilidad implicaría la modificación y reestructuración del programa de guiado de la consola de control del telescopio. Igualmente, podría adaptarse la interfaz y los periféricos con los que se comunica, para que opere utilizando las denominadas interrupciones por hardware. Sin embargo, esta opción también implicaría modificaciones en el programa de guiado de la consola de control. Independientemente, podría considerarse la posibilidad de implementar un sensor de humedad más fiable, tal como alguno perteneciente a la familia HIH-3610 de *Honeywell*. De esta manera, podría conseguirse un diseño más sólido, en el sentido de que

el sensor podría ser más fácilmente reemplazado por otro similar, en caso de algún mal funcionamiento, aunque el costo de este sensor sería más elevado.

El creciente aumento en los productos electrónicos utilizados para interactuar con la computadora, hace cada vez más necesario que el ingeniero se involucre más frecuentemente con el desarrollo de productos como éste, creando una perspectiva profesional más amplia. De aquí que, aunque la propuesta de diseño presentada en este trabajo tenga una aplicación limitada a ciertos Observatorios, varios de sus conceptos y utilidades pueden ser explotadas para otras aplicaciones, particularmente aquellas donde se involucren conteos de trenes de pulsos.

La extrapolación de todos los conceptos anteriores, aunados a la inventiva y a la capacidad de adaptación a las circunstancias específicas requeridas, constituyen así la esencia del ingeniero profesionista.

VII
BIBLIOGRAFÍA

Angulo Usategui, José María. **ELECTRÓNICA DIGITAL MODERNA**, Ed. Paraninfo, Decimosexta Edición, Madrid 1996.

Bernal, Abel y Gutiérrez, Leonel. **SISTEMA COMPUTARIZADO DE CONTROL DEL TELESCOPIO DE TONANTZINTLA. LOS PROGRAMAS**, Reporte Técnico RT-95-01, Instituto de Astronomía de la UNAM.

Birmelin, Michael. **MANUAL DE LOS PROCESADORES 80xxx Y PENTIUM**, Marcombo Boixareu Editores, Barcelona 1995.

Brey, Barry. **LOS MICROPROCESADORES INTEL**, Quinta Edición, Pearson Educación, México 2001.

Eggebrecht, Lewis C. **INTERFACING TO THE IBM PERSONAL COMPUTER**, Howard W. Sams & Co., E.E.U.U, 1987.

Encyclopaedia Britannica, Inc., **HOMBRE, CIENCIA Y TECNOLOGÍA**, Editorial Océano, México, 1986.

Fishbane Paul, Gasiorowicz Stephen, Thornton Stephen. **FÍSICA PARA CIENCIAS E INGENIERÍA**, Volumen II, Editorial Prentice-Hall Hispanoamericana, México, 1994.

Hetch, Eugene & Zajac, Alfred. **ÓPTICA**, Fondo Educativo Interamericano, E.E.U.U., 1977.

Horowitz, Paul & Hill, Winfield. **THE ART OF ELECTRONICS**, Cambridge University Press, E.E.U.U., 1989.

Khambata, A.J. **MICROPROCESADORES/ MICROCOMPUTADORES**, Colección Ciencia Electrónica, De. Calypso, México, 1987.

Malvino, Albert Paul. **PRINCIPIOS DE ELECTRÓNICA**, Editorial McGraw-Hill, Quinta Edición, Colombia, 1998.

Mano, M. Morris. **DISEÑO DIGITAL**, Editorial Prentice Hall, México, 1996.

Mompín Poblet, José. **INTERCONEXIÓN DE PERIFÉRICOS A MICROPROCESADORES**, Serie Mundo Electrónico, Publicaciones Marcombo, México, 1984.

Norton, Peter & Wilton, Richard. **PROGRAMMERS GUIDE TO THE IBM PC & PS/2**, Microsoft Press, E.E.U.U., 1988.

Oliver, José María. **MANUAL PRÁCTICO DEL ASTRÓNOMO AFICIONADO**, Editorial De Vecchi, Barcelona 1990.

Restrepo Arredondo, Perta. **GUÍA PARA VIAJEROS DEL CIELO**, Editorial Planeta, México, 1998.

Rükl, Antonin. **THE HAMLYN ENCYCLOPEDIA OF STARS AND PLANETS**, Editorial Hamlyn, Praga, 1988.

THE NEW ENCYCLOPAEDIA BRITANNICA, Décimo quinta Edición, E.E.U.U., 1986.

Tompkins, Willis J. & Webster, John G.(Editores). **INTERFACING SENSORS TO THE IBM PC**, Editorial Prentice Hall., E.E.U.U., 1988.

Walker, Gordon. **ASTRONOMICAL OBSERVATIONS**, Cambridge University Press, E.E.U.U., 1987.

- *MANUALES*

DATABOOK, Altera Corporation, E.E.U.U. 1987

MAX+PLUS II CAD DESIGN SYSTEM, Versión 8.3, Altera Corporation, 1998

PCL-750 PROTOTYPE DEVELOPMENT CARD USER'S MANUAL, Advantech Co., Ltd., Taiwan, 1989.

THE BYPASS CAPACITOR IN HIGH SPEED ENVIRONMENTS, Texas Instruments, SCBA007A, Noviembre 1996

THE PROGRAMMABLE LOGIC DATA BOOK. XILINX. E.E.U.U. 1996

- *REVISTAS*

ASTRONOMY, Kamblach Publishing Co., Vol. 22, No. 1, Enero 1994.

SKY AND TELESCOPE, Sky Publishing Corporation, Vol. 94, No. 1, Julio 1997 y Volumen 94, No. 2, Agosto 1997.

UNIVERSO, Antares Ciencia y Ediciones, Año 2, No. 19, Noviembre 1996.

SABER ELECTRÓNICA, Edición Mexicana, Vallejo, Horacio. Tipos de Fuentes Conmutadas, No.6 Año 11, Junio 2000.

- *DIRECCIONES EN INTERNET*

ALTERA MAX+PLUS II LOGIC DESIGN.
http://www1.uop.edu/eng/courses/elec/elec171/doc/altera/parts.html

CLASES DE DISPOSITIVOS LÓGICOS PROGRAMABLES.
http://calvin.univalle.edu.co/~rubenpal/lecturas/pdls.html

CUPL Starkit, **DEMO MANUAL,**
http://www.logicaldevices.com/cupl/docs/strktmanual.htm

EMBEDDED PENTIUM PROCESSOR FAMILY DEVELOPER'S MANUAL, Intel,
http://developer.intel.com/design/intarch/techinfo/Pentium/

INDEX OF JENKO GLOSSARY. http://hompages.enterprise.net/jenko/Glossary/

INTRODUCCIÓN A LA LÁ LÓGICA RECONFIGURABLE, BRASS-Berkeley Reconfigurable Architectures, Systems and Software, University of California at Berkeley. http://brass.cs.berkeley.edu/ Traducción de Calvo, Germán; Miguez, Juan y Myszne, Jorge.

Instituto de Astronomía de la UNAM, **OBSERVATORIO ASTRONÓMICO NACIONAL**. http://www.astroscu.unam.mx

Johnson, Woody. **BUILD YOUR OWN BYTEBALSTER,**
http://freecore.com/nonsuport/blaster.htm

MAX7000 DEVICES, Altera. http://www.altera.com/products/devices/max7k/m7k-index.html y http://www.altera.com/literature/lit-m7k.html

PÁGINA PRINCIPAL DE ALTERA, Altera Products, http://www.altera.com

Peacock, Craig. **CRAIG PEACKOCK'S INTERFACING THE PC: USING INTERRUPTS**. http://www.senet.com.au/~cpeacock

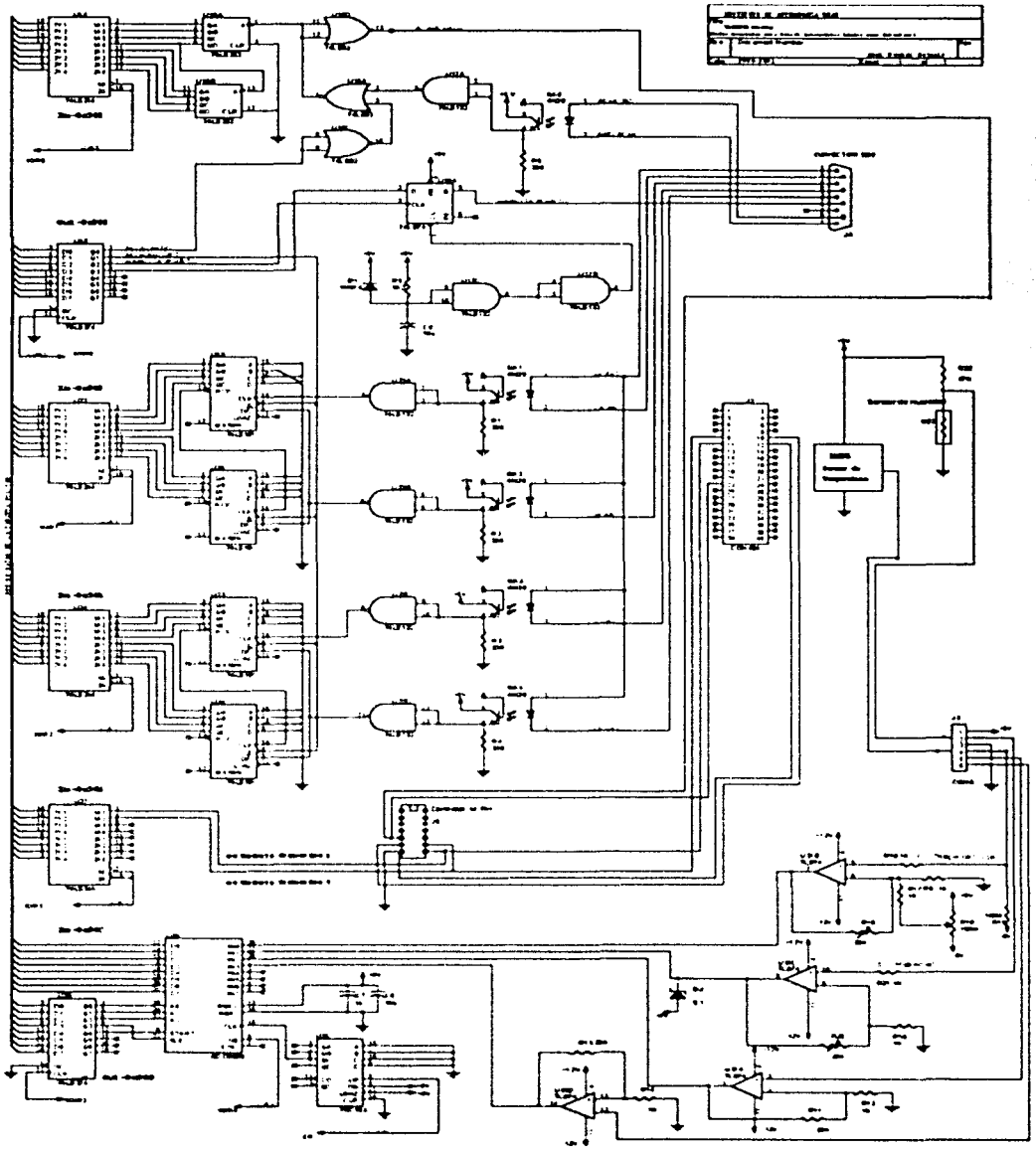
SPECS. SHEET: ADC0808/ADC0809, National Semiconductor. www.national.com

Taylor, Marvin, **THE HISTORY OF THE MICROPROCESSOR.**
http://www.sit.wisc.edu/mptaylor/microprocessor.html

TRANSFERENCIAS VIA EL BUS ISA, PC KITS. http:www.ctv.es/pckits/home.html

APÉNDICE A:

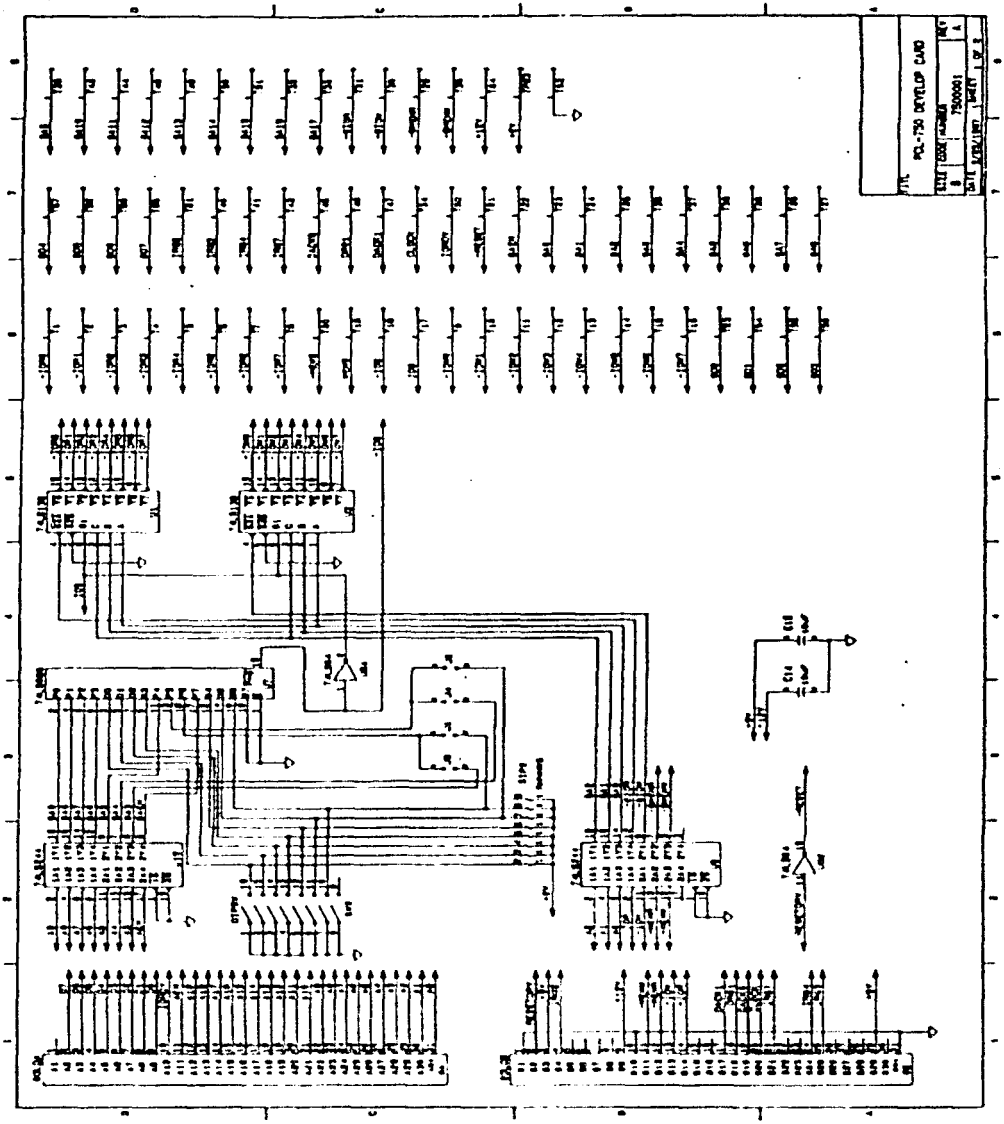
Diagrama de la tarjeta MAGALI



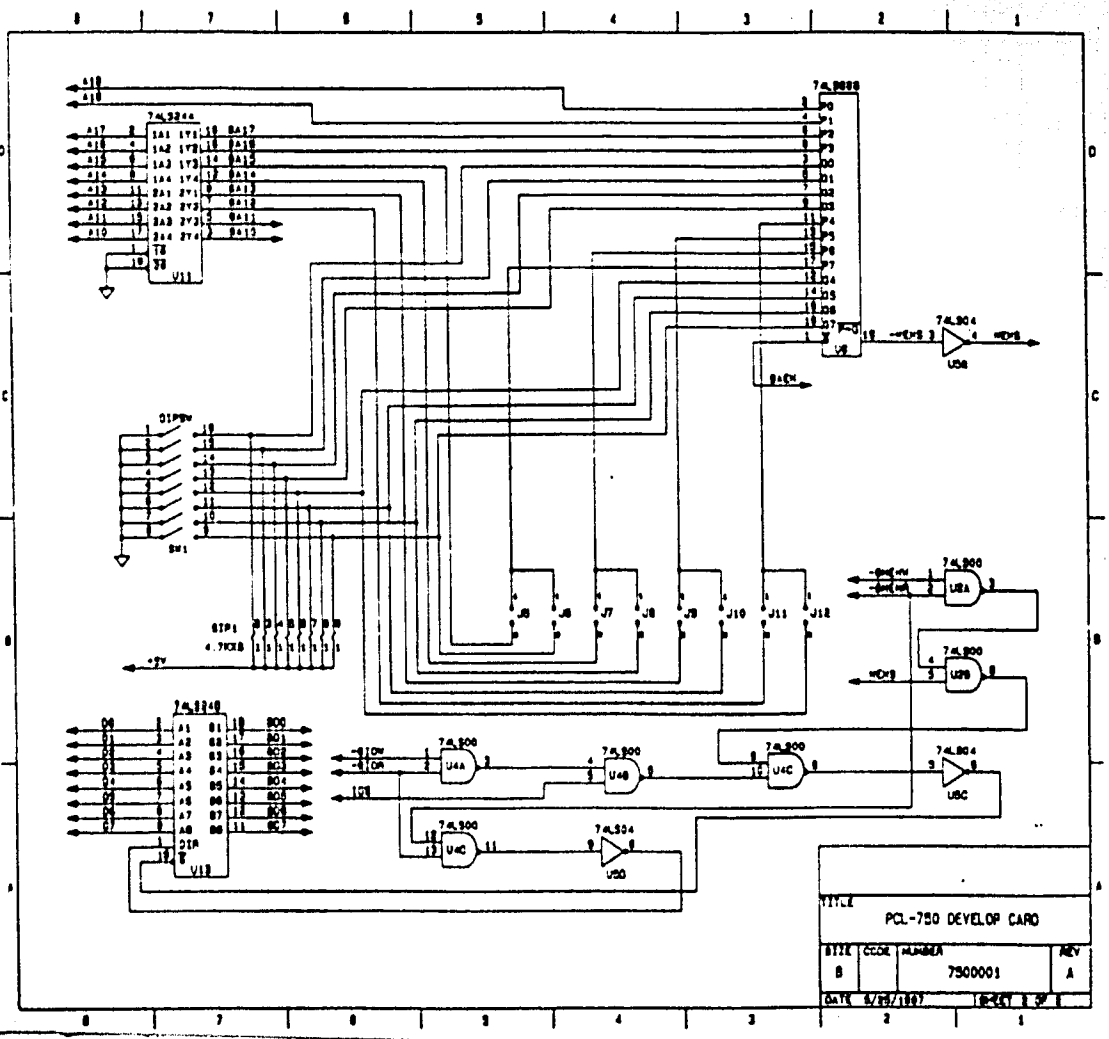
A.3

APÉNDICE B:

Diagrama de la tarjeta PCL-750

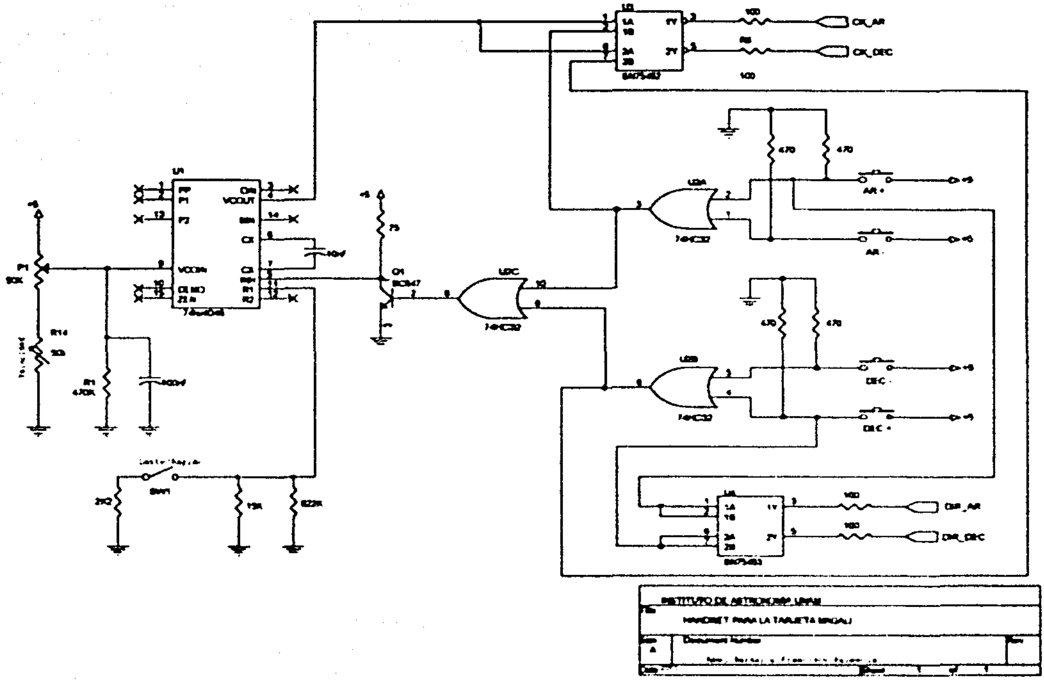


TESIS CON
FALLA DE ORIGEN



APÉNDICE C:

Diagrama del controlador manual del telescopio


INSTITUTO DE ASTRONOMIA UNAM
HANDSET PARA LA TARJETA MIGAL
74HC32
SN75453
CK_AR
CK_DEC
DIR_AR
DIR_DEC
AR +
AR -
DEC -
DEC +
BC547

APÉNDICE D

Programación del CPLD para decodificar y habilitar a puertos y a memoria

Debido a los requisitos de diseño de la interfaz de este trabajo, sólo se utiliza la decodifcación y habilitación de escritura y lectura a puertos mediante las señales IOW e IOR. Sin embargo, también es posible realizar esta habilitación y decodificación a memoria con las señales MEMW y MEMR. Para lograr esto, se necesitan señales de entrada adicionales al CPLD, por lo que el EPM7064LC68 resulta insuficiente. La programación que se presenta a continuación se puede emplear en un EPM7096LC100 el cual cuenta con 100 pines, de los cuales 72 son disponibles para el programador.

Los módulos de programación que difieren de los presentados en la sección 4 son el del control de habilitaciones y el de las decodificaciones. El diagrama esquemático de la lógica utilizada para manejar las señales de control que provienen del *bus* ISA se muestra en la fig. D.1.

La lógica que rige a la señal SENAL_DAT_DIR es función de BUFFER_MEM_READ y BUFFER_MEM_READ que ingresan a una compuerta AND, mientras que la lógica de la señal SENAL_DAT_TRANSFER es función de las señales BUFFER_IO_WRITE, BUFFER_IO_READ, HABILITA_DECOD_DIR, BUFFER_MEM_WRITE, BUFFER_MEM_READ Y HABILITA_DECOD_MEM, como se muestra en la tabla D.1.

Para la lógica de la decodificación de puertos y de memoria se necesita un segundo *dip-switch* para habilitar la señal HABILITA_DECOD_MEM. Esto sucede cuando las señales provenientes del *dip-switch* son las mismas que las que ingresan por las señales A12 a A19 del *bus* de direcciones, como se muestra en el diagrama esquemático de la figura D.2

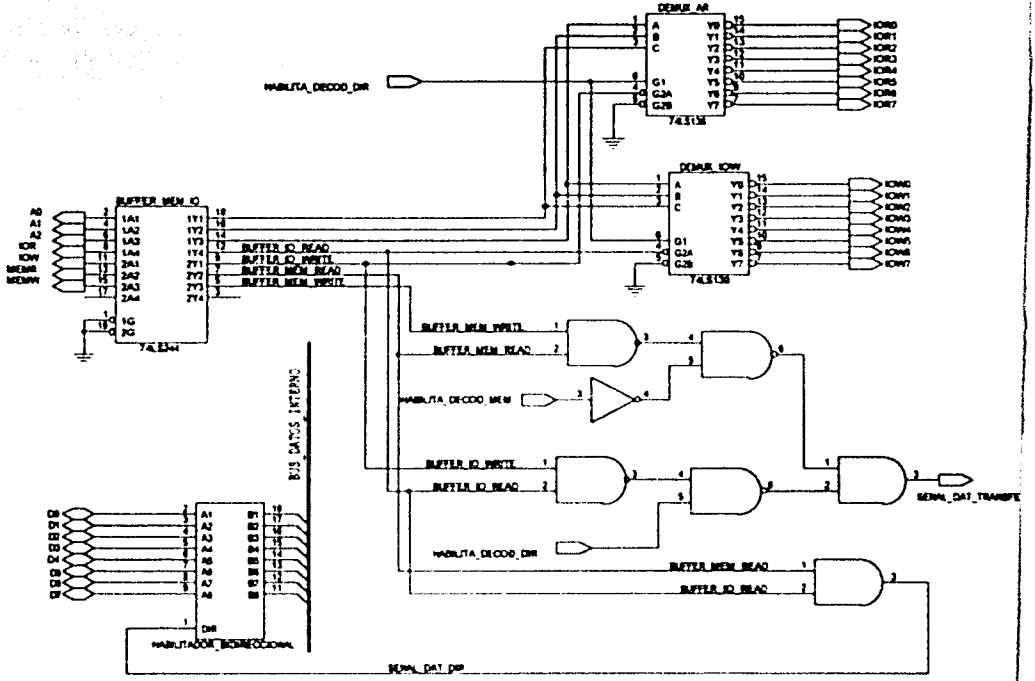


Figura D.1. Módulo de las habilitaciones para la habilitación a puertos y a memoria.

BUFFER IO WRITE	BUFFER IO READ	HABILITA DECOD DIR	BUFFER MEM WRITE	BUFFER MEM READ	HABILITA DECOD MEM	SENAL DAT TRANSFER
0	0	0	0	0	0	0
0	0	0	0	0	1	1
0	0	0	0	1	0	0
0	0	0	0	1	1	1
0	0	0	1	0	0	0
0	0	0	1	0	1	1
0	0	0	1	1	0	1
0	0	0	1	1	1	1
0	0	1	0	0	0	0
0	0	1	0	0	1	0
0	0	1	0	1	0	0
0	0	1	0	1	1	0
0	0	1	1	0	0	0
0	0	1	1	0	1	0
0	0	1	1	1	0	0
0	0	1	1	1	1	0
0	1	0	0	0	0	0
0	1	0	0	0	1	1
0	1	0	0	1	0	0
0	1	0	0	1	1	1
0	1	0	1	0	0	0
0	1	0	1	0	1	1
0	1	0	1	1	0	1
0	1	0	1	1	1	1
0	1	1	0	0	0	0
0	1	1	0	0	1	0
0	1	1	0	1	0	0
0	1	1	0	1	1	0
0	1	1	1	0	0	0
0	1	1	1	0	1	0
0	1	1	1	1	0	0
0	1	1	1	1	1	0
1	0	0	0	0	0	0
1	0	0	0	0	1	1
1	0	0	0	1	0	0

Tabla D.1. Habilitación del *bus* de datos interno del CPLD. (Continúa)

BUFFER IO WRITE	BUFFER IO READ	HABILITA DECOD DIR	BUFFER MEM WRITE	BUFFER MEM READ	HABILITA DECOD MEM	SENAL DAT TRANSFER
1	0	0	0	1	1	1
1	0	0	1	0	0	0
1	0	0	1	0	1	1
1	0	0	1	1	0	1
1	0	0	1	1	1	1
1	0	1	0	0	0	0
1	0	1	0	0	1	0
1	0	1	0	1	0	0
1	0	1	0	1	1	0
1	0	1	1	0	0	0
1	0	1	1	0	1	0
1	0	1	1	1	0	0
1	0	1	1	1	1	0
1	1	0	0	0	0	0
1	1	0	0	0	1	1
1	1	0	0	1	0	0
1	1	0	0	1	1	1
1	1	0	1	0	0	0
1	1	0	1	0	1	1
1	1	0	1	1	0	1
1	1	0	1	1	1	1
1	1	1	0	0	0	0
1	1	1	0	0	1	1
1	1	1	0	1	0	0
1	1	1	0	1	1	1
1	1	1	1	0	0	0
1	1	1	1	0	1	1
1	1	1	1	1	0	1
1	1	1	1	1	1	1

Tabla D.1. Habilitación del *bus* de datos interno del CPLD.

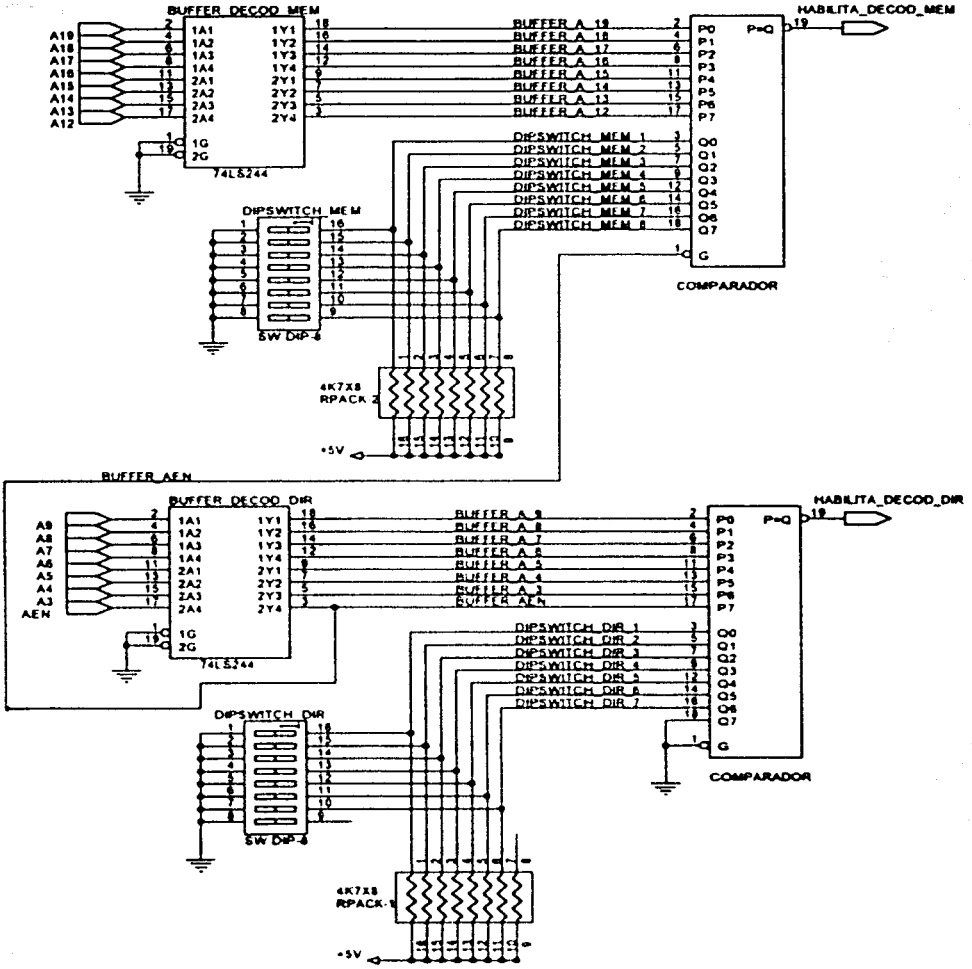


Figura D.2. Módulo de las habilitaciones para la decodificación de puertos y de memoria.

Las señales provenientes de estos módulos, de tal manera que puedan interactuar con el resto de la interfaz, se relacionan como se muestra en la fig. D.3.

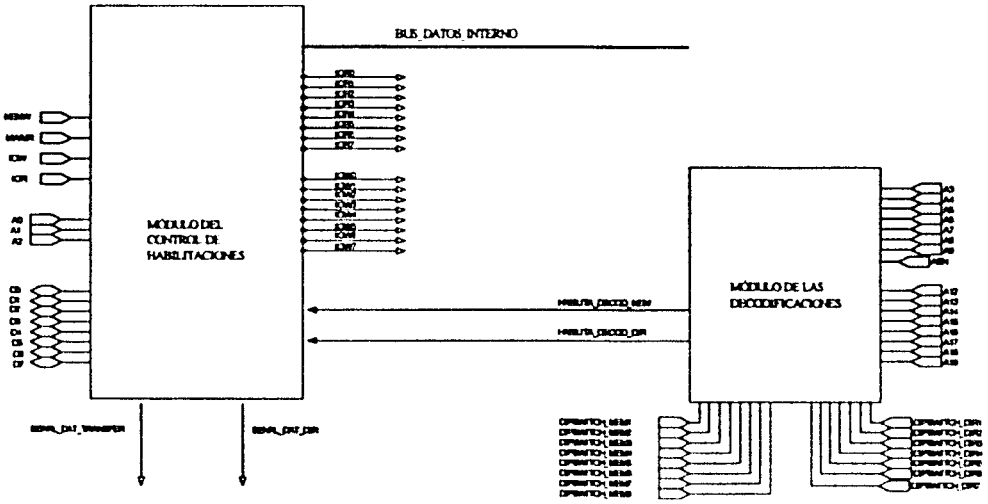


Figura D.3. Relación entre los módulos programados en el CPLD integrados en la interfaz ampliada.

De esta manera, la programación de los módulos de habilitación y de decodificación son los siguientes:

1) Módulo de control de habilitaciones utilizando IOR, IOW, MEMR y MEMW.-

```
%MÓDULO DEL CONTROL DE HABILITACIONES UTILIZANDO IO Y MEM%

INCLUDE "74244";
INCLUDE "74138";
OPTIONS BIT0=ANY;
```

```
subdesign X_MODULO_HABILITA_IO_MEM
(
A[2..0]:INPUT;
MEMW,MEMR,IOW_,IOR_:INPUT;
D[7..0]:BIDIR;
HABILITA_DECOD_DIR:INPUT;              %EN EL GLOBAL ES UN NODO%
HABILITA_DECOD_MEM:INPUT;              %EN EL GLOBAL ES UN NODO%
IOR[7..0]:OUTPUT;                  %EN EL GLOBAL SON NODOS%
IOW[7..0]:OUTPUT;                  %EN EL GLOBAL SON NODOS%
SENAL_DAT_TRANSFER:OUTPUT;         %EN EL GLOBAL SON NODOS%

ENTRADA_PRUEBA[7..0]:INPUT;            %VARIABLE DE PRUEBA. NO VA EN EL
GLOBAL%
HABILITA_SALIDA_DATOS:INPUT;       %VARIABLE DE PRUEBA. NO VA EN EL GLOBAL%

)

VARIABLE
BUFFER_MEM_IO:74244;
DEMUX_IOR,DEMUX_IOW:74138;

BUFFER_DAT_IN 74244;       %SOLO PARA PRUEBA, NO VA EN EL GLOBAL%

BUS_DATOS_INTERNO[7..0] TRI_STATE_NODE;
BUFFER_IO_WRITE,BUFFER_IO_READ NODE;
BUFFER_MEM_WRITE,BUFFER_MEM_READ NODE;
SENAL_DAT_DIR NODE;

BEGIN

%PARTE DEL CONTROL DE IOW E IOR%

BUFFER_MEM_IO 1A[1]=A[2];
BUFFER_MEM_IO 1A[2]=A[1];
BUFFER_MEM_IO 1A[3]=A[0];
BUFFER_MEM_IO 1A[4]=IOR_;
BUFFER_MEM_IO 2A[1]=IOW_;
BUFFER_MEM_IO 2A[2]=MEMR;
BUFFER_MEM_IO 2A[3]=MEMW;

BUFFER_MEM_IO 1GN=GND;
BUFFER_MEM_IO 2GN=GND;

BUFFER_IO_READ=BUFFER_MEM_IO 1Y[4];
BUFFER_IO_WRITE=BUFFER_MEM_IO 2Y[1];
BUFFER_MEM_READ=BUFFER_MEM_IO 2Y[2];
BUFFER_MEM_WRITE=BUFFER_MEM_IO 2Y[3];


DEMUX_IOR.A=BUFFER_MEM_IO 1Y[3];
DEMUX_IOR.B=BUFFER_MEM_IO 1Y[2];
```

```
DEMUX_IOR.C=BUFFER_MEM_IO.1Y[1];
DEMUX_IOR.G2AN=BUFFER_MEM_IO.2Y[1];
DEMUX_IOR.G2BN=GND;
DEMUX_IOR.G1=HABILITA_DECOD_DIR;

DEMUX_IOW.A=BUFFER_MEM_IO.1Y[3];
DEMUX_IOW.B=BUFFER_MEM_IO.1Y[2];
DEMUX_IOW.C=BUFFER_MEM_IO.1Y[1];
DEMUX_IOW.G2AN=BUFFER_MEM_IO.1Y[4];
DEMUX_IOW.G2BN=GND;
DEMUX_IOW.G1=HABILITA_DECOD_DIR;

IOR0=DEMUX_IOR.Y0N;
IOR1=DEMUX_IOR.Y1N;
IOR2=DEMUX_IOR.Y2N;
IOR3=DEMUX_IOR.Y3N;
IOR4=DEMUX_IOR.Y4N;
IOR5=DEMUX_IOR.Y5N;
IOR6=DEMUX_IOR.Y6N;
IOR7=DEMUX_IOR.Y7N;

IOW0=DEMUX_IOW.Y0N;
IOW1=DEMUX_IOW.Y1N;
IOW2=DEMUX_IOW.Y2N;
IOW3=DEMUX_IOW.Y3N;
IOW4=DEMUX_IOW.Y4N;
IOW5=DEMUX_IOW.Y5N;
IOW6=DEMUX_IOW.Y6N;
IOW7=DEMUX_IOW.Y7N;

%PARTE DE LAS COMPUERTAS%

SENAL_DAT_DIR=BUFFER_IO_READ AND BUFFER_MEM_READ;

TABLE
BUFFER_IO_WRITE,BUFFER_IO_READ,HABILITA_DECOD_DIR,BUFFER_MEM_WRITE,BUF
FER_MEM_READ,HABILITA_DECOD_MEM=>SENAL_DAT_TRANSFER;

0,0,0,0,0,0=>0;
0,0,0,0,0,1=>1;
0,0,0,0,1,0=>0;
0,0,0,0,1,1=>1;
0,0,0,1,0,0=>0;
0,0,0,1,0,1=>1;
0,0,0,1,1,0=>1;
0,0,0,1,1,1=>1;
0,0,1,0,0,0=>0;
0,0,1,0,0,1=>0;
0,0,1,0,1,0=>0;
0,0,1,0,1,1=>0;
0,0,1,1,0,0=>0;
```

0,0,1,1,0,1=>0;
0,0,1,1,1,0=>0;
0,0,1,1,1,1=>0;
0,1,0,0,0,0=>0;
0,1,0,0,0,1=>1;
0,1,0,0,1,0=>0;
0,1,0,0,1,1=>1;
0,1,0,1,0,0=>0;
0,1,0,1,0,1=>1;
0,1,0,1,1,0=>1;
0,1,0,1,1,1=>1;
0,1,1,0,0,0=>0;
0,1,1,0,0,1=>0;
0,1,1,0,1,0=>0;
0,1,1,0,1,1=>0;
0,1,1,1,0,0=>0;
0,1,1,1,0,1=>0;
0,1,1,1,1,0=>0;
0,1,1,1,1,1=>0;
1,0,0,0,0,0=>0;
1,0,0,0,0,1=>1;
1,0,0,0,1,0=>0;
1,0,0,0,1,1=>1;
1,0,0,1,0,0=>0;
1,0,0,1,0,1=>1;
1,0,0,1,1,0=>1;
1,0,0,1,1,1=>1;
1,0,1,0,0,0=>0;
1,0,1,0,0,1=>0;
1,0,1,0,1,0=>0;
1,0,1,0,1,1=>0;
1,0,1,1,0,0=>0;
1,0,1,1,0,1=>0;
1,0,1,1,1,0=>0;
1,0,1,1,1,1=>0;
1,1,0,0,0,0=>0;
1,1,0,0,0,1=>1;
1,1,0,0,1,0=>0;
1,1,0,0,1,1=>1;
1,1,0,1,0,0=>0;
1,1,0,1,0,1=>1;
1,1,0,1,1,0=>1;
1,1,0,1,1,1=>1;
1,1,1,0,0,0=>0;
1,1,1,0,0,1=>1;
1,1,1,0,1,0=>0;
1,1,1,0,1,1=>1;
1,1,1,1,0,0=>0;
1,1,1,1,0,1=>1;
1,1,1,1,1,0=>1;
1,1,1,1,1,1=>1;

```
END TABLE;

%PARTE DE CONTROL DEL BUS%

D[0]=TRI(BUS_DATOS_INTERNO[0],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[1]=TRI(BUS_DATOS_INTERNO[1],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[2]=TRI(BUS_DATOS_INTERNO[2],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[3]=TRI(BUS_DATOS_INTERNO[3],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[4]=TRI(BUS_DATOS_INTERNO[4],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[5]=TRI(BUS_DATOS_INTERNO[5],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[6]=TRI(BUS_DATOS_INTERNO[6],SENAL_DAT_DIR & HABILITA_DECOD_DIR);
D[7]=TRI(BUS_DATOS_INTERNO[7],SENAL_DAT_DIR & HABILITA_DECOD_DIR);

%PARTE DE BUS_DATOS_INTERNO PARA PRUEBA, NO VA EN EL GLOBAL%

BUFFER_DAT_IN.1A[1]=ENTRADA_PRUEBA[0].
BUFFER_DAT_IN.1A[2]=ENTRADA_PRUEBA[1].
BUFFER_DAT_IN.1A[3]=ENTRADA_PRUEBA[2].
BUFFER_DAT_IN.1A[4]=ENTRADA_PRUEBA[3].
BUFFER_DAT_IN.2A[1]=ENTRADA_PRUEBA[4].
BUFFER_DAT_IN.2A[2]=ENTRADA_PRUEBA[5].
BUFFER_DAT_IN.2A[3]=ENTRADA_PRUEBA[6].
BUFFER_DAT_IN.2A[4]=ENTRADA_PRUEBA[7].

BUS_DATOS_INTERNO[0]=TRI(BUFFER_DAT_IN.1Y[1],HABILITA_SALIDA_DATOS);
BUS_DATOS_INTERNO[1]=TRI(BUFFER_DAT_IN.1Y[2],HABILITA_SALIDA_DATOS);
BUS_DATOS_INTERNO[2]=TRI(BUFFER_DAT_IN.1Y[3],HABILITA_SALIDA_DATOS);
BUS_DATOS_INTERNO[3]=TRI(BUFFER_DAT_IN.1Y[4],HABILITA_SALIDA_DATOS);
BUS_DATOS_INTERNO[4]=TRI(BUFFER_DAT_IN.2Y[1],HABILITA_SALIDA_DATOS);
BUS_DATOS_INTERNO[5]=TRI(BUFFER_DAT_IN.2Y[2],HABILITA_SALIDA_DATOS);
BUS_DATOS_INTERNO[6]=TRI(BUFFER_DAT_IN.2Y[3],HABILITA_SALIDA_DATOS);
BUS_DATOS_INTERNO[7]=TRI(BUFFER_DAT_IN.2Y[4],HABILITA_SALIDA_DATOS);

END;
```

2) Módulo de las decodificaciones utilizando A3 a A9 y A12 a A19.-

```
%MODULO DE LAS DECODIFICACIONES UTILIZANDO IO Y MEM%

INCLUDE"74244";
OPTIONS BIT0=ANY;

SUBDESIGN X_MODULO_DECOD_IO_MEM
(
A[9..3],AEN:INPUT;                          %BUS DE DIRS DEL BUS ISA%
DIPSWITCH_DIR[7..1]:INPUT;
A_[19..12]:INPUT;                           %BUS DE DIRS DEL BUS ISA%
DIPSWITCH_MEM[8..1]:INPUT;
```

```
HABILITA_DECOD_MEM:OUTPUT;                %EN EL GLOBAL ES UN NODO%
HABILITA_DECOD_DIR:OUTPUT;                %EN EL GLOBAL ES UN NODO%
)

VARIABLE
BUFFER_DECOD_DIR:74244;
BUFFER_A[9..3]:NODE;
BUFFER_AEN:NODE;

BUFFER_DECOD_MEM:74244;
BUFFER_A_[19..12]:NODE;

BEGIN

%PARTE DE DECODIFICACIÓN DE DIRECCIÓN%

BUFFER_DECOD_DIR.1A[1]=A[9],
BUFFER_DECOD_DIR.1A[2]=A[8],
BUFFER_DECOD_DIR.1A[3]=A[7],
BUFFER_DECOD_DIR.1A[4]=A[6],
BUFFER_DECOD_DIR.2A[1]=A[5],
BUFFER_DECOD_DIR.2A[2]=A[4],
BUFFER_DECOD_DIR.2A[3]=A[3],
BUFFER_DECOD_DIR.2A[4]=AEN,

BUFFER_A[3]=BUFFER_DECOD_DIR.1Y[1],
BUFFER_A[4]=BUFFER_DECOD_DIR.1Y[2],
BUFFER_A[5]=BUFFER_DECOD_DIR.1Y[3],
BUFFER_A[6]=BUFFER_DECOD_DIR.1Y[4],
BUFFER_A[7]=BUFFER_DECOD_DIR.2Y[1],
BUFFER_A[8]=BUFFER_DECOD_DIR.2Y[2],
BUFFER_A[9]=BUFFER_DECOD_DIR.2Y[3],

BUFFER_AEN=BUFFER_DECOD_DIR.2Y[4],

IF ((DIPSWITCH_DIR[7..1] == BUFFER_A[9..3]) & BUFFER_AEN==0) THEN
        HABILITA_DECOD_DIR=B"1",
        ELSE
        HABILITA_DECOD_DIR=B"0",
        END IF,

%PARTE DE DECODIFICACIÓN DE MEMORIA%

BUFFER_DECOD_MEM.1A[1]=A_[19],
BUFFER_DECOD_MEM.1A[2]=A_[18],
BUFFER_DECOD_MEM.1A[3]=A_[17],
BUFFER_DECOD_MEM.1A[4]=A_[16],
BUFFER_DECOD_MEM.2A[1]=A_[15],
BUFFER_DECOD_MEM.2A[2]=A_[14],
BUFFER_DECOD_MEM.2A[3]=A_[13],
BUFFER_DECOD_MEM.2A[4]=A_[12],
```

```
BUFFER_A_[12]=BUFFER_DECOD_MEM.1Y[1];
BUFFER_A_[13]=BUFFER_DECOD_MEM.1Y[2];
BUFFER_A_[14]=BUFFER_DECOD_MEM.1Y[3];
BUFFER_A_[15]=BUFFER_DECOD_MEM.1Y[4];
BUFFER_A_[16]=BUFFER_DECOD_MEM.2Y[1];
BUFFER_A_[17]=BUFFER_DECOD_MEM.2Y[2];
BUFFER_A_[18]=BUFFER_DECOD_MEM.2Y[3];
BUFFER_A_[19]=BUFFER_DECOD_MEM.2Y[4];

IF ((DIPSWITCH_MEM[8..1] == BUFFER_A_[19..12]) & BUFFER_AEN==0) THEN
        HABILITA_DECOD_MEM=B"1";
        ELSE
        HABILITA_DECOD_MEM=B"0";
        END IF;

END;
```

APÉNDICE E:

Hojas de especificaciones del EPM7064LC68

MAX 7000

Programmable Logic Device Family

November 2001, ver. 6.3 | Data Sheet

Features...

- High-performance, EEPROM-based programmable logic devices (PLDs) based on second-generation MAX® architecture
- 5.0-V in-system programmability (ISP) through the built-in IEEE Std. 1149.1 Joint Test Action Group (JTAG) interface available in MAX 7000S devices
  - ISP circuitry compatible with IEEE Std. 1532
- Includes 5.0-V MAX 7000 devices and 5.0-V ISP-based MAX 7000S devices
- Built-in JTAG boundary-scan test (BST) circuitry in MAX 7000S devices with 128 or more macrocells
- Complete EPLD family with logic densities ranging from 600 to 5,000 usable gates (see Tables 1 and 2)
- 5-ns pin-to-pin logic delays with up to 175.4 MHz counter frequencies (including interconnect)
- PCI-compliant devices available

For information on in-system programmable 3.3-V MAX 7000A or 2.5-V MAX 7000B devices, see the *MAX 7000A Programmable Logic Device Family Data Sheet* or the *MAX 7000B Programmable Logic Device Family Data Sheet*.

Table 1. MAX 7000 Device Features

Feature	EPM7032	EPM7064	EPM7096	EPM7128E	EPM7160E	EPM7192E	EPM7256E
Usable gates	600	1,250	1,800	2,500	3,200	3,750	5,000
Macrocells	32	64	96	128	160	192	256
Logic array blocks	2	4	6	8	10	12	16
Maximum user I/O pins	36	68	76	100	104	124	164
t_{PD} (ns)	6	6	7.5	7.5	10	12	12
t_{SU} (ns)	5	5	6	6	7	7	7
t_{FSU} (ns)	2.5	2.5	3	3	3	3	3
t_{CO1} (ns)	4	4	4.5	4.5	5	6	6
f_{CNT} (MHz)	151.5	151.5	125.0	125.0	100.0	90.9	90.9

Table 2. MAX 7000S Device Features

Feature	EPM7032S	EPM7064S	EPM7128S	EPM7160S	EPM7192S	EPM7256S
Usable gates	600	1,250	2,500	3,200	3,750	5,000
Macrocells	32	64	128	160	192	256
Logic array blocks	2	4	8	10	12	16
Maximum user I/O pins	36	68	100	104	124	164
t_{PD} (ns)	5	5	6	6	7.5	7.5
t_{SU} (ns)	2.9	2.9	3.4	3.4	4.1	3.9
t_{FSU} (ns)	2.5	2.5	2.5	2.5	3	3
t_{CO1} (ns)	3.2	3.2	4	3.9	4.7	4.7
f_{CNT} (MHz)	175.4	175.4	147.1	149.3	125.0	128.2

...and More Features

- Open-drain output option in MAX 7000S devices
- Programmable macrocell flipflops with individual clear, preset, clock, and clock enable controls
- Programmable power-saving mode for a reduction of over 50% in each macrocell
- Configurable expander product-term distribution, allowing up to 32 product terms per macrocell
- 44 to 208 pins available in plastic J-lead chip carrier (PLCC), ceramic pin-grid array (PGA), plastic quad flat pack (PQFP), power quad flat pack (RQFP), and 1.0-mm thin quad flat pack (TQFP) packages
- Programmable security bit for protection of proprietary designs
- 3.3-V or 5.0-V operation
  - MultiVolt™ I/O interface operation, allowing devices to interface with 3.3-V or 5.0-V devices (MultiVolt I/O operation is not available in 44-pin packages)
  - Pin compatible with low-voltage MAX 7000A and MAX 7000B devices
- Enhanced features available in MAX 7000E and MAX 7000S devices
  - Six pin- or logic-driven output enable signals
  - Two global clock signals with optional inversion
  - Enhanced interconnect resources for improved routability
  - Fast input setup times provided by a dedicated path from I/O pin to macrocell registers
  - Programmable output slew-rate control
- Software design support and automatic place-and-route provided by Altera's development system for Windows-based PCs and Sun SPARCstation, and HP 9000 Series 700/800 workstations

- Additional design entry and simulation support provided by EDIF 2 0 0 and 3 0 0 netlist files, library of parameterized modules (LPM), Verilog HDL, VHDL, and other interfaces to popular EDA tools from manufacturers such as Cadence, Exemplar Logic, Mentor Graphics, OrCAD, Synopsys, and VeriBest
- Programming support
  - Altera's Master Programming Unit (MPU) and programming hardware from third-party manufacturers program all MAX 7000 devices
  - The BitBlaster™ serial download cable, ByteBlasterMV™ parallel port download cable, and MasterBlaster™ serial/universal serial bus (USB) download cable program MAX 7000S devices

General Description

The MAX 7000 family of high-density, high-performance PLDs is based on Altera's second-generation MAX architecture. Fabricated with advanced CMOS technology, the EEPROM-based MAX 7000 family provides 600 to 5,000 usable gates, ISP, pin-to-pin delays as fast as 5 ns, and counter speeds of up to 175.4 MHz. MAX 7000S devices in the -5, -6, -7, and -10 speed grades as well as MAX 7000 and MAX 7000E devices in -5, -6, -7, -10P, and -12P speed grades comply with the PCI Special Interest Group (PCI SIG) *PCI Local Bus Specification, Revision 2.2*. See Table 3 for available speed grades.

Table 3. MAX 7000 Speed Grades

Device	Speed Grade									
	-5	-6	-7	-10P	-10	-12P	-12	-15	-15T	-20
EPM7032		✓	✓		✓		✓	✓	✓	
EPM7032S	✓	✓	✓		✓					
EPM7064		✓	✓		✓		✓	✓		
EPM7064S	✓	✓	✓		✓					
EPM7096			✓		✓		✓	✓		
EPM7128E			✓	✓	✓		✓	✓		✓
EPM7128S		✓	✓		✓			✓		
EPM7160E				✓	✓		✓	✓		✓
EPM7160S		✓	✓		✓			✓		
EPM7192E						✓	✓	✓		✓
EPM7192S			✓		✓			✓		
EPM7256E						✓	✓	✓		✓
EPM7256S			✓		✓			✓		

The MAX 7000E devices—including the EPM7128E, EPM7160E, EPM7192E, and EPM7256E devices—have several enhanced features: additional global clocking, additional output enable controls, enhanced interconnect resources, fast input registers, and a programmable slew rate.

In-system programmable MAX 7000 devices—called MAX 7000S devices—include the EPM7032S, EPM7064S, EPM7128S, EPM7160S, EPM7192S, and EPM7256S devices. MAX 7000S devices have the enhanced features of MAX 7000E devices as well as JTAG BST circuitry in devices with 128 or more macrocells, ISP, and an open-drain output option. See Table 4.

Table 4. MAX 7000 Device Features

Feature	EPM7032 EPM7064 EPM7096	All MAX 7000E Devices	All MAX 7000S Devices
ISP via JTAG interface			✓
JTAG BST circuitry			✓ (1)
Open-drain output option			✓
Fast input registers		✓	✓
Six global output enables		✓	✓
Two global clocks		✓	✓
Slew-rate control		✓	✓
MultiVolt interface (2)	✓	✓	✓
Programmable register	✓	✓	✓
Parallel expanders	✓	✓	✓
Shared expanders	✓	✓	✓
Power-saving mode	✓	✓	✓
Security bit	✓	✓	✓
PCI-compliant devices available	✓	✓	✓

Notes:
(1) Available only in EPM7128S, EPM7160S, EPM7192S, and EPM7256S devices only.
(2) The MultiVolt I/O interface is not available in 44-pin packages.

The MAX 7000 architecture supports 100% TTL emulation and high-density integration of SSI, MSI, and LSI logic functions. The MAX 7000 architecture easily integrates multiple devices ranging from PALs, GALs, and 22V10s to MACH and pLSI devices. MAX 7000 devices are available in a wide range of packages, including PLCC, PGA, PQFP, RQFP, and TQFP packages. See Table 5.

Table 5. MAX 7000 Maximum User I/O Pins *Note (1)*

Device	44-Pin PLCC	44-Pin PQFP	44-Pin TQFP	68-Pin PLCC	84-Pin PLCC	100-Pin PQFP	100-Pin TQFP	160-Pin PQFP	160-Pin PGA	192-Pin PGA	208-Pin PQFP	208-Pin RQFP
EPM7032	36	36	36									
EPM7032S	36		36									
EPM7064	36		36	52	68	68						
EPM7064S	36		36		68		68					
EPM7096				52	64	76						
EPM7128E					68	84		100				
EPM7128S					68	84	84 *(2)*	100				
EPM7160E					64	84		104				
EPM7160S					64		84 *(2)*	104				
EPM7192E								124	124			
EPM7192S								124				
EPM7256E								132 *(2)*		164		164
EPM7256S											164 *(2)*	164

Notes:

(1) When the JTAG interface in MAX 7000S devices is used for either boundary-scan testing or for ISP, four I/O pins become JTAG pins.

(2) Perform a complete thermal analysis before committing a design to this device package. For more information, see the *Operating Requirements for Altera Devices Data Sheet*.

MAX 7000 devices use CMOS EEPROM cells to implement logic functions. The user-configurable MAX 7000 architecture accommodates a variety of independent combinatorial and sequential logic functions. The devices can be reprogrammed for quick and efficient iterations during design development and debug cycles, and can be programmed and erased up to 100 times.

MAX 7000 devices contain from 32 to 256 macrocells that are combined into groups of 16 macrocells, called logic array blocks (LABs). Each macrocell has a programmable-AND/fixed-OR array and a configurable register with independently programmable clock, clock enable, clear, and preset functions. To build complex logic functions, each macrocell can be supplemented with both shareable expander product terms and high-speed parallel expander product terms to provide up to 32 product terms per macrocell.

The MAX 7000 family provides programmable speed/power optimization. Speed-critical portions of a design can run at high speed/full power, while the remaining portions run at reduced speed/low power. This speed/power optimization feature enables the designer to configure one or more macrocells to operate at 50% or lower power while adding only a nominal timing delay. MAX 7000E and MAX 7000S devices also provide an option that reduces the slew rate of the output buffers, minimizing noise transients when non-speed-critical signals are switching. The output drivers of all MAX 7000 devices (except 44-pin devices) can be set for either 3.3-V or 5.0-V operation, allowing MAX 7000 devices to be used in mixed-voltage systems.

The MAX 7000 family is supported by Altera development systems, which are integrated packages that offer schematic, text—including VHDL, Verilog HDL, and the Altera Hardware Description Language (AHDL)—and waveform design entry, compilation and logic synthesis, simulation and timing analysis, and device programming. The software provides EDIF 2 0 0 and 3 0 0, LPM, VHDL, Verilog HDL, and other interfaces for additional design entry and simulation support from other industry-standard PC- and UNIX-workstation-based EDA tools. The software runs on Windows-based PCs, as well as Sun SPARCstation, and HP 9000 Series 700/800 workstations.

For more information on development tools, see the *MAX+PLUS II Programmable Logic Development System & Software Data Sheet* and the *Quartus Programmable Logic Development System & Software Data Sheet*.

Functional Description

The MAX 7000 architecture includes the following elements:

- Logic array blocks
- Macrocells
- Expander product terms (shareable and parallel)
- Programmable interconnect array
- I/O control blocks

The MAX 7000 architecture includes four dedicated inputs that can be used as general-purpose inputs or as high-speed, global control signals (clock, clear, and two output enable signals) for each macrocell and I/O pin. Figure 1 shows the architecture of EPM7032, EPM7064, and EPM7096 devices.

Figure 1. EPM7032, EPM7064 & EPM7096 Device Block Diagram

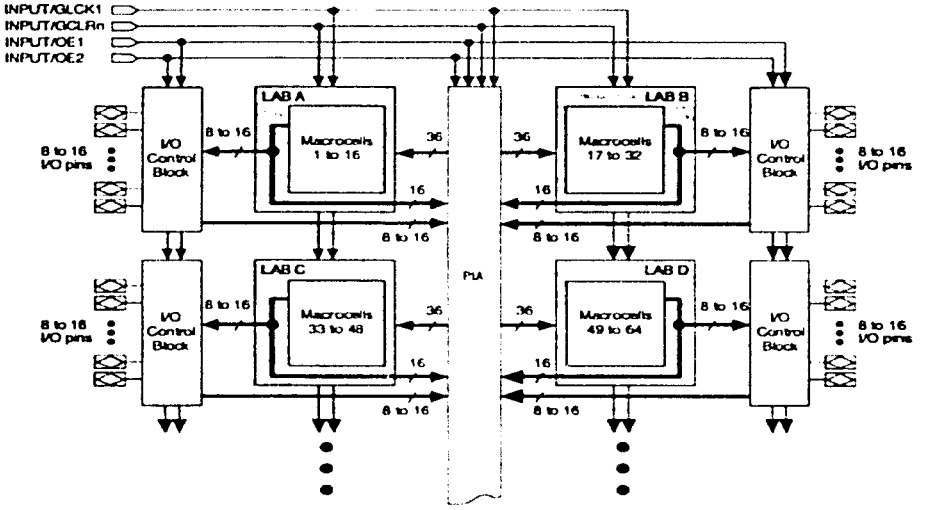
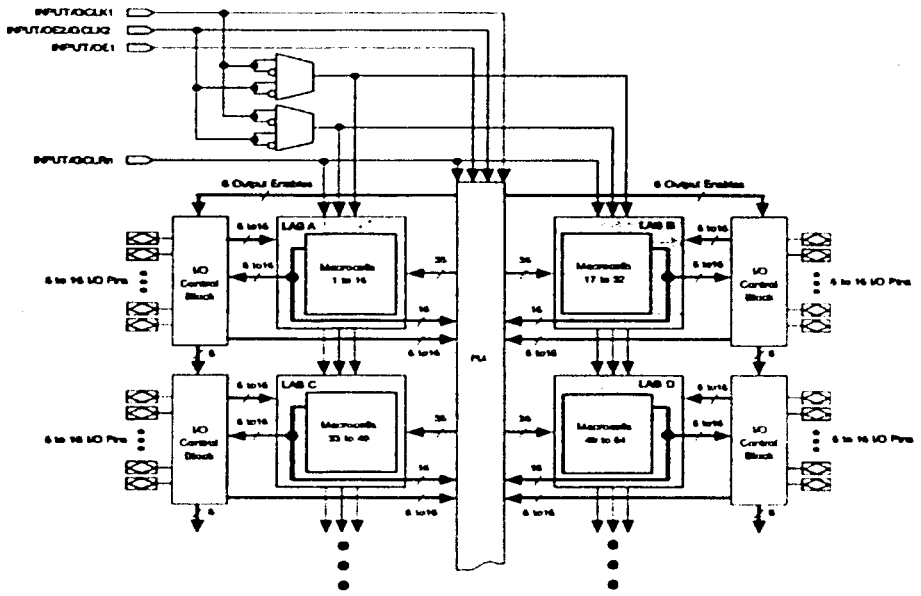


Figure 2 shows the architecture of MAX 7000E and MAX 7000S devices.

Figure 2. MAX 7000E & MAX 7000S Device Block Diagram



Logic Array Blocks

The MAX 7000 device architecture is based on the linking of high-performance, flexible, logic array modules called logic array blocks (LABs). LABs consist of 16-macrocell arrays, as shown in Figures 1 and 2. Multiple LABs are linked together via the programmable interconnect array (PIA), a global bus that is fed by all dedicated inputs, I/O pins, and macrocells.

Each LAB is fed by the following signals:

- 36 signals from the PIA that are used for general logic inputs
- Global controls that are used for secondary register functions
- Direct input paths from I/O pins to the registers that are used for fast setup times for MAX 7000E and MAX 7000S devices

Macrocells

The MAX 7000 macrocell can be individually configured for either sequential or combinatorial logic operation. The macrocell consists of three functional blocks: the logic array, the product-term select matrix, and the programmable register. The macrocell of EPM7032, EPM7064, and EPM7096 devices is shown in Figure 3.

Figure 3. EPM7032, EPM7064 & EPM7096 Device Macrocell

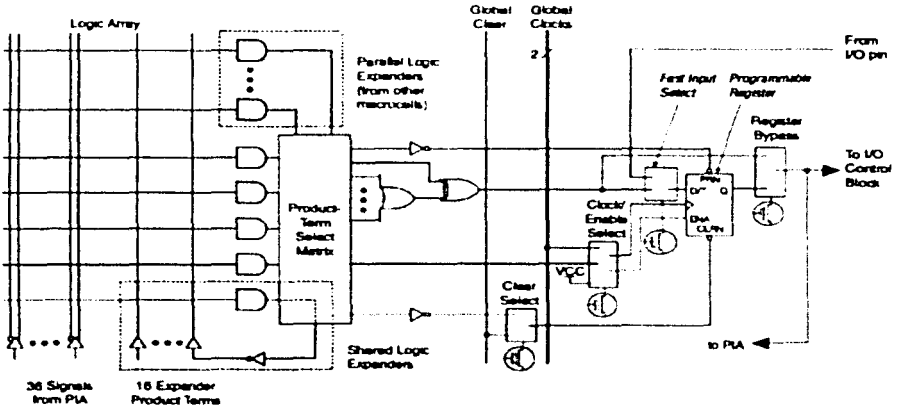
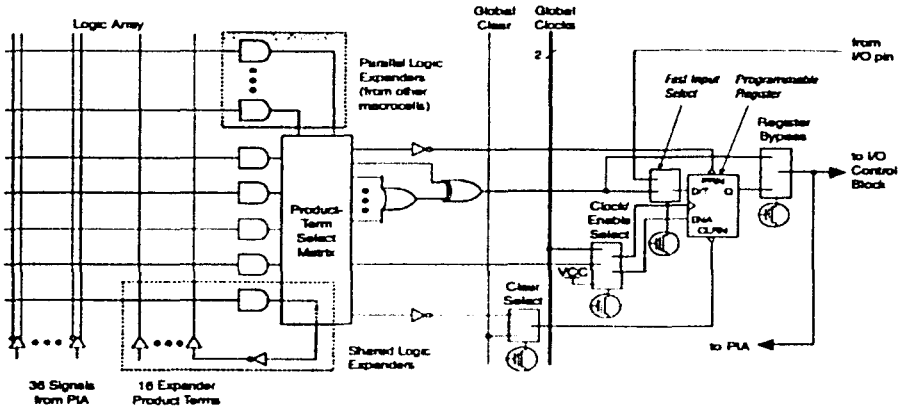


Figure 4 shows a MAX 7000E and MAX 7000S device macrocell.

Figure 4. MAX 7000E & MAX 7000S Device Macrocell



Combinatorial logic is implemented in the logic array, which provides five product terms per macrocell. The product-term select matrix allocates these product terms for use as either primary logic inputs (to the OR and XOR gates) to implement combinatorial functions, or as secondary inputs to the macrocell's register clear, preset, clock, and clock enable control functions. Two kinds of expander product terms ("expanders") are available to supplement macrocell logic resources:

- Shareable expanders, which are inverted product terms that are fed back into the logic array
- Parallel expanders, which are product terms borrowed from adjacent macrocells

The Altera development system automatically optimizes product-term allocation according to the logic requirements of the design.

For registered functions, each macrocell flipflop can be individually programmed to implement D, T, JK, or SR operation with programmable clock control. The flipflop can be bypassed for combinatorial operation. During design entry, the designer specifies the desired flipflop type; the Altera development software then selects the most efficient flipflop operation for each registered function to optimize resource utilization.

Each programmable register can be clocked in three different modes:

- By a global clock signal. This mode achieves the fastest clock-to-output performance.
- By a global clock signal and enabled by an active-high clock enable. This mode provides an enable on each flipflop while still achieving the fast clock-to-output performance of the global clock.
- By an array clock implemented with a product term. In this mode, the flipflop can be clocked by signals from buried macrocells or I/O pins.

In EPM7032, EPM7064, and EPM7096 devices, the global clock signal is available from a dedicated clock pin, GCLK1, as shown in Figure 1. In MAX 7000E and MAX 7000S devices, two global clock signals are available. As shown in Figure 2, these global clock signals can be the true or the complement of either of the global clock pins, GCLK1 or GCLK2.

Each register also supports asynchronous preset and clear functions. As shown in Figures 3 and 4, the product-term select matrix allocates product terms to control these operations. Although the product-term-driven preset and clear of the register are active high, active-low control can be obtained by inverting the signal within the logic array. In addition, each register clear function can be individually driven by the active-low dedicated global clear pin (GCLRn). Upon power-up, each register in the device will be set to a low state.

All MAX 7000E and MAX 7000S I/O pins have a fast input path to a macrocell register. This dedicated path allows a signal to bypass the PIA and combinatorial logic and be driven to an input D flipflop with an extremely fast (2.5 ns) input setup time.

Expander Product Terms

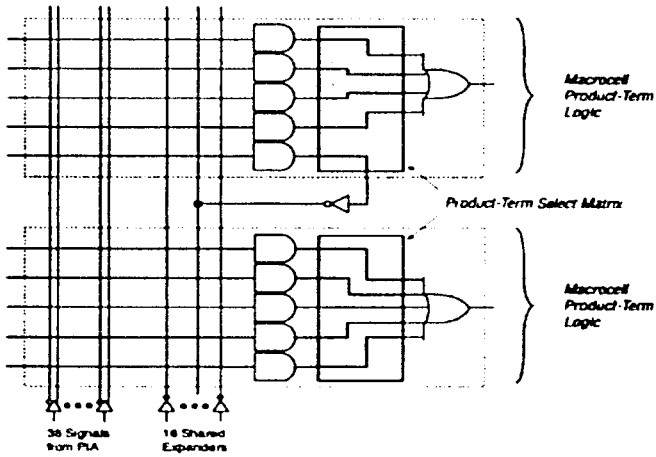
Although most logic functions can be implemented with the five product terms available in each macrocell, the more complex logic functions require additional product terms. Another macrocell can be used to supply the required logic resources; however, the MAX 7000 architecture also allows both shareable and parallel expander product terms ("expanders") that provide additional product terms directly to any macrocell in the same LAB. These expanders help ensure that logic is synthesized with the fewest possible logic resources to obtain the fastest possible speed.

Shareable Expanders

Each LAB has 16 shareable expanders that can be viewed as a pool of uncommitted single product terms (one from each macrocell) with inverted outputs that feed back into the logic array. Each shareable expander can be used and shared by any or all macrocells in the LAB to build complex logic functions. A small delay (t_{SEXP}) is incurred when shareable expanders are used. Figure 5 shows how shareable expanders can feed multiple macrocells.

Figure 5. Shareable Expanders

Shareable expanders can be shared by any or all macrocells in an LAB



Parallel Expanders

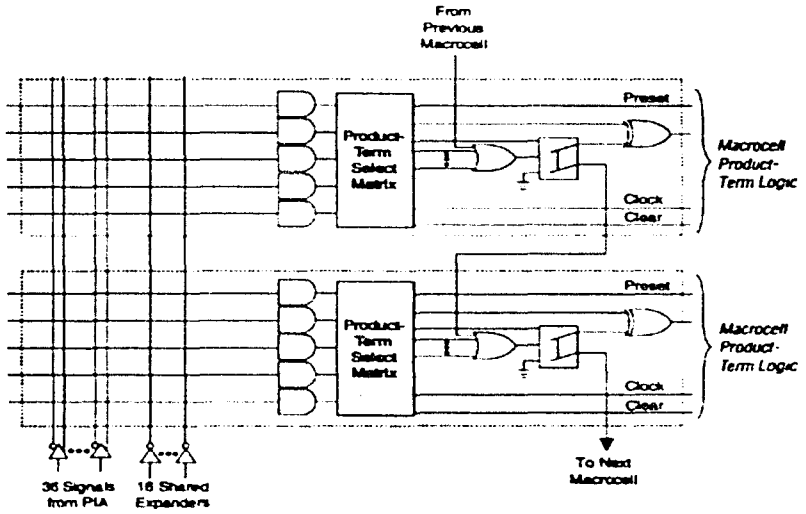
Parallel expanders are unused product terms that can be allocated to a neighboring macrocell to implement fast, complex logic functions. Parallel expanders allow up to 20 product terms to directly feed the macrocell OR logic, with five product terms provided by the macrocell and 15 parallel expanders provided by neighboring macrocells in the LAB.

The compiler can allocate up to three sets of up to five parallel expanders automatically to the macrocells that require additional product terms. Each set of five parallel expanders incurs a small, incremental timing delay (t_{PEXP}). For example, if a macrocell requires 14 product terms, the Compiler uses the five dedicated product terms within the macrocell and allocates two sets of parallel expanders; the first set includes five product terms and the second set includes four product terms, increasing the total delay by $2 \times t_{PEXP}$.

Two groups of 8 macrocells within each LAB (e.g., macrocells 1 through 8 and 9 through 16) form two chains to lend or borrow parallel expanders. A macrocell borrows parallel expanders from lower-numbered macrocells. For example, macrocell 8 can borrow parallel expanders from macrocell 7, from macrocells 7 and 6, or from macrocells 7, 6, and 5. Within each group of 8, the lowest-numbered macrocell can only lend parallel expanders and the highest-numbered macrocell can only borrow them. Figure 6 shows how parallel expanders can be borrowed from a neighboring macrocell.

Figure 6. Parallel Expanders

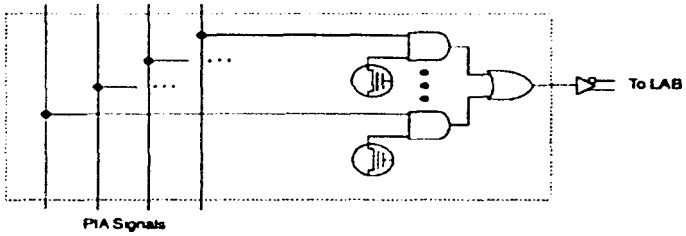
Unused product terms in a macrocell can be allocated to a neighboring macrocell.



Programmable Interconnect Array

Logic is routed between LABs via the programmable interconnect array (PIA). This global bus is a programmable path that connects any signal source to any destination on the device. All MAX 7000 dedicated inputs, I/O pins, and macrocell outputs feed the PIA, which makes the signals available throughout the entire device. Only the signals required by each LAB are actually routed from the PIA into the LAB. Figure 7 shows how the PIA signals are routed into the LAB. An EEPROM cell controls one input to a 2-input AND gate, which selects a PIA signal to drive into the LAB.

Figure 7. PIA Routing



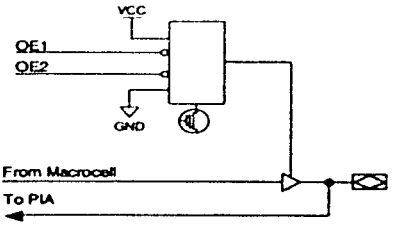
While the routing delays of channel-based routing schemes in masked or FPGAs are cumulative, variable, and path-dependent, the MAX 7000 PIA has a fixed delay. The PIA thus eliminates skew between signals and makes timing performance easy to predict.

I/O Control Blocks

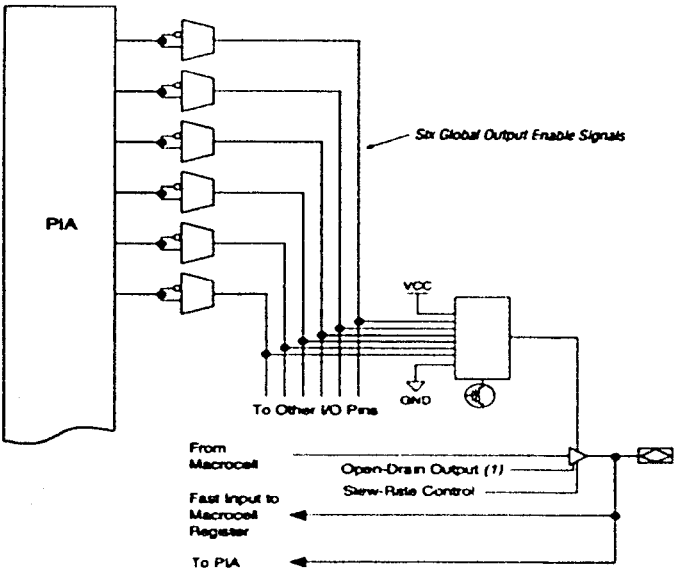
The I/O control block allows each I/O pin to be individually configured for input, output, or bidirectional operation. All I/O pins have a tri-state buffer that is individually controlled by one of the global output enable signals or directly connected to ground or V_{CC}. Figure 8 shows the I/O control block for the MAX 7000 family. The I/O control block of EPM7032, EPM7064, and EPM7096 devices has two global output enable signals that are driven by two dedicated active-low output enable pins (OE1 and OE2). The I/O control block of MAX 7000E and MAX 7000S devices has six global output enable signals that are driven by the true or complement of two output enable signals, a subset of the I/O pins, or a subset of the I/O macrocells.

Figure 8. I/O Control Block of MAX 7000 Devices

EPM7032, EPM7064 & EPM7096 Devices



MAX 7000E & MAX 7000S Devices



Note:
(1) The open-drain output option is available only in MAX 7000S devices.

When the tri-state buffer control is connected to ground, the output is tri-stated (high impedance) and the I/O pin can be used as a dedicated input. When the tri-state buffer control is connected to V_{CC}, the output is enabled.

The MAX 7000 architecture provides dual I/O feedback, in which macrocell and pin feedbacks are independent. When an I/O pin is configured as an input, the associated macrocell can be used for buried logic.

In-System Programmability (ISP)

MAX 7000S devices are in-system programmable via an industry-standard 4-pin Joint Test Action Group (JTAG) interface (IEEE Std. 1149.1-1990). ISP allows quick, efficient iterations during design development and debugging cycles. The MAX 7000S architecture internally generates the high programming voltage required to program EEPROM cells, allowing in-system programming with only a single 5.0 V power supply. During in-system programming, the I/O pins are tri-stated and pulled-up to eliminate board conflicts. The pull-up value is nominally 50 kΩ.

ISP simplifies the manufacturing flow by allowing devices to be mounted on a printed circuit board with standard in-circuit test equipment before they are programmed. MAX 7000S devices can be programmed by downloading the information via in-circuit testers (ICT), embedded processors, or the Altera MasterBlaster, ByteBlasterMV, ByteBlaster, BitBlaster download cables. (The ByteBlaster cable is obsolete and is replaced by the ByteBlasterMV cable, which can program and configure 2.5-V, 3.3-V, and 5.0-V devices.) Programming the devices after they are placed on the board eliminates lead damage on high-pin-count packages (e.g., QFP packages) due to device handling and allows devices to be reprogrammed after a system has already shipped to the field. For example, product upgrades can be performed in the field via software or modem.

In-system programming can be accomplished with either an adaptive or constant algorithm. An adaptive algorithm reads information from the unit and adapts subsequent programming steps to achieve the fastest possible programming time for that unit. Because some in-circuit testers cannot support an adaptive algorithm, Altera offers devices tested with a constant algorithm. Devices tested to the constant algorithm are marked with an "F" suffix in the ordering code.

The Jam™ Standard Test and Programming Language (STAPL) can be used to program MAX 7000S devices with in-circuit testers, PCs, or embedded processor.

For more information on using the Jam language, see *Application Note 88 (Using the Jam Language for ISP & ICR via an Embedded Processor)*.

The ISP circuitry in MAX 7000S devices is compatible with IEEE Std. 1532 specification. The IEEE Std. 1532 is a standard developed to allow concurrent ISP between multiple PLD vendors.

Programmable Speed/Power Control

MAX 7000 devices offer a power-saving mode that supports low-power operation across user-defined signal paths or the entire device. This feature allows total power dissipation to be reduced by 50% or more, because most logic applications require only a small fraction of all gates to operate at maximum frequency.

The designer can program each individual macrocell in a MAX 7000 device for either high-speed (i.e., with the Turbo Bit™ option turned on) or low-power (i.e., with the Turbo Bit option turned off) operation. As a result, speed-critical paths in the design can run at high speed, while the remaining paths can operate at reduced power. Macrocells that run at low power incur a nominal timing delay adder (t_{LPA}) for the t_{LAD}, t_{LAC}, t_{IC}, t_{EN}, and t_{SEXP}, t_{ACL}, and t_{CPPW} parameters.

Output Configuration

MAX 7000 device outputs can be programmed to meet a variety of system-level requirements.

MultiVolt I/O Interface

MAX 7000 devices—except 44-pin devices—support the MultiVolt I/O interface feature, which allows MAX 7000 devices to interface with systems that have differing supply voltages. The 5.0-V devices in all packages can be set for 3.3-V or 5.0-V I/O pin operation. These devices have one set of VCC pins for internal operation and input buffers (VCCINT), and another set for I/O output drivers (VCCIO).

The VCCINT pins must always be connected to a 5.0-V power supply. With a 5.0-V V_{CCINT} level, input voltage thresholds are at TTL levels, and are therefore compatible with both 3.3-V and 5.0-V inputs.

The VCCIO pins can be connected to either a 3.3-V or a 5.0-V power supply, depending on the output requirements. When the VCCIO pins are connected to a 5.0-V supply, the output levels are compatible with 5.0-V systems. When V_{CCIO} is connected to a 3.3-V supply, the output high is 3.3 V and is therefore compatible with 3.3-V or 5.0-V systems. Devices operating with V_{CCIO} levels lower than 4.75 V incur a nominally greater timing delay of t_{OD2} instead of t_{OD1}.

Open-Drain Output Option (MAX 7000S Devices Only)

MAX 7000S devices provide an optional open-drain (functionally equivalent to open-collector) output for each I/O pin. This open-drain output enables the device to provide system-level control signals (e.g., interrupt and write enable signals) that can be asserted by any of several devices. It can also provide an additional wired-OR plane.

By using an external 5.0-V pull-up resistor, output pins on MAX 7000S devices can be set to meet 5.0-V CMOS input voltages. When V_{CCIO} is 3.3 V, setting the open drain option will turn off the output pull-up transistor, allowing the external pull-up resistor to pull the output high enough to meet 5.0-V CMOS input voltages. When V_{CCIO} is 5.0 V, setting the output drain option is not necessary because the pull-up transistor will already turn off when the pin exceeds approximately 3.8 V, allowing the external pull-up resistor to pull the output high enough to meet 5.0-V CMOS input voltages.

Slew-Rate Control

The output buffer for each MAX 7000E and MAX 7000S I/O pin has an adjustable output slew rate that can be configured for low-noise or high-speed performance. A faster slew rate provides high-speed transitions for high-performance systems. However, these fast transitions may introduce noise transients into the system. A slow slew rate reduces system noise, but adds a nominal delay of 4 to 5 ns. In MAX 7000E devices, when the Turbo Bit is turned off, the slew rate is set for low noise performance. For MAX 7000S devices, each I/O pin has an individual EEPROM bit that controls the slew rate, allowing designers to specify the slew rate on a pin-by-pin basis.

Programming with External Hardware

MAX 7000 devices can be programmed on Windows-based PCs with the Altera Logic Programmer card, the Master Programming Unit (MPU), and the appropriate device adapter. The MPU performs a continuity check to ensure adequate electrical contact between the adapter and the device.

For more information, see the *Altera Programming Hardware Data Sheet*.

The Altera development system can use text- or waveform-format test vectors created with the Text Editor or Waveform Editor to test the programmed device. For added design verification, designers can perform functional testing to compare the functional behavior of a MAX 7000 device with the results of simulation. Moreover, Data I/O, BP Microsystems, and other programming hardware manufacturers also provide programming support for Altera devices.

For more information, see the *Programming Hardware Manufacturers*.

IEEE Std. 1149.1 (JTAG) Boundary-Scan Support

MAX 7000 devices support JTAG BST circuitry as specified by IEEE Std. 1149.1-1990. Table 6 describes the JTAG instructions supported by the MAX 7000 family. The pin-out tables (see the Altera web site (http://www.altera.com) or the *Altera Digital Library* for pin-out information) show the location of the JTAG control pins for each device. If the JTAG interface is not required, the JTAG pins are available as user I/O pins.

Table 6. MAX 7000 JTAG Instructions

JTAG Instruction	Devices	Description
SAMPLE/PRELOAD	EPM7128S EPM7160S EPM7192S EPM7256S	Allows a snapshot of signals at the device pins to be captured and examined during normal device operation, and permits an initial data pattern output at the device pins.
EXTEST	EPM7128S EPM7160S EPM7192S EPM7256S	Allows the external circuitry and board-level interconnections to be tested by forcing a test pattern at the output pins and capturing test results at the input pins.
BYPASS	EPM7032S EPM7064S EPM7128S EPM7160S EPM7192S EPM7256S	Places the 1-bit bypass register between the TDI and TDO pins, which allows the BST data to pass synchronously through a selected device to adjacent devices during normal device operation.
IDCODE	EPM7032S EPM7064S EPM7128S EPM7160S EPM7192S EPM7256S	Selects the IDCODE register and places it between TDI and TDO, allowing the IDCODE to be serially shifted out of TDO.
ISP Instructions	EPM7032S EPM7064S EPM7128S EPM7160S EPM7192S EPM7256S	These instructions are used when programming MAX 7000S devices via the JTAG ports with the MasterBlaster, ByteBlasterMV, BitBlaster download cable, or using a Jam File (.jam), Jam Byte-Code file (.jbc), or Serial Vector Format file (.svf) via an embedded processor or test equipment.

The instruction register length of MAX 7000S devices is 10 bits. Tables 7 and 8 show the boundary-scan register length and device IDCODE information for MAX 7000S devices.

Table 7. MAX 7000S Boundary-Scan Register Length

Device	Boundary-Scan Register Length
EPM7032S	1 *(1)*
EPM7064S	1 *(1)*
EPM7128S	288
EPM7160S	312
EPM7192S	360
EPM7256S	480

Note:
(1) This device does not support JTAG boundary-scan testing. Selecting either the EXTEST or SAMPLE/PRELOAD instruction will select the one-bit bypass register.

Table 8. 32-Bit MAX 7000 Device IDCODE *Note (1)*

Device	IDCODE (32 Bits)			
	Version (4 Bits)	Part Number (16 Bits)	Manufacturer's Identity (11 Bits)	1 (1 Bit) *(2)*
EPM7032S	0000	0111 0000 0011 0010	00001101110	1
EPM7064S	0000	0111 0000 0110 0100	00001101110	1
EPM7128S	0000	0111 0001 0010 1000	00001101110	1
EPM7160S	0000	0111 0001 0110 0000	00001101110	1
EPM7192S	0000	0111 0001 1001 0010	00001101110	1
EPM7256S	0000	0111 0010 0101 0110	00001101110	1

Notes:
(1) The most significant bit (MSB) is on the left.
(2) The least significant bit (LSB) for all JTAG IDCODEs is 1.

Figure 9 shows the timing requirements for the JTAG signals.

Figure 9. MAX 7000 JTAG Waveforms

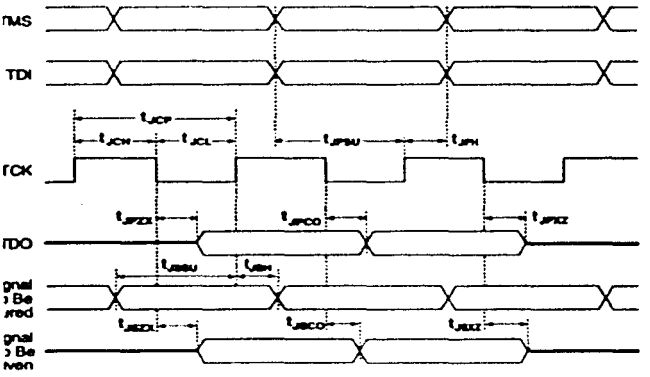


Table 9 shows the JTAG timing parameters and values for MAX 7000S devices.

Table 9. JTAG Timing Parameters & Values for MAX 7000S Devices

Symbol	Parameter	Min	Max	Unit
t_{JCP}	TCK clock period	100		ns
t_{JCH}	TCK clock high time	50		ns
t_{JCL}	TCK clock low time	50		ns
t_{JPSU}	JTAG port setup time	20		ns
t_{JPH}	JTAG port hold time	45		ns
t_{JPCO}	JTAG port clock to output		25	ns
t_{JPZX}	JTAG port high impedance to valid output		25	ns
t_{JPXZ}	JTAG port valid output to high impedance		25	ns
t_{JSSU}	Capture register setup time	20		ns
t_{JSH}	Capture register hold time	45		ns
t_{JSCO}	Update register clock to output		25	ns
t_{JSZX}	Update register high impedance to valid output		25	ns
t_{JSXZ}	Update register valid output to high impedance		25	ns

For more information, see *Application Note 39 (IEEE 1149.1 (JTAG) Boundary-Scan Testing in Altera Devices)*.

Design Security

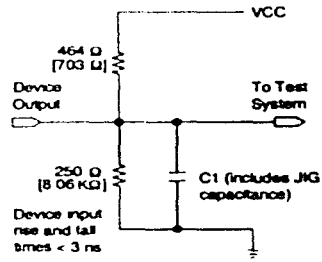
All MAX 7000 devices contain a programmable security bit that controls access to the data programmed into the device. When this bit is programmed, a proprietary design implemented in the device cannot be copied or retrieved. This feature provides a high level of design security because programmed data within EEPROM cells is invisible. The security bit that controls this function, as well as all other programmed data, is reset only when the device is reprogrammed.

Generic Testing

Each MAX 7000 device is functionally tested. Complete testing of each programmable EEPROM bit and all internal logic elements ensures 100% programming yield. AC test measurements are taken under conditions equivalent to those shown in Figure 10. Test patterns can be used and then erased during early stages of the production flow.

Figure 10. MAX 7000 AC Test Conditions

Power supply transients can affect AC measurements. Simultaneous transitions of multiple outputs should be avoided for accurate measurement. Threshold tests must not be performed under AC conditions. Large-amplitude, fast ground-current transients normally occur as the device outputs discharge the load capacitances. When these transients flow through the parasitic inductance between the device ground pin and the test system ground, significant reductions in observable noise immunity can result. Numbers in brackets are for 2.5-V devices and outputs. Numbers without brackets are for 3.3-V devices and outputs.



QFP Carrier & Development Socket

MAX 7000 and MAX 7000E devices in QFP packages with 100 or more pins are shipped in special plastic carriers to protect the QFP leads. The carrier is used with a prototype development socket and special programming hardware available from Altera. This carrier technology makes it possible to program, test, erase, and reprogram a device without exposing the leads to mechanical stress.

For detailed information and carrier dimensions, refer to the *QFP Carrier & Development Socket Data Sheet*.

MAX 7000S devices are not shipped in carriers.

Operating Conditions

Tables 10 through 15 provide information about absolute maximum ratings, recommended operating conditions, operating conditions, and capacitance for 5.0-V MAX 7000 devices.

Table 10. MAX 7000 5.0-V Device Absolute Maximum Ratings *Note (1)*

Symbol	Parameter	Conditions	Min	Max	Unit
V_{CC}	Supply voltage	With respect to ground *(2)*	–2.0	7.0	V
V_I	DC input voltage		–2.0	7.0	V
I_{OUT}	DC output current, per pin		–25	25	mA
T_{STG}	Storage temperature	No bias	–65	150	°C
T_{AMB}	Ambient temperature	Under bias	–65	135	°C
T_J	Junction temperature	Ceramic packages, under bias		150	°C
		PQFP and RQFP packages, under bias		135	°C

Table 11. MAX 7000 5.0-V Device Recommended Operating Conditions

Symbol	Parameter	Conditions	Min	Max	Unit
V_{CCINT}	Supply voltage for internal logic and input buffers	*(3)*, *(4)*	4.75 (4.50)	5.25 (5.50)	V
V_{CCIO}	Supply voltage for output drivers, 5.0-V operation	*(3)*, *(4)*	4.75 (4.50)	5.25 (5.50)	V
	Supply voltage for output drivers, 3.3-V operation	*(3)*, *(4)*, *(5)*	3.00 (3.00)	3.60 (3.60)	V
V_{CCISP}	Supply voltage during ISP	*(6)*	4.75	5.25	V
V_I	Input voltage		–0.5 *(7)*	V_{CCINT} + 0.5	V
V_O	Output voltage		0	V_{CCIO}	V
T_A	Ambient temperature	For commercial use	0	70	°C
		For industrial use	–40	85	°C
T_J	Junction temperature	For commercial use	0	90	°C
		For industrial use	–40	105	°C
t_R	Input rise time			40	ns
t_F	Input fall time			40	ns

Table 12. MAX 7000 5.0-V Device DC Operating Conditions *Note (8)*

Symbol	Parameter	Conditions	Min	Max	Unit
V_{IH}	High-level input voltage		2.0	V_{CCINT} + 0.5	V
V_{IL}	Low-level input voltage		–0.5 *(7)*	0.8	V
V_{OH}	5.0-V high-level TTL output voltage	I_{OH} = –4 mA DC, V_{CCIO} = 4.75 V *(9)*	2.4		V
	3.3-V high-level TTL output voltage	I_{OH} = –4 mA DC, V_{CCIO} = 3.00 V *(9)*	2.4		V
	3.3-V high-level CMOS output voltage	I_{OH} = –0.1 mA DC, V_{CCIO} = 3.0 V *(9)*	V_{CCIO} – 0.2		V
V_{OL}	5.0-V low-level TTL output voltage	I_{OL} = 12 mA DC, V_{CCIO} = 4.75 V *(10)*		0.45	V
	3.3-V low-level TTL output voltage	I_{OL} = 12 mA DC, V_{CCIO} = 3.00 V *(10)*		0.45	V
	3.3-V low-level CMOS output voltage	I_{OL} = 0.1 mA DC, V_{CCIO} = 3.0 V *(10)*		0.2	V
I_I	Leakage current of dedicated input pins	V_I = –0.5 to 5.5 V *(10)*	–10	10	µA
I_{OZ}	I/O pin tri-state output off-state current	V_I = –0.5 to 5.5 V *(10)*, *(11)*	–40	40	µA

Table 13. MAX 7000 5.0-V Device Capacitance: EPM7032, EPM7064 & EPM7096 Devices *Note (12)*

Symbol	Parameter	Conditions	Min	Max	Unit
C_{IN}	Input pin capacitance	V_{IN} = 0 V, f = 1.0 MHz		12	pF
$C_{I/O}$	I/O pin capacitance	V_{OUT} = 0 V, f = 1.0 MHz		12	pF

Table 14. MAX 7000 5.0-V Device Capacitance: MAX 7000E Devices *Note (12)*

Symbol	Parameter	Conditions	Min	Max	Unit
C_{IN}	Input pin capacitance	V_{IN} = 0 V, f = 1.0 MHz		15	pF
$C_{I/O}$	I/O pin capacitance	V_{OUT} = 0 V, f = 1.0 MHz		15	pF

Table 15. MAX 7000 5.0-V Device Capacitance: MAX 7000S Devices *Note (12)*

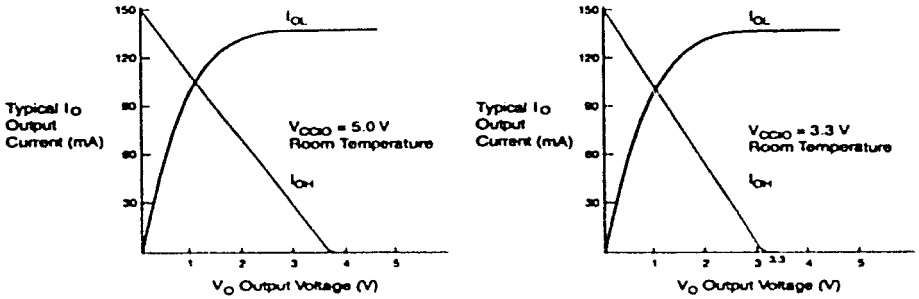
Symbol	Parameter	Conditions	Min	Max	Unit
C_{IN}	Dedicated input pin capacitance	V_{IN} = 0 V, f = 1.0 MHz		10	pF
$C_{I/O}$	I/O pin capacitance	V_{OUT} = 0 V, f = 1.0 MHz		10	pF

Notes to tables:

(1) See the *Operating Requirements for Altera Devices Data Sheet*.
(2) Minimum DC input voltage on I/O pins is –0.5 V and on 4 dedicated input pins is –0.3 V. During transitions, the inputs may undershoot to –2.0 V or overshoot to 7.0 V for input currents less than 100 mA and periods shorter than 20 ns.
(3) Numbers in parentheses are for industrial-temperature-range devices.
(4) V_{CC} must rise monotonically.
(5) 3.3-V I/O operation is not available for 44-pin packages.
(6) The V_{CCISP} parameter applies only to MAX 7000S devices.
(7) During in-system programming, the minimum DC input voltage is –0.3 V.
(8) These values are specified under the MAX 7000 recommended operating conditions in Table 11 on page 23.
(9) The parameter is measured with 50% of the outputs each sourcing the specified current. The I_{OH} parameter refers to high-level TTL or CMOS output current.
(10) The parameter is measured with 50% of the outputs each sinking the specified current. The I_{OL} parameter refers to low-level TTL, PCI, or CMOS output current.
(11) When the JTAG interface is enabled in MAX 7000S devices, the input leakage current on the JTAG pins is typically –60 µA.
(12) Capacitance is measured at 25° C and is sample-tested only. The OE1 pin has a maximum capacitance of 20 pF.

Figure 11 shows the typical output drive characteristics of MAX 7000 devices.

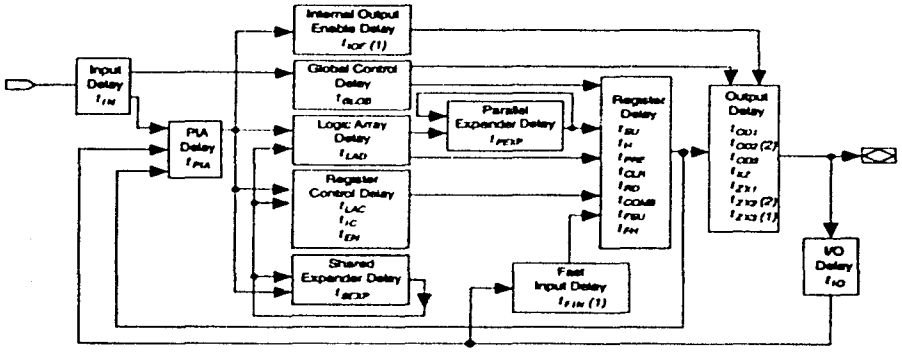
Figure 11. Output Drive Characteristics of 5.0-V MAX 7000 Devices



Timing Model

MAX 7000 device timing can be analyzed with the Altera software, with a variety of popular industry-standard EDA simulators and timing analyzers, or with the timing model shown in Figure 12. MAX 7000 devices have fixed internal delays that enable the designer to determine the worst-case timing of any design. The Altera software provides timing simulation, point-to-point delay prediction, and detailed timing analysis for a device-wide performance evaluation.

Figure 12. MAX 7000 Timing Model



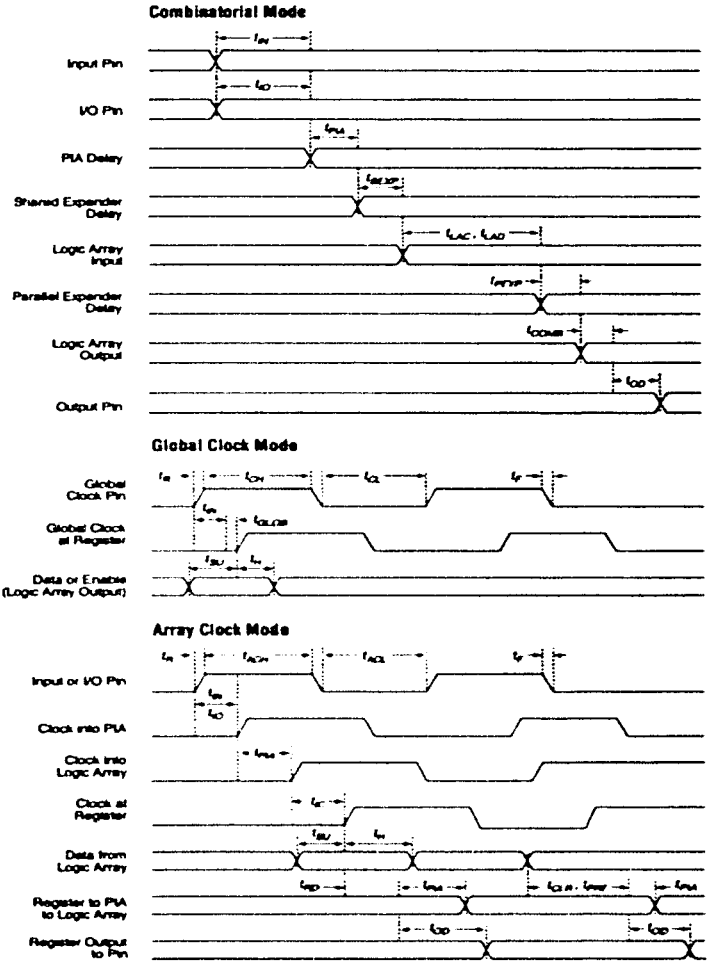
Notes:
(1) Only available in MAX 7000E and MAX 7000S devices.
(2) Not available in 44-pin devices.

The timing characteristics of any signal path can be derived from the timing model and parameters of a particular device. External timing parameters, which represent pin-to-pin timing delays, can be calculated as the sum of internal parameters. Figure 13 shows the internal timing relationship of internal and external delay parameters.

For more information, see *Application Note 94 (Understanding MAX 7000 Timing)*.

Figure 13. Switching Waveforms

t_R & t_F < 3 ns. Inputs are driven at 3 V for a logic high and 0 V for a logic low. All timing characteristics are measured at 1.5 V.



Tables 16 through 23 show the MAX 7000 and MAX 7000E AC operating conditions.

Table 16. MAX 7000 & MAX 7000E External Timing Parameters *Note (1)*

Symbol	Parameter	Conditions	-6 Speed Grade		-7 Speed Grade		Unit
			Min	Max	Min	Max	
t_{PD1}	Input to non-registered output	C1 = 35 pF		6.0		7.5	ns
t_{PD2}	I/O input to non-registered output	C1 = 35 pF		6.0		7.5	ns
t_{SU}	Global clock setup time		5.0		6.0		ns
t_{H}	Global clock hold time		0.0		0.0		ns
t_{FSU}	Global clock setup time of fast input	*(2)*	2.5		3.0		ns
t_{FH}	Global clock hold time of fast input	*(2)*	0.5		0.5		ns
t_{CO1}	Global clock to output delay	C1 = 35 pF		4.0		4.5	ns
t_{CH}	Global clock high time		2.5		3.0		ns
t_{CL}	Global clock low time		2.5		3.0		ns
t_{ASU}	Array clock setup time		2.5		3.0		ns
t_{AH}	Array clock hold time		2.0		2.0		ns
t_{ACO1}	Array clock to output delay	C1 = 35 pF		6.5		7.5	ns
t_{ACH}	Array clock high time		3.0		3.0		ns
t_{ACL}	Array clock low time		3.0		3.0		ns
t_{CPPW}	Minimum pulse width for clear and preset	*(3)*	3.0		3.0		ns
t_{ODH}	Output data hold time after clock	C1 = 35 pF *(4)*	1.0		1.0		ns
t_{CNT}	Minimum global clock period			6.6		8.0	ns
f_{CNT}	Maximum internal global clock frequency	*(5)*	151.5		125.0		MHz
t_{ACNT}	Minimum array clock period			6.6		8.0	ns
f_{ACNT}	Maximum internal array clock frequency	*(5)*	151.5		125.0		MHz
f_{MAX}	Maximum clock frequency	*(6)*	200		166.7		MHz

Table 17. MAX 7000 & MAX 7000E Internal Timing Parameters *Note (1)*

Symbol	Parameter	Conditions	Speed Grade -6		Speed Grade -7		Unit
			Min	Max	Min	Max	
t_{IN}	Input pad and buffer delay			0.4		0.5	ns
t_{IO}	I/O input pad and buffer delay			0.4		0.5	ns
t_{FIN}	Fast input delay	*(2)*		0.8		1.0	ns
t_{SEXP}	Shared expander delay			3.5		4.0	ns
t_{PEXP}	Parallel expander delay			0.8		0.8	ns
t_{LAD}	Logic array delay			2.0		3.0	ns
t_{LAC}	Logic control array delay			2.0		3.0	ns
t_{IOE}	Internal output enable delay	*(2)*				2.0	ns
t_{OD1}	Output buffer and pad delay Slow slew rate = off, V_{CCIO} = 5.0 V	C1 = 35 pF		2.0		2.0	ns
t_{OD2}	Output buffer and pad delay Slow slew rate = off, V_{CCIO} = 3.3 V	C1 = 35 pF *(7)*		2.5		2.5	ns
t_{OD3}	Output buffer and pad delay Slow slew rate = on, V_{CCIO} = 5.0 V or 3.3 V	C1 = 35 pF *(2)*		7.0		7.0	ns
t_{ZX1}	Output buffer enable delay Slow slew rate = off, V_{CCIO} = 5.0 V	C1 = 35 pF		4.0		4.0	ns
t_{ZX2}	Output buffer enable delay Slow slew rate = off, V_{CCIO} = 3.3 V	C1 = 35 pF *(7)*		4.5		4.5	ns
t_{ZX3}	Output buffer enable delay Slow slew rate = on V_{CCIO} = 5.0 V or 3.3 V	C1 = 35 pF *(2)*		9.0		9.0	ns
t_{XZ}	Output buffer disable delay	C1 = 5 pF		4.0		4.0	ns
t_{SU}	Register setup time		3.0		3.0		ns
t_{H}	Register hold time		1.5		2.0		ns
t_{FSU}	Register setup time of fast input	*(2)*	2.5		3.0		ns
t_{FH}	Register hold time of fast input	*(2)*	0.5		0.5		ns
t_{RD}	Register delay			0.8		1.0	ns
t_{COMB}	Combinational delay			0.8		1.0	ns
t_{IC}	Array clock delay			2.5		3.0	ns
t_{EN}	Register enable time			2.0		3.0	ns
t_{GLOB}	Global control delay			0.8		1.0	ns
t_{PRE}	Register preset time			2.0		2.0	ns
t_{CLR}	Register clear time			2.0		2.0	ns
t_{PIA}	PIA delay			0.8		1.0	ns
t_{LPA}	Low-power adder	*(8)*		10.0		10.0	ns

Table 18. MAX 7000 & MAX 7000E External Timing Parameters *Note (1)*

Symbol	Parameter	Conditions	Speed Grade				Unit
			MAX 7000E (-10P)		MAX 7000 (-10) MAX 7000E (-10)		
			Min	Max	Min	Max	
t_{PD1}	Input to non-registered output	C1 = 35 pF		10.0		10.0	ns
t_{PD2}	I/O input to non-registered output	C1 = 35 pF		10.0		10.0	ns
t_{SU}	Global clock setup time		7.0		8.0		ns
t_{H}	Global clock hold time		0.0		0.0		ns
t_{FSU}	Global clock setup time of fast input	*(2)*	3.0		3.0		ns
t_{FH}	Global clock hold time of fast input	*(2)*	0.5		0.5		ns
t_{CO1}	Global clock to output delay	C1 = 35 pF		5.0		5	ns
t_{CH}	Global clock high time		4.0		4.0		ns
t_{CL}	Global clock low time		4.0		4.0		ns
t_{ASU}	Array clock setup time		2.0		3.0		ns
t_{AH}	Array clock hold time		3.0		3.0		ns
t_{ACO1}	Array clock to output delay	C1 = 35 pF		10.0		10.0	ns
t_{ACH}	Array clock high time		4.0		4.0		ns
t_{ACL}	Array clock low time		4.0		4.0		ns
t_{CPPW}	Minimum pulse width for clear and preset	*(3)*	4.0		4.0		ns
t_{ODH}	Output data hold time after clock	C1 = 35 pF *(4)*	1.0		1.0		ns
t_{CNT}	Minimum global clock period			10.0		10.0	ns
f_{CNT}	Maximum internal global clock frequency	*(5)*	100.0		100.0		MHz
t_{ACNT}	Minimum array clock period			10.0		10.0	ns
f_{ACNT}	Maximum internal array clock frequency	*(5)*	100.0		100.0		MHz
f_{MAX}	Maximum clock frequency	*(6)*	125.0		125.0		MHz

Table 19. MAX 7000 & MAX 7000E Internal Timing Parameters *Note (1)*

Symbol	Parameter	Conditions	Speed Grade				Unit
			MAX 7000E (-10P)		MAX 7000 (-10) MAX 7000E (-10)		
			Min	Max	Min	Max	
t_{IN}	Input pad and buffer delay			0.5		1.0	ns
t_{IO}	I/O input pad and buffer delay			0.5		1.0	ns
t_{FIN}	Fast input delay	*(2)*		1.0		1.0	ns
t_{SEXP}	Shared expander delay			5.0		5.0	ns
t_{PEXP}	Parallel expander delay			0.8		0.8	ns
t_{LAD}	Logic array delay			5.0		5.0	ns
t_{LAC}	Logic control array delay			5.0		5.0	ns
t_{IOE}	Internal output enable delay	*(2)*		2.0		2.0	ns
t_{OD1}	Output buffer and pad delay Slow slew rate = off V_{CCIO} = 5.0 V	C1 = 35 pF		1.5		2.0	ns
t_{OD2}	Output buffer and pad delay Slow slew rate = off V_{CCIO} = 3.3 V	C1 = 35 pF *(7)*		2.0		2.5	ns
t_{OD3}	Output buffer and pad delay Slow slew rate = on V_{CCIO} = 5.0 V or 3.3 V	C1 = 35 pF *(2)*		5.5		6.0	ns
t_{ZX1}	Output buffer enable delay Slow slew rate = off V_{CCIO} = 5.0 V	C1 = 35 pF		5.0		5.0	ns
t_{ZX2}	Output buffer enable delay Slow slew rate = off V_{CCIO} = 3.3 V	C1 = 35 pF *(7)*		5.5		5.5	ns
t_{ZX3}	Output buffer enable delay Slow slew rate = on V_{CCIO} = 5.0 V or 3.3 V	C1 = 35 pF *(2)*		9.0		9.0	ns
t_{XZ}	Output buffer disable delay	C1 = 5 pF		5.0		5.0	ns
t_{SU}	Register setup time		2.0		3.0		ns
t_{H}	Register hold time		3.0		3.0		ns
t_{FSU}	Register setup time of fast input	*(2)*	3.0		3.0		ns
t_{FH}	Register hold time of fast input	*(2)*	0.5		0.5		ns
t_{RD}	Register delay			2.0		1.0	ns
t_{COMB}	Combinatorial delay			2.0		1.0	ns
t_{IC}	Array clock delay			5.0		5.0	ns
t_{EN}	Register enable time			5.0		5.0	ns
t_{GLOB}	Global control delay			1.0		1.0	ns
t_{PRE}	Register preset time			3.0		3.0	ns
t_{CLR}	Register clear time			3.0		3.0	ns
t_{PIA}	PIA delay			1.0		1.0	ns
t_{LPA}	Low-power adder	*(8)*		11.0		11.0	ns

Table 20. MAX 7000 & MAX 7000E External Timing Parameters *Note (1)*

Symbol	Parameter	Conditions	Speed Grade				Unit
			MAX 7000E (-12P)		MAX 7000 (-12) MAX 7000E (-12)		
			Min	Max	Min	Max	
t_{PD1}	Input to non-registered output	C1 = 35 pF		12.0		12.0	ns
t_{PD2}	I/O input to non-registered output	C1 = 35 pF		12.0		12.0	ns
t_{SU}	Global clock setup time		7.0		10.0		ns
t_{H}	Global clock hold time		0.0		0.0		ns
t_{FSU}	Global clock setup time of fast input	*(2)*	3.0		3.0		ns
t_{FH}	Global clock hold time of fast input	*(2)*	0.0		0.0		ns
t_{CO1}	Global clock to output delay	C1 = 35 pF		6.0		6.0	ns
t_{CH}	Global clock high time		4.0		4.0		ns
t_{CL}	Global clock low time		4.0		4.0		ns
t_{ASU}	Array clock setup time		3.0		4.0		ns
t_{AH}	Array clock hold time		4.0		4.0		ns
t_{ACO1}	Array clock to output delay	C1 = 35 pF		12.0		12.0	ns
t_{ACH}	Array clock high time		5.0		5.0		ns
t_{ACL}	Array clock low time		5.0		5.0		ns
t_{CPPW}	Minimum pulse width for clear and preset	*(3)*	5.0		5.0		ns
t_{ODH}	Output data hold time after clock	C1 = 35 pF *(4)*	1.0		1.0		ns
t_{CNT}	Minimum global clock period			11.0		11.0	ns
f_{CNT}	Maximum internal global clock frequency	*(5)*	90.9		90.9		MHz
t_{ACNT}	Minimum array clock period			11.0		11.0	ns
f_{ACNT}	Maximum internal array clock frequency	*(5)*	90.9		90.9		MHz
f_{MAX}	Maximum clock frequency	*(6)*	125.0		125.0		MHz

Table 21. MAX 7000 & MAX 7000E Internal Timing Parameters *Note (1)*

Symbol	Parameter	Conditions	Speed Grade				Unit
			MAX 7000E (-12P)		MAX 7000 (-12) MAX 7000E (-12)		
			Min	Max	Min	Max	
t_{IN}	Input pad and buffer delay			1.0		2.0	ns
t_{IO}	I/O input pad and buffer delay			1.0		2.0	ns
t_{FIN}	Fast input delay	*(2)*		1.0		1.0	ns
t_{SEXP}	Shared expander delay			7.0		7.0	ns
t_{PEXP}	Parallel expander delay			1.0		1.0	ns
t_{LAD}	Logic array delay			7.0		5.0	ns
t_{LAC}	Logic control array delay			5.0		5.0	ns
t_{IOE}	Internal output enable delay	*(2)*		2.0		2.0	ns
t_{OD1}	Output buffer and pad delay Slow slew rate = off V_{CCIO} = 5.0 V	C1 = 35 pF		1.0		3.0	ns
t_{OD2}	Output buffer and pad delay Slow slew rate = off V_{CCIO} = 3.3 V	C1 = 35 pF *(7)*		2.0		4.0	ns
t_{OD3}	Output buffer and pad delay Slow slew rate = on V_{CCIO} = 5.0 V or 3.3 V	C1 = 35 pF *(2)*		5.0		7.0	ns
t_{ZX1}	Output buffer enable delay Slow slew rate = off V_{CCIO} = 5.0 V	C1 = 35 pF		6.0		6.0	ns
t_{ZX2}	Output buffer enable delay Slow slew rate = off V_{CCIO} = 3.3 V	C1 = 35 pF *(7)*		7.0		7.0	ns
t_{ZX3}	Output buffer enable delay Slow slew rate = on V_{CCIO} = 5.0 V or 3.3 V	C1 = 35 pF *(2)*		10.0		10.0	ns
t_{XZ}	Output buffer disable delay	C1 = 5 pF		6.0		6.0	ns
t_{SU}	Register setup time		1.0		4.0		ns
t_{H}	Register hold time		6.0		4.0		ns
t_{FSU}	Register setup time of fast input	*(2)*	4.0		2.0		ns
t_{FH}	Register hold time of fast input	*(2)*	0.0		2.0		ns
t_{RD}	Register delay			2.0		1.0	ns
t_{COMB}	Combinatorial delay			2.0		1.0	ns
t_{IC}	Array clock delay			5.0		5.0	ns
t_{EN}	Register enable time			7.0		5.0	ns
t_{GLOB}	Global control delay			2.0		0.0	ns
t_{PRE}	Register preset time			4.0		3.0	ns
t_{CLR}	Register clear time			4.0		3.0	ns
t_{PIA}	PIA delay			1.0		1.0	ns
t_{LPA}	Low-power adder	*(8)*		12.0		12.0	ns

Table 22. MAX 7000 & MAX 7000E External Timing Parameters *Note (1)*

Symbol	Parameter	Conditions	Speed Grade						Unit
			-15		-15T		-20		
			Min	Max	Min	Max	Min	Max	
t_{PD1}	Input to non-registered output	C1 = 35 pF		15.0		15.0		20.0	ns
t_{PD2}	I/O input to non-registered output	C1 = 35 pF		15.0		15.0		20.0	ns
t_{SU}	Global clock setup time		11.0		11.0		12.0		ns
t_{H}	Global clock hold time		0.0		0.0		0.0		ns
t_{FSU}	Global clock setup time of fast input	*(2)*	3.0		–		5.0		ns
t_{FH}	Global clock hold time of fast input	*(2)*	0.0		–		0.0		ns
t_{CO1}	Global clock to output delay	C1 = 35 pF		8.0		8.0		12.0	ns
t_{CH}	Global clock high time		5.0		6.0		6.0		ns
t_{CL}	Global clock low time		5.0		6.0		6.0		ns
t_{ASU}	Array clock setup time		4.0		4.0		5.0		ns
t_{AH}	Array clock hold time		4.0		4.0		5.0		ns
t_{ACO1}	Array clock to output delay	C1 = 35 pF		15.0		15.0		20.0	ns
t_{ACH}	Array clock high time		6.0		6.5		8.0		ns
t_{ACL}	Array clock low time		6.0		6.5		8.0		ns
t_{CPPW}	Minimum pulse width for clear and preset	*(3)*	6.0		6.5		8.0		ns
t_{ODH}	Output data hold time after clock	C1 = 35 pF *(4)*	1.0		1.0		1.0		ns
t_{CNT}	Minimum global clock period			13.0		13.0		16.0	ns
f_{CNT}	Maximum internal global clock frequency	*(5)*	76.9		76.9		62.5		MHz
t_{ACNT}	Minimum array clock period			13.0		13.0		16.0	ns
f_{ACNT}	Maximum internal array clock frequency	*(5)*	76.9		76.9		62.5		MHz
f_{MAX}	Maximum clock frequency	*(6)*	100		83.3		83.3		MHz

Table 23. MAX 7000 & MAX 7000E Internal Timing Parameters *Note (1)*

Symbol	Parameter	Conditions	Speed Grade						Unit
			-15		-15T		-20		
			Min	Max	Min	Max	Min	Max	
t_{IN}	Input pad and buffer delay			2.0		2.0		3.0	ns
t_{IO}	I/O input pad and buffer delay			2.0		2.0		3.0	ns
t_{FIN}	Fast input delay	*(2)*		2.0		–		4.0	ns
t_{SEXP}	Shared expander delay			8.0		10.0		9.0	ns
t_{PEXP}	Parallel expander delay			1.0		1.0		2.0	ns
t_{LAD}	Logic array delay			6.0		6.0		8.0	ns
t_{LAC}	Logic control array delay			6.0		6.0		8.0	ns
t_{IOE}	Internal output enable delay	*(2)*		3.0		–		4.0	ns
t_{OD1}	Output buffer and pad delay Slow slew rate = off V_{CCIO} = 5.0 V	C1 = 35 pF		4.0		4.0		5.0	ns
t_{OD2}	Output buffer and pad delay Slow slew rate = off V_{CCIO} = 3.3 V	C1 = 35 pF *(7)*		5.0		–		6.0	ns
t_{OD3}	Output buffer and pad delay Slow slew rate = on V_{CCIO} = 5.0 V or 3.3 V	C1 = 35 pF *(2)*		8.0		–		9.0	ns
t_{ZX1}	Output buffer enable delay Slow slew rate = off V_{CCIO} = 5.0 V	C1 = 35 pF		6.0		6.0		10.0	ns
t_{ZX2}	Output buffer enable delay Slow slew rate = off V_{CCIO} = 3.3 V	C1 = 35 pF *(7)*		7.0		–		11.0	ns
t_{ZX3}	Output buffer enable delay Slow slew rate = on V_{CCIO} = 5.0 V or 3.3 V	C1 = 35 pF *(2)*		10.0		–		14.0	ns
t_{XZ}	Output buffer disable delay	C1 = 5 pF		6.0		6.0		10.0	ns
t_{SU}	Register setup time		4.0		4.0		4.0		ns
t_{H}	Register hold time		4.0		4.0		5.0		ns
t_{FSU}	Register setup time of fast input	*(2)*	2.0		–		4.0		ns
t_{FH}	Register hold time of fast input	*(2)*	2.0		–		3.0		ns
t_{RD}	Register delay			1.0		1.0		1.0	ns
t_{COMB}	Combinational delay			1.0		1.0		1.0	ns
t_{IC}	Array clock delay			6.0		6.0		8.0	ns
t_{EN}	Register enable time			6.0		6.0		8.0	ns
t_{GLOB}	Global control delay			1.0		1.0		3.0	ns
t_{PRE}	Register preset time			4.0		4.0		4.0	ns
t_{CLR}	Register clear time			4.0		4.0		4.0	ns
t_{PIA}	PIA delay			2.0		2.0		3.0	ns
t_{LPA}	Low-power adder	*(8)*		13.0		15.0		15.0	ns

Notes to tables:

(1) These values are specified under the recommended operating conditions shown in Table 11. See Figure 13 for more information on switching waveforms.
(2) This parameter applies to MAX 7000E devices only.
(3) This minimum pulse width for preset and clear applies for both global clear and array controls. The t_{LPA} parameter must be added to this minimum width if the clear or reset signal incorporates the t_{LAD} parameter into the signal path.
(4) This parameter is a guideline that is sample-tested only and is based on extensive device characterization. This parameter applies for both global and array clocking.
(5) These parameters are measured with a 16-bit loadable, enabled, up/down counter programmed into each LAB.
(6) The f_{MAX} values represent the highest frequency for pipelined data.
(7) Operating conditions: V_{CCIO} = 3.3 V ± 10% for commercial and industrial use.
(8) The t_{LPA} parameter must be added to the t_{LAD}, t_{LAC}, t_{IC}, t_{EN}, t_{SEXP}, t_{ACL} and t_{CPPW} parameters for macrocells running in the low-power mode.

Tables 24 and 25 show the EPM7032S AC operating conditions.

Table 24. EPM7032S External Timing Parameters (Part 1 of 2) *Note (1)*

Symbol	Parameter	Conditions	Speed Grade								Unit
			-5		-6		-7		-10		
			Min	Max	Min	Max	Min	Max	Min	Max	
t_{PD1}	Input to non-registered output	C1 = 35 pF		5.0		6.0		7.5		10.0	ns
t_{PD2}	I/O input to non-registered output	C1 = 35 pF		5.0		6.0		7.5		10.0	ns
t_{SU}	Global clock setup time		2.9		4.0		5.0		7.0		ns
t_{H}	Global clock hold time		0.0		0.0		0.0		0.0		ns
t_{FSU}	Global clock setup time of fast input		2.5		2.5		2.5		3.0		ns
t_{FH}	Global clock hold time of fast input		0.0		0.0		0.0		0.5		ns
t_{CO1}	Global clock to output delay	C1 = 35 pF		3.2		3.5		4.3		5.0	ns
t_{CH}	Global clock high time		2.0		2.5		3.0		4.0		ns
t_{CL}	Global clock low time		2.0		2.5		3.0		4.0		ns
t_{ASU}	Array clock setup time		0.7		0.9		1.1		2.0		ns
t_{AH}	Array clock hold time		1.8		2.1		2.7		3.0		ns
t_{ACO1}	Array clock to output delay	C1 = 35 pF		5.4		6.6		8.2		10.0	ns
t_{ACH}	Array clock high time		2.5		2.5		3.0		4.0		ns
t_{ACL}	Array clock low time		2.5		2.5		3.0		4.0		ns
t_{CPPW}	Minimum pulse width for clear and preset	*(2)*	2.5		2.5		3.0		4.0		ns
t_{ODH}	Output data hold time after clock	C1 = 35 pF *(3)*	1.0		1.0		1.0		1.0		ns
t_{CNT}	Minimum global clock period			5.7		7.0		8.6		10.0	ns
f_{CNT}	Maximum internal global clock frequency	*(4)*	175.4		142.9		116.3		100.0		MHz
t_{ACNT}	Minimum array clock period			5.7		7.0		8.6		10.0	ns

Table 24. EPM7032S External Timing Parameters (Part 2 of 2) *Note (1)*

Symbol	Parameter	Conditions	Speed Grade								Unit
			-5		-6		-7		-10		
			Min	Max	Min	Max	Min	Max	Min	Max	
f_{ACNT}	Maximum internal array clock frequency	*(4)*	175.4		142.9		116.3		100.0		MHz
f_{MAX}	Maximum clock frequency	*(5)*	250.0		200.0		166.7		125.0		MHz

Table 25. EPM7032S Internal Timing Parameters *Note (1)*

Symbol	Parameter	Conditions	Speed Grade								Unit
			-5		-6		-7		-10		
			Min	Max	Min	Max	Min	Max	Min	Max	
t_{IN}	Input pad and buffer delay			0.2		0.2		0.3		0.5	ns
t_{IO}	I/O input pad and buffer delay			0.2		0.2		0.3		0.5	ns
t_{FIN}	Fast input delay			2.2		2.1		2.5		1.0	ns
t_{SEXP}	Shared expander delay			3.1		3.8		4.6		5.0	ns
t_{PEXP}	Parallel expander delay			0.9		1.1		1.4		0.8	ns
t_{LAD}	Logic array delay			2.6		3.3		4.0		5.0	ns
t_{LAC}	Logic control array delay			2.5		3.3		4.0		5.0	ns
t_{IOE}	Internal output enable delay			0.7		0.8		1.0		2.0	ns
t_{OD1}	Output buffer and pad delay	C1 = 35 pF		0.2		0.3		0.4		1.5	ns
t_{OD2}	Output buffer and pad delay	C1 = 35 pF *(6)*		0.7		0.8		0.9		2.0	ns
t_{OD3}	Output buffer and pad delay	C1 = 35 pF		5.2		5.3		5.4		5.5	ns
t_{ZX1}	Output buffer enable delay	C1 = 35 pF		4.0		4.0		4.0		5.0	ns
t_{ZX2}	Output buffer enable delay	C1 = 35 pF *(6)*		4.5		4.5		4.5		5.5	ns
t_{ZX3}	Output buffer enable delay	C1 = 35 pF		9.0		9.0		9.0		9.0	ns
t_{XZ}	Output buffer disable delay	C1 = 5 pF		4.0		4.0		4.0		5.0	ns
t_{SU}	Register setup time		0.8		1.0		1.3		2.0		ns
t_{H}	Register hold time		1.7		2.0		2.5		3.0		ns
t_{FSU}	Register setup time of fast input		1.9		1.8		1.7		3.0		ns
t_{FH}	Register hold time of fast input		0.6		0.7		0.8		0.5		ns
t_{RD}	Register delay			1.2		1.6		1.9		2.0	ns
t_{COMB}	Combinatorial delay			0.9		1.1		1.4		2.0	ns
t_{IC}	Array clock delay			2.7		3.4		4.2		5.0	ns
t_{EN}	Register enable time			2.6		3.3		4.0		5.0	ns
t_{GLOB}	Global control delay			1.6		1.4		1.7		1.0	ns
t_{PRE}	Register preset time			2.0		2.4		3.0		3.0	ns

Table 25. EPM7032S Internal Timing Parameters *Note (1)*

Symbol	Parameter	Conditions	Speed Grade								Unit
			-5		-6		-7		-10		
			Min	Max	Min	Max	Min	Max	Min	Max	
t_{CLR}	Register clear time			2.0		2.4		3.0		3.0	ns
t_{PIA}	PIA delay	(7)		1.1		1.1		1.4		1.0	ns
t_{LPA}	Low-power adder	(8)		12.0		10.0		10.0		11.0	ns

Notes to tables:

(1) These values are specified under the recommended operating conditions shown in Table 11. See Figure 13 for more information on switching waveforms.

(2) This minimum pulse width for preset and clear applies for both global clear and array controls. The t_{LPA} parameter must be added to this minimum width if the clear or reset signal incorporates the t_{LAD} parameter into the signal path.

(3) This parameter is a guideline that is sample-tested only and is based on extensive device characterization. This parameter applies for both global and array clocking.

(4) These parameters are measured with a 16-bit loadable, enabled, up/down counter programmed into each LAB.

(5) The f_{MAX} values represent the highest frequency for pipelined data.

(6) Operating conditions: V_{CCIO} = 3.3 V ± 10% for commercial and industrial use.

(7) For EPM7064S-5, EPM7064S-6, EPM7128S-6, EPM7160S-6, EPM7160S-7, EPM7192S-7, and EPM7256S-7 devices, these values are specified for a PIA fan-out of one LAB (16 macrocells). For each additional LAB fan-out in these devices, add an additional 0.1 ns to the PIA timing value.

(8) The t_{LPA} parameter must be added to the t_{LAD}, t_{LAC}, t_{IC}, t_{EN}, t_{SEXP}, t_{ACL}, and t_{CPPW} parameters for macrocells running in the low-power mode.

Tables 26 and 27 show the EPM7064S AC operating conditions.

Table 26. EPM7064S External Timing Parameters (Part 1 of 2) *Note (1)*

Symbol	Parameter	Conditions	Speed Grade								Unit
			-5		-6		-7		-10		
			Min	Max	Min	Max	Min	Max	Min	Max	
t_{PD1}	Input to non-registered output	C1 = 35 pF		5.0		6.0		7.5		10.0	ns
t_{PD2}	I/O input to non-registered output	C1 = 35 pF		5.0		6.0		7.5		10.0	ns
t_{SU}	Global clock setup time		2.9		3.6		6.0		7.0		ns
t_{H}	Global clock hold time		0.0		0.0		0.0		0.0		ns
t_{FSU}	Global clock setup time of fast input		2.5		2.5		3.0		3.0		ns
t_{FH}	Global clock hold time of fast input		0.0		0.0		0.5		0.5		ns
t_{CO1}	Global clock to output delay	C1 = 35 pF		3.2		4.0		4.5		5.0	ns
t_{CH}	Global clock high time		2.0		2.5		3.0		4.0		ns
t_{CL}	Global clock low time		2.0		2.5		3.0		4.0		ns
t_{ASU}	Array clock setup time		0.7		0.9		3.0		2.0		ns

Table 26. EPM7064S External Timing Parameters (Part 2 of 2) *Note (1)*

Symbol	Parameter	Conditions	Speed Grade								Unit
			-5		-6		-7		-10		
			Min	Max	Min	Max	Min	Max	Min	Max	
t_{AH}	Array clock hold time		1.8		2.1		2.0		3.0		ns
t_{ACO1}	Array clock to output delay	C1 = 35 pF		5.4		6.7		7.5		10.0	ns
t_{ACH}	Array clock high time		2.5		2.5		3.0		4.0		ns
t_{ACL}	Array clock low time		2.5		2.5		3.0		4.0		ns
t_{CPPW}	Minimum pulse width for clear and preset	*(2)*	2.5		2.5		3.0		4.0		ns
t_{ODH}	Output data hold time after clock	C1 = 35 pF *(3)*	1.0		1.0		1.0		1.0		ns
t_{CNT}	Minimum global clock period			5.7		7.1		8.0		10.0	ns
f_{CNT}	Maximum internal global clock frequency	*(4)*	175.4		140.8		125.0		100.0		MHz
t_{ACNT}	Minimum array clock period			5.7		7.1		8.0		10.0	ns
f_{ACNT}	Maximum internal array clock frequency	*(4)*	175.4		140.8		125.0		100.0		MHz
f_{MAX}	Maximum clock frequency	*(5)*	250.0		200.0		166.7		125.0		MHz

Table 27. EPM7064S Internal Timing Parameters (Part 1 of 2) *Note (1)*

Symbol	Parameter	Conditions	Speed Grade								Unit
			-5		-6		-7		-10		
			Min	Max	Min	Max	Min	Max	Min	Max	
t_{IN}	Input pad and buffer delay			0.2		0.2		0.5		0.5	ns
t_{IO}	I/O input pad and buffer delay			0.2		0.2		0.5		0.5	ns
t_{FIN}	Fast input delay			2.2		2.6		1.0		1.0	ns
t_{SEXP}	Shared expander delay			3.1		3.8		4.0		5.0	ns
t_{PEXP}	Parallel expander delay			0.9		1.1		0.8		0.8	ns
t_{LAD}	Logic array delay			2.6		3.2		3.0		5.0	ns
t_{LAC}	Logic control array delay			2.5		3.2		3.0		5.0	ns
t_{IOE}	Internal output enable delay			0.7		0.8		2.0		2.0	ns
t_{OD1}	Output buffer and pad delay	C1 = 35 pF		0.2		0.3		2.0		1.5	ns
t_{OD2}	Output buffer and pad delay	C1 = 35 pF *(6)*		0.7		0.8		2.5		2.0	ns
t_{OD3}	Output buffer and pad delay	C1 = 35 pF		5.2		5.3		7.0		5.5	ns
t_{ZX1}	Output buffer enable delay	C1 = 35 pF		4.0		4.0		4.0		5.0	ns
t_{ZX2}	Output buffer enable delay	C1 = 35 pF *(6)*		4.5		4.5		4.5		5.5	ns
t_{ZX3}	Output buffer enable delay	C1 = 35 pF		9.0		9.0		9.0		9.0	ns
t_{XZ}	Output buffer disable delay	C1 = 5 pF		4.0		4.0		4.0		5.0	ns
t_{SU}	Register setup time		0.8		1.0		3.0		2.0		ns

Table 27. EPM7064S Internal Timing Parameters (Part 2 of 2) *Note (1)*

Symbol	Parameter	Conditions	Speed Grade								Unit
			-5		-6		-7		-10		
			Min	Max	Min	Max	Min	Max	Min	Max	
t_{IH}	Register hold time		1.7		2.0		2.0		3.0		ns
t_{FSU}	Register setup time of fast input		1.9		1.8		3.0		3.0		ns
t_{FH}	Register hold time of fast input		0.6		0.7		0.5		0.5		ns
t_{RD}	Register delay			1.2		1.6		1.0		2.0	ns
t_{COMB}	Combinatorial delay			0.9		1.0		1.0		2.0	ns
t_{IC}	Array clock delay			2.7		3.3		3.0		5.0	ns
t_{EN}	Register enable time			2.6		3.2		3.0		5.0	ns
t_{GLOB}	Global control delay			1.6		1.9		1.0		1.0	ns
t_{PRE}	Register preset time			2.0		2.4		2.0		3.0	ns
t_{CLR}	Register clear time			2.0		2.4		2.0		3.0	ns
t_{PIA}	PIA delay	*(7)*		1.1		1.3		1.0		1.0	ns
t_{LPA}	Low-power adder	*(8)*		12.0		11.0		10.0		11.0	ns

Notes to tables:

(1) These values are specified under the recommended operating conditions shown in Table 11. See Figure 13 for more information on switching waveforms.

(2) This minimum pulse width for preset and clear applies for both global clear and array controls. The t_{LPA} parameter must be added to this minimum width if the clear or reset signal incorporates the t_{LAD} parameter into the signal path.

(3) This parameter is a guideline that is sample-tested only and is based on extensive device characterization. This parameter applies for both global and array clocking.

(4) These parameters are measured with a 16-bit loadable, enabled, up/down counter programmed into each LAB.

(5) The f_{MAX} values represent the highest frequency for pipelined data.

(6) Operating conditions: V_{CCIO} = 3.3 V ± 10% for commercial and industrial use.

(7) For EPM7064S-5, EPM7064S-6, EPM7128S-6, EPM7160S-6, EPM7160S-7, EPM7192S-7, and EPM7256S-7 devices, these values are specified for a PIA fan-out of one LAB (16 macrocells). For each additional LAB fan-out in these devices, add an additional 0.1 ns to the PIA timing value.

(8) The t_{LPA} parameter must be added to the t_{LAD}, t_{LAC}, t_{IC}, t_{EN}, t_{SEXP}, t_{ACL}, and t_{CPPW} parameters for macrocells running in the low-power mode.

Tables 28 and 29 show the EPM7128S AC operating conditions.

Table 28. EPM7128S External Timing Parameters *Note (1)*

Symbol	Parameter	Conditions	Speed Grade								Unit
			-6		-7		-10		-15		
			Min	Max	Min	Max	Min	Max	Min	Max	
t_{PD1}	Input to non-registered output	C1 = 35 pF		6.0		7.5		10.0		15.0	ns
t_{PD2}	I/O input to non-registered output	C1 = 35 pF		6.0		7.5		10.0		15.0	ns
t_{SU}	Global clock setup time		3.4		6.0		7.0		11.0		ns
t_{H}	Global clock hold time		0.0		0.0		0.0		0.0		ns
t_{FSU}	Global clock setup time of fast input		2.5		3.0		3.0		3.0		ns
t_{FH}	Global clock hold time of fast input		0.0		0.5		0.5		0.0		ns
t_{CO1}	Global clock to output delay	C1 = 35 pF		4.0		4.5		5.0		8.0	ns
t_{CH}	Global clock high time		3.0		3.0		4.0		5.0		ns
t_{CL}	Global clock low time		3.0		3.0		4.0		5.0		ns
t_{ASU}	Array clock setup time		0.9		3.0		2.0		4.0		ns
t_{AH}	Array clock hold time		1.8		2.0		5.0		4.0		ns
t_{ACO1}	Array clock to output delay	C1 = 35 pF		6.5		7.5		10.0		15.0	ns
t_{ACH}	Array clock high time		3.0		3.0		4.0		6.0		ns
t_{ACL}	Array clock low time		3.0		3.0		4.0		6.0		ns
t_{CPPW}	Minimum pulse width for clear and preset	*(2)*	3.0		3.0		4.0		6.0		ns
t_{ODH}	Output data hold time after clock	C1 = 35 pF *(3)*	1.0		1.0		1.0		1.0		ns
t_{CNT}	Minimum global clock period			6.8		8.0		10.0		13.0	ns
f_{CNT}	Maximum internal global clock frequency	*(4)*	147.1		125.0		100.0		76.9		MHz
t_{ACNT}	Minimum array clock period			6.8		8.0		10.0		13.0	ns
f_{ACNT}	Maximum internal array clock frequency	*(4)*	147.1		125.0		100.0		76.9		MHz
f_{MAX}	Maximum clock frequency	*(5)*	166.7		166.7		125.0		100.0		MHz

Table 29. EPM7128S Internal Timing Parameters *Note (1)*

Symbol	Parameter	Conditions	Speed Grade								Unit
			-6		-7		-10		-15		
			Min	Max	Min	Max	Min	Max	Min	Max	
t_{IN}	Input pad and buffer delay			0.2		0.5		0.5		2.0	ns
t_{IO}	I/O input pad and buffer delay			0.2		0.5		0.5		2.0	ns
t_{FIN}	Fast input delay			2.6		1.0		1.0		2.0	ns
t_{SEXP}	Shared expander delay			3.7		4.0		5.0		8.0	ns
t_{PEXP}	Parallel expander delay			1.1		0.8		0.8		1.0	ns
t_{LAD}	Logic array delay			3.0		3.0		5.0		6.0	ns
t_{LAC}	Logic control array delay			3.0		3.0		5.0		6.0	ns
t_{IOE}	Internal output enable delay			0.7		2.0		2.0		3.0	ns
t_{OD1}	Output buffer and pad delay	C1 = 35 pF		0.4		2.0		1.5		4.0	ns
t_{OD2}	Output buffer and pad delay	C1 = 35 pF *(6)*		0.9		2.5		2.0		5.0	ns
t_{OD3}	Output buffer and pad delay	C1 = 35 pF		5.4		7.0		5.5		8.0	ns
t_{ZX1}	Output buffer enable delay	C1 = 35 pF		4.0		4.0		5.0		6.0	ns
t_{ZX2}	Output buffer enable delay	C1 = 35 pF *(6)*		4.5		4.5		5.5		7.0	ns
t_{ZX3}	Output buffer enable delay	C1 = 35 pF		9.0		9.0		9.0		10.0	ns
t_{XZ}	Output buffer disable delay	C1 = 5 pF		4.0		4.0		5.0		6.0	ns
t_{SU}	Register setup time		1.0		3.0		2.0		4.0		ns
t_{H}	Register hold time		1.7		2.0		5.0		4.0		ns
t_{FSU}	Register setup time of fast input		1.9		3.0		3.0		2.0		ns
t_{FH}	Register hold time of fast input		0.6		0.5		0.5		1.0		ns
t_{RD}	Register delay			1.4		1.0		2.0		1.0	ns
t_{COMB}	Combinatorial delay			1.0		1.0		2.0		1.0	ns
t_{IC}	Array clock delay			3.1		3.0		5.0		6.0	ns
t_{EN}	Register enable time			3.0		3.0		5.0		6.0	ns
t_{GLOB}	Global control delay			2.0		1.0		1.0		1.0	ns
t_{PRE}	Register preset time			2.4		2.0		3.0		4.0	ns
t_{CLR}	Register clear time			2.4		2.0		3.0		4.0	ns
t_{PIA}	PIA delay	*(7)*		1.4		1.0		1.0		2.0	ns
t_{LPA}	Low-power adder	*(8)*		11.0		10.0		11.0		13.0	ns

Notes to tables:

(1) These values are specified under the recommended operating conditions shown in Table 11. See Figure 13 for more information on switching waveforms.
(2) This minimum pulse width for preset and clear applies for both global clear and array controls. The t_{LPA} parameter must be added to this minimum width if the clear or reset signal incorporates the t_{LAD} parameter into the signal path.
(3) This parameter is a guideline that is sample-tested only and is based on extensive device characterization. This parameter applies for both global and array clocking.
(4) These parameters are measured with a 16-bit loadable, enabled, up/down counter programmed into each LAB.
(5) The f_{MAX} values represent the highest frequency for pipelined data.
(6) Operating conditions: V_{CCIO} = 3.3 V ± 10% for commercial and industrial use.
(7) For EPM7064S-5, EPM7064S-6, EPM7128S-6, EPM7160S-6, EPM7160S-7, EPM7192S-7, and EPM7256S-7 devices, these values are specified for a PIA fan-out of one LAB (16 macrocells). For each additional LAB fan-out in these devices, add an additional 0.1 ns to the PIA timing value.
(8) The t_{LPA} parameter must be added to the t_{LAD}, t_{LAC}, t_{IC}, t_{EN}, t_{SEXP}, t_{ACL}, and t_{CPPW} parameters for macrocells running in the low-power mode.

Tables 30 and 31 show the EPM7160S AC operating conditions.

Table 30. EPM7160S External Timing Parameters (Part 1 of 2) *Note (1)*

Symbol	Parameter	Conditions	Speed Grade								Unit
			-6		-7		-10		-15		
			Min	Max	Min	Max	Min	Max	Min	Max	
t_{PD1}	Input to non-registered output	C1 = 35 pF		6.0		7.5		10.0		15.0	ns
t_{PD2}	I/O input to non-registered output	C1 = 35 pF		6.0		7.5		10.0		15.0	ns
t_{SU}	Global clock setup time		3.4		4.2		7.0		11.0		ns
t_{H}	Global clock hold time		0.0		0.0		0.0		0.0		ns
t_{FSU}	Global clock setup time of fast input		2.5		3.0		3.0		3.0		ns
t_{FH}	Global clock hold time of fast input		0.0		0.0		0.5		0.0		ns
t_{CO1}	Global clock to output delay	C1 = 35 pF		3.9		4.8		5		8	ns
t_{CH}	Global clock high time		3.0		3.0		4.0		5.0		ns
t_{CL}	Global clock low time		3.0		3.0		4.0		5.0		ns
t_{ASU}	Array clock setup time		0.9		1.1		2.0		4.0		ns
t_{AH}	Array clock hold time		1.7		2.1		3.0		4.0		ns
t_{ACO1}	Array clock to output delay	C1 = 35 pF		6.4		7.9		10.0		15.0	ns
t_{ACH}	Array clock high time		3.0		3.0		4.0		6.0		ns
t_{ACL}	Array clock low time		3.0		3.0		4.0		6.0		ns
t_{CPPW}	Minimum pulse width for clear and preset	*(7)*	2.5		3.0		4.0		6.0		ns
t_{ODH}	Output data hold time after clock	C1 = 35 pF *(3)*	1.0		1.0		1.0		1.0		ns
t_{CNT}	Minimum global clock period			6.7		8.2		10.0		13.0	ns
f_{CNT}	Maximum internal global clock frequency	*(4)*	149.3		122.0		100.0		76.9		MHz

Table 30. EPM7160S External Timing Parameters (Part 2 of 2) *Note (1)*

Symbol	Parameter	Conditions	Speed Grade								Unit
			-6		-7		-10		-15		
			Min	Max	Min	Max	Min	Max	Min	Max	
t_{ACNT}	Minimum array clock period			6.7		8.2		10.0		13.0	ns
f_{ACNT}	Maximum internal array clock frequency	*(4)*	149.3		122.0		100.0		76.9		MHz
f_{MAX}	Maximum clock frequency	*(5)*	166.7		166.7		125.0		100.0		MHz

Table 31. EPM7160S Internal Timing Parameters (Part 1 of 2) *Note (1)*

Symbol	Parameter	Conditions	Speed Grade								Unit
			-6		-7		-10		-15		
			Min	Max	Min	Max	Min	Max	Min	Max	
t_{IN}	Input pad and buffer delay			0.2		0.3		0.5		2.0	ns
t_{IO}	I/O input pad and buffer delay			0.2		0.3		0.5		2.0	ns
t_{FIN}	Fast input delay			2.6		3.2		1.0		2.0	ns
t_{SEXP}	Shared expander delay			3.6		4.3		5.0		8.0	ns
t_{PEXP}	Parallel expander delay			1.0		1.3		0.8		1.0	ns
t_{LAD}	Logic array delay			2.8		3.4		5.0		6.0	ns
t_{LAC}	Logic control array delay			2.8		3.4		5.0		6.0	ns
t_{IOE}	Internal output enable delay			0.7		0.9		2.0		3.0	ns
t_{OD1}	Output buffer and pad delay	C1 = 35 pF		0.4		0.5		1.5		4.0	ns
t_{OD2}	Output buffer and pad delay	C1 = 35 pF *(6)*		0.9		1.0		2.0		5.0	ns
t_{OD3}	Output buffer and pad delay	C1 = 35 pF		5.4		5.5		5.5		8.0	ns
t_{ZX1}	Output buffer enable delay	C1 = 35 pF		4.0		4.0		5.0		6.0	ns
t_{ZX2}	Output buffer enable delay	C1 = 35 pF *(6)*		4.5		4.5		5.5		7.0	ns
t_{ZX3}	Output buffer enable delay	C1 = 35 pF		9.0		9.0		9.0		10.0	ns
t_{XZ}	Output buffer disable delay	C1 = 5 pF		4.0		4.0		5.0		6.0	ns
t_{SU}	Register setup time		1.0		1.2		2.0		4.0		ns
t_{H}	Register hold time		1.6		2.0		3.0		4.0		ns
t_{FSU}	Register setup time of fast input		1.9		2.2		3.0		2.0		ns
t_{FH}	Register hold time of fast input		0.6		0.8		0.5		1.0		ns
t_{RD}	Register delay			1.3		1.6		2.0		1.0	ns
t_{COMB}	Combinatorial delay			1.0		1.3		2.0		1.0	ns
t_{IC}	Array clock delay			2.9		3.5		5.0		6.0	ns
t_{EN}	Register enable time			2.8		3.4		5.0		6.0	ns
t_{GLOB}	Global control delay			2.0		2.4		1.0		1.0	ns

Table 31. EPM7160S Internal Timing Parameters (Part 2 of 2) *Note (1)*

Symbol	Parameter	Conditions	Speed Grade								Unit
			-6		-7		-10		-15		
			Min	Max	Min	Max	Min	Max	Min	Max	
t_{PRE}	Register preset time			2.4		3.0		3.0		4.0	ns
t_{CLR}	Register clear time			2.4		3.0		3.0		4.0	ns
t_{PIA}	PIA delay	(7)		1.6		2.0		1.0		2.0	ns
t_{LPA}	Low-power adder	(8)		11.0		10.0		11.0		13.0	ns

Notes to tables:

(1) These values are specified under the recommended operating conditions shown in Table 11. See Figure 13 for more information on switching waveforms.
(2) This minimum pulse width for preset and clear applies for both global clear and array controls. The t_{LPA} parameter must be added to this minimum width if the clear or reset signal incorporates the t_{LAD} parameter into the signal path.
(3) This parameter is a guideline that is sample-tested only and is based on extensive device characterization. This parameter applies for both global and array clocking.
(4) These parameters are measured with a 16-bit loadable, enabled, up/down counter programmed into each LAB.
(5) The f_{MAX} values represent the highest frequency for pipelined data.
(6) Operating conditions: V_{CCIO} = 3.3 V ± 10% for commercial and industrial use.
(7) For EPM7064S-5, EPM7064S-6, EPM7128S-6, EPM7160S-6, EPM7160S-7, EPM7192S-7, and EPM7256S-7 devices, these values are specified for a PIA fan-out of one LAB (16 macrocells). For each additional LAB fan-out in these devices, add an additional 0.1 ns to the PIA timing value.
(8) The t_{LPA} parameter must be added to the t_{LAD}, t_{LAC}, t_{IC}, t_{EN}, t_{SEXP}, t_{ACL}, and t_{CPPW} parameters for macrocells running in the low-power mode.

Tables 32 and 33 show the EPM7192S AC operating conditions.

Table 32. EPM7192S External Timing Parameters (Part 1 of 2) *Note (1)*

Symbol	Parameter	Conditions	Speed Grade						Unit
			-7		-10		-15		
			Min	Max	Min	Max	Min	Max	
t_{PD1}	Input to non-registered output	C1 = 35 pF		7.5		10.0		15.0	ns
t_{PD2}	I/O input to non-registered output	C1 = 35 pF		7.5		10.0		15.0	ns
t_{SU}	Global clock setup time		4.1		7.0		11.0		ns
t_{H}	Global clock hold time		0.0		0.0		0.0		ns
t_{FSU}	Global clock setup time of fast input		3.0		3.0		3.0		ns
t_{FH}	Global clock hold time of fast input		0.0		0.5		0.0		ns
t_{CO1}	Global clock to output delay	C1 = 35 pF		4.7		5.0		8.0	ns
t_{CH}	Global clock high time		3.0		4.0		5.0		ns

Table 32. EPM7192S External Timing Parameters (Part 2 of 2) *Note (1)*

Symbol	Parameter	Conditions	Speed Grade						Unit
			-7		-10		-15		
			Min	Max	Min	Max	Min	Max	
t_{CL}	Global clock low time		3.0		4.0		5.0		ns
t_{ASU}	Array clock setup time		1.0		2.0		4.0		ns
t_{AH}	Array clock hold time		1.8		3.0		4.0		ns
t_{ACO1}	Array clock to output delay	C1 = 35 pF		7.8		10.0		15.0	ns
t_{ACH}	Array clock high time		3.0		4.0		6.0		ns
t_{ACL}	Array clock low time		3.0		4.0		6.0		ns
t_{CPPW}	Minimum pulse width for clear and preset	*(2)*	3.0		4.0		6.0		ns
t_{ODH}	Output data hold time after clock	C1 = 35 pF *(3)*	1.0		1.0		1.0		ns
t_{CNT}	Minimum global clock period			8.0		10.0		13.0	ns
f_{CNT}	Maximum internal global clock frequency	*(4)*	125.0		100.0		76.9		MHz
t_{ACNT}	Minimum array clock period			8.0		10.0		13.0	ns
f_{ACNT}	Maximum internal array clock frequency	*(4)*	125.0		100.0		76.9		MHz
f_{MAX}	Maximum clock frequency	*(5)*	166.7		125.0		100.0		MHz

Table 33. EPM7192S Internal Timing Parameters (Part 1 of 2) *Note (1)*

Symbol	Parameter	Conditions	Speed Grade						Unit
			-7		-10		-15		
			Min	Max	Min	Max	Min	Max	
t_{IN}	Input pad and buffer delay			0.3		0.5		2.0	ns
t_{IO}	I/O input pad and buffer delay			0.3		0.5		2.0	ns
t_{FIN}	Fast input delay			3.2		1.0		2.0	ns
t_{SEXP}	Shared expander delay			4.2		5.0		8.0	ns
t_{PEXP}	Parallel expander delay			1.2		0.8		1.0	ns
t_{LAD}	Logic array delay			3.1		5.0		6.0	ns
t_{LAC}	Logic control array delay			3.1		5.0		6.0	ns
t_{IOE}	Internal output enable delay			0.9		2.0		3.0	ns
t_{OD1}	Output buffer and pad delay	C1 = 35 pF		0.5		1.5		4.0	ns
t_{OD2}	Output buffer and pad delay	C1 = 35 pF *(6)*		1.0		2.0		5.0	ns
t_{OD3}	Output buffer and pad delay	C1 = 35 pF		5.5		5.5		7.0	ns
t_{ZX1}	Output buffer enable delay	C1 = 35 pF		4.0		5.0		6.0	ns
t_{ZX2}	Output buffer enable delay	C1 = 35 pF *(6)*		4.5		5.5		7.0	ns
t_{ZX3}	Output buffer enable delay	C1 = 35 pF		9.0		9.0		10.0	ns

Table 33. EPM7192S Internal Timing Parameters (Part 2 of 2) *Note (1)*

Symbol	Parameter	Conditions	Speed Grade						Unit
			-7		-10		-15		
			Min	Max	Min	Max	Min	Max	
t_{XZ}	Output buffer disable delay	C1 = 5 pF		4.0		5.0		6.0	ns
t_{SU}	Register setup time		1.1		2.0		4.0		ns
t_{H}	Register hold time		1.7		3.0		4.0		ns
t_{FSU}	Register setup time of fast input		2.3		3.0		2.0		ns
t_{FH}	Register hold time of fast input		0.7		0.5		1.0		ns
t_{RD}	Register delay			1.4		2.0		1.0	ns
t_{COMB}	Combinatorial delay			1.2		2.0		1.0	ns
t_{IC}	Array clock delay			3.2		5.0		6.0	ns
t_{EN}	Register enable time			3.1		5.0		6.0	ns
t_{GLOB}	Global control delay			2.5		1.0		1.0	ns
t_{PRE}	Register preset time			2.7		3.0		4.0	ns
t_{CLR}	Register clear time			2.7		3.0		4.0	ns
t_{PIA}	PIA delay	*(7)*		2.4		1.0		2.0	ns
t_{LPA}	Low-power adder	*(8)*		10.0		11.0		13.0	ns

Notes to tables:

(1) These values are specified under the recommended operating conditions shown in Table 11. See Figure 13 for more information on switching waveforms.

(2) This minimum pulse width for preset and clear applies for both global clear and array controls. The t_{LPA} parameter must be added to this minimum width if the clear or reset signal incorporates the t_{LAD} parameter into the signal path.

(3) This parameter is a guideline that is sample-tested only and is based on extensive device characterization. This parameter applies for both global and array clocking.

(4) These parameters are measured with a 16-bit loadable, enabled, up/down counter programmed into each LAB.

(5) The f_{MAX} values represent the highest frequency for pipelined data.

(6) Operating conditions: V_{CCIO} = 3.3 V ± 10% for commercial and industrial use.

(7) For EPM7064S-5, EPM7064S-6, EPM7128S-6, EPM7160S-6, EPM7160S-7, EPM7192S-7, and EPM7256S-7 devices, these values are specified for a PIA fan-out of one LAB (16 macrocells). For each additional LAB fan-out in these devices, add an additional 0.1 ns to the PIA timing value.

(8) The t_{LPA} parameter must be added to the t_{LAD}, t_{LAC}, t_{IC}, t_{EN}, t_{SEXP}, t_{ACL}, and t_{CPPW} parameters for macrocells running in the low-power mode.

Tables 34 and 35 show the EPM7256S AC operating conditions.

Table 34. EPM7256S External Timing Parameters *Note (1)*

Symbol	Parameter	Conditions	Speed Grade						Unit
			-7		-10		-15		
			Min	Max	Min	Max	Min	Max	
t_{PD1}	Input to non-registered output	C1 = 35 pF		7.5		10.0		15.0	ns
t_{PD2}	I/O input to non-registered output	C1 = 35 pF		7.5		10.0		15.0	ns
t_{SU}	Global clock setup time		3.9		7.0		11.0		ns
t_{H}	Global clock hold time		0.0		0.0		0.0		ns
t_{FSU}	Global clock setup time of fast input		3.0		3.0		3.0		ns
t_{FH}	Global clock hold time of fast input		0.0		0.5		0.0		ns
t_{CO1}	Global clock to output delay	C1 = 35 pF		4.7		5.0		8.0	ns
t_{CH}	Global clock high time		3.0		4.0		5.0		ns
t_{CL}	Global clock low time		3.0		4.0		5.0		ns
t_{ASU}	Array clock setup time		0.8		2.0		4.0		ns
t_{AH}	Array clock hold time		1.9		3.0		4.0		ns
t_{ACO1}	Array clock to output delay	C1 = 35 pF		7.8		10.0		15.0	ns
t_{ACH}	Array clock high time		3.0		4.0		6.0		ns
t_{ACL}	Array clock low time		3.0		4.0		6.0		ns
t_{CPPW}	Minimum pulse width for clear and preset	*(2)*	3.0		4.0		6.0		ns
t_{ODH}	Output data hold time after clock	C1 = 35 pF *(3)*	1.0		1.0		1.0		ns
t_{CNT}	Minimum global clock period			7.8		10.0		13.0	ns
f_{CNT}	Maximum internal global clock frequency	*(4)*	128.2		100.0		76.9		MHz
t_{ACNT}	Minimum array clock period			7.8		10.0		13.0	ns
f_{ACNT}	Maximum internal array clock frequency	*(4)*	128.2		100.0		76.9		MHz
f_{MAX}	Maximum clock frequency	*(5)*	166.7		125.0		100.0		MHz

Table 35. EPM7256S Internal Timing Parameters *Note (1)*

Symbol	Parameter	Conditions	Speed Grade						Unit
			-7		-10		-15		
			Min	Max	Min	Max	Min	Max	
t_{IN}	Input pad and buffer delay			0.3		0.5		2.0	ns
t_{IO}	I/O input pad and buffer delay			0.3		0.5		2.0	ns
t_{FIN}	Fast input delay			3.4		1.0		2.0	ns
t_{SEXP}	Shared expander delay			3.9		5.0		8.0	ns
t_{PEXP}	Parallel expander delay			1.1		0.8		1.0	ns
t_{LAD}	Logic array delay			2.6		5.0		6.0	ns
t_{LAC}	Logic control array delay			2.6		5.0		6.0	ns
t_{IOE}	Internal output enable delay			0.8		2.0		3.0	ns
t_{OD1}	Output buffer and pad delay	C1 = 35 pF		0.5		1.5		4.0	ns
t_{OD2}	Output buffer and pad delay	C1 = 35 pF *(6)*		1.0		2.0		5.0	ns
t_{OD3}	Output buffer and pad delay	C1 = 35 pF		5.5		5.5		8.0	ns
t_{ZX1}	Output buffer enable delay	C1 = 35 pF		4.0		5.0		6.0	ns
t_{ZX2}	Output buffer enable delay	C1 = 35 pF *(6)*		4.5		5.5		7.0	ns
t_{ZX3}	Output buffer enable delay	C1 = 35 pF		9.0		9.0		10.0	ns
t_{XZ}	Output buffer disable delay	C1 = 5 pF		4.0		5.0		6.0	ns
t_{SU}	Register setup time		1.1		2.0		4.0		ns
t_{H}	Register hold time		1.6		3.0		4.0		ns
t_{FSU}	Register setup time of fast input		2.4		3.0		2.0		ns
t_{FH}	Register hold time of fast input		0.6		0.5		1.0		ns
t_{RD}	Register delay			1.1		2.0		1.0	ns
t_{COMB}	Combinatorial delay			1.1		2.0		1.0	ns
t_{IC}	Array clock delay			2.9		5.0		6.0	ns
t_{EN}	Register enable time			2.6		5.0		6.0	ns
t_{GLOB}	Global control delay			2.8		1.0		1.0	ns
t_{PRE}	Register preset time			2.7		3.0		4.0	ns
t_{CLR}	Register clear time			2.7		3.0		4.0	ns
t_{PIA}	PIA delay	*(7)*		3.0		1.0		2.0	ns
t_{LPA}	Low-power adder	*(8)*		10.0		11.0		13.0	ns

Notes to tables:

(1) These values are specified under the recommended operating conditions shown in Table 11. See Figure 13 for more information on switching waveforms.
(2) This minimum pulse width for preset and clear applies for both global clear and array controls. The t_{LPA} parameter must be added to this minimum width if the clear or reset signal incorporates the t_{LAD} parameter into the signal path.
(3) This parameter is a guideline that is sample-tested only and is based on extensive device characterization. This parameter applies for both global and array clocking.
(4) These parameters are measured with a 16-bit loadable, enabled, up/down counter programmed into each LAB.
(5) The f_{MAX} values represent the highest frequency for pipelined data.
(6) Operating conditions: V_{CCIO} = 3.3 V ± 10% for commercial and industrial use.
(7) For EPM7064S-5, EPM7064S-6, EPM7128S-6, EPM7160S-6, EPM7160S-7, EPM7192S-7, and EPM7256S-7 devices, these values are specified for a PIA fan-out of one LAB (16 macrocells). For each additional LAB fan-out in these devices, add an additional 0.1 ns to the PIA timing value.
(8) The t_{LPA} parameter must be added to the t_{LAD}, t_{LAC}, t_{IC}, t_{EN}, t_{SEXP}, t_{ACL}, and t_{CPPW} parameters for macrocells running in the low-power mode.

Power Consumption

Supply power (P) versus frequency (f_{MAX} in MHz) for MAX 7000 devices is calculated with the following equation:

$$P = P_{INT} + P_{IO} = I_{CCINT} \times V_{CC} + P_{IO}$$

The P_{IO} value, which depends on the device output load characteristics and switching frequency, can be calculated using the guidelines given in *Application Note 74 (Evaluating Power for Altera Devices)*.

The I_{CCINT} value, which depends on the switching frequency and the application logic, is calculated with the following equation:

$$I_{CCINT} =$$

$$A \times MC_{TON} + B \times (MC_{DEV} - MC_{TON}) + C \times MC_{USED} \times f_{MAX} \times tog_{LC}$$

The parameters in this equation are shown below:

MC_{TON}	=	Number of macrocells with the Turbo Bit option turned on, as reported in the MAX+PLUS II Report File (.rpt)
MC_{DEV}	=	Number of macrocells in the device
MC_{USED}	=	Total number of macrocells in the design, as reported in the MAX+PLUS II Report File (.rpt)
f_{MAX}	=	Highest clock frequency to the device
tog_{LC}	=	Average ratio of logic cells toggling at each clock (typically 0.125)
A, B, C	=	Constants, shown in Table 36

Table 36. MAX 7000 I_{CC} Equation Constants

Device	A	B	C
EPM7032	1.87	0.52	0.144
EPM7064	1.63	0.74	0.144
EPM7096	1.63	0.74	0.144
EPM7128E	1.17	0.54	0.096
EPM7160E	1.17	0.54	0.096
EPM7192E	1.17	0.54	0.096
EPM7256E	1.17	0.54	0.096
EPM7032S	0.93	0.40	0.040
EPM7064S	0.93	0.40	0.040
EPM7128S	0.93	0.40	0.040
EPM7160S	0.93	0.40	0.040
EPM7192S	0.93	0.40	0.040
EPM7256S	0.93	0.40	0.040

This calculation provides an I_{CC} estimate based on typical conditions using a pattern of a 16-bit, loadable, enabled, up/down counter in each LAB with no output load. Actual I_{CC} values should be verified during operation because this measurement is sensitive to the actual pattern in the device and the environmental operating conditions.

Figure 14 shows typical supply current versus frequency for MAX 7000 devices.

Figure 14. I_{CC} vs. Frequency for MAX 7000 Devices (Part 1 of 2)

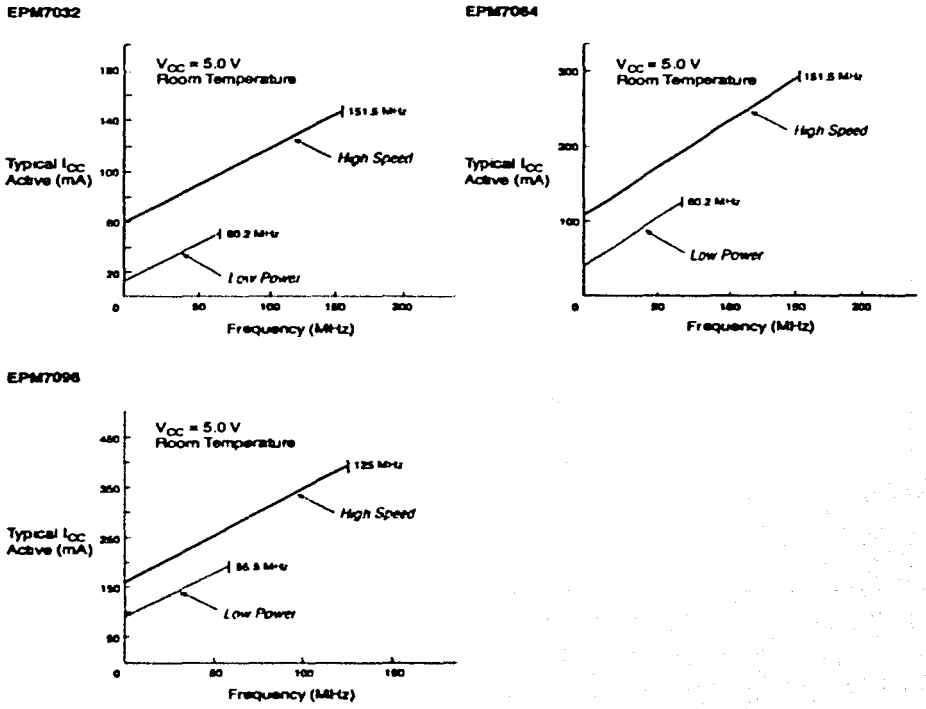
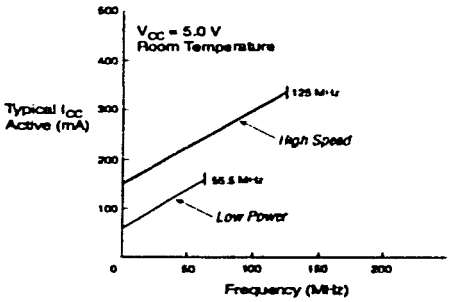
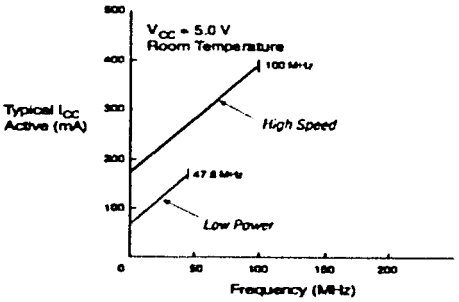


Figure 14. I_{CC} vs. Frequency for MAX 7000 Devices (Part 2 of 2)

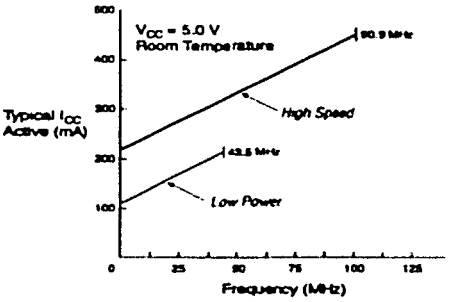
EPM7128E



EPM7160E



EPM7192E



EPM7256E

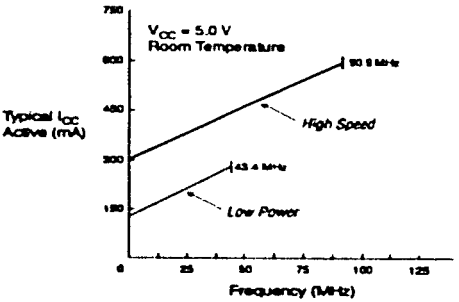


Figure 15 shows typical supply current versus frequency for MAX 7000S devices.

Figure 15. I_{CC} vs. Frequency for MAX 7000S Devices (Part 1 of 2)

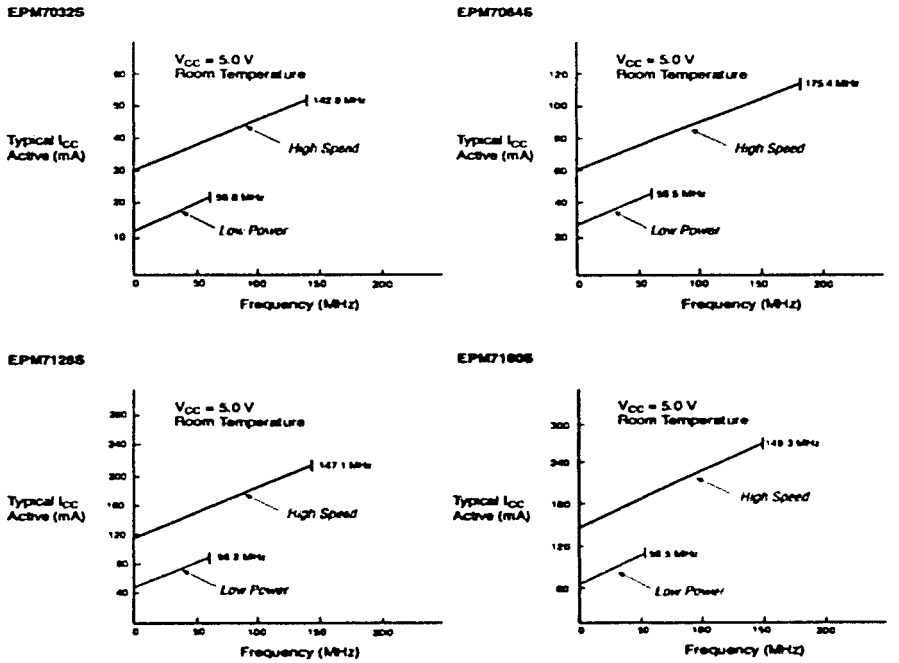
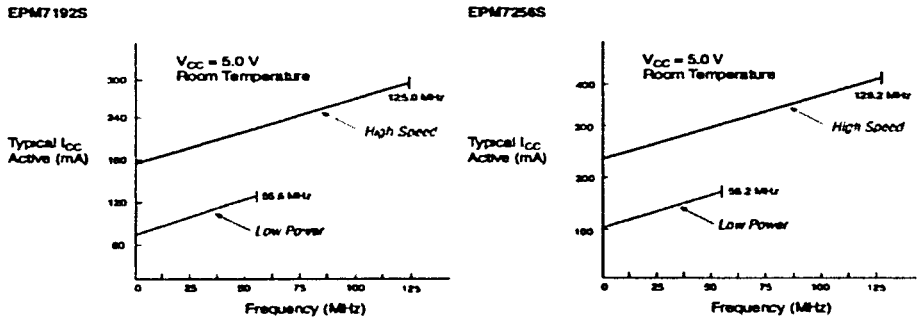


Figure 15. I_{CC} vs. Frequency for MAX 7000S Devices (Part 2 of 2)



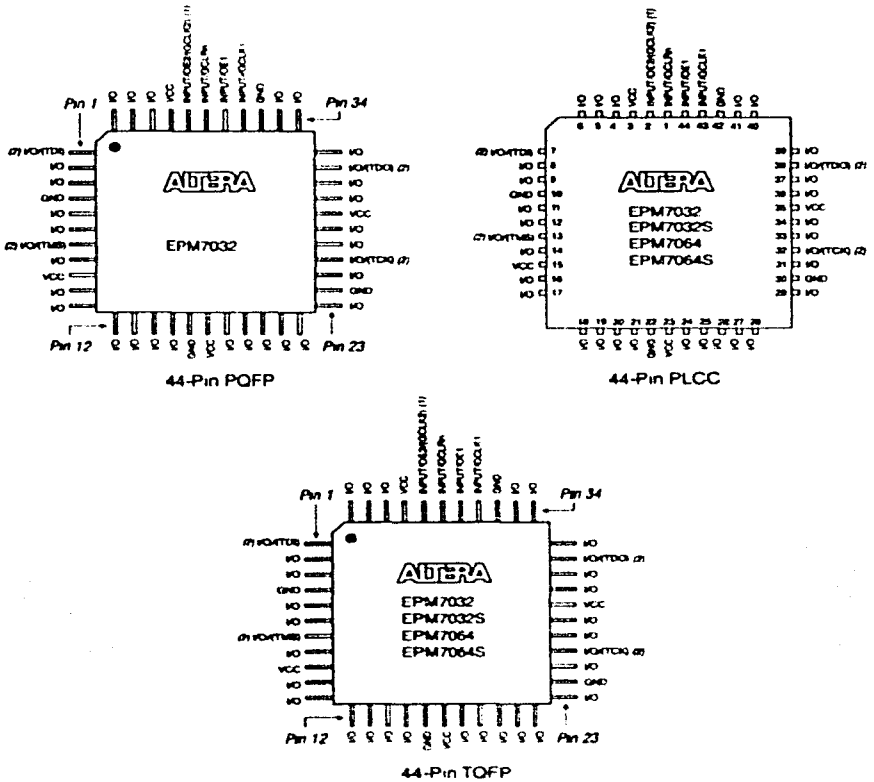
Device Pin-Outs

See the Altera web site (http://www.altera.com) or the *Altera Digital Library* for pin-out information.

Figures 16 through 22 show the package pin-out diagrams for MAX 7000 devices.

Figure 16. 44-Pin Package Pin-Out Diagram

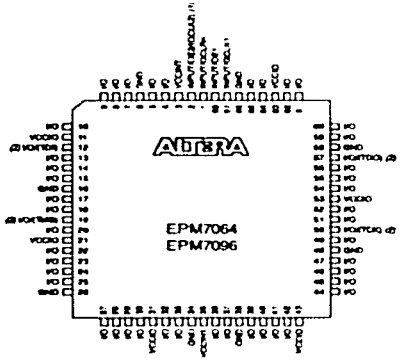
Package outlines not drawn to scale



Notes:
(1) The pin functions shown in parenthesis are only available in MAX 7000E and MAX 7000S devices.
(2) JTAG ports are available in MAX 7000S devices only

Figure 17. 68-Pin Package Pin-Out Diagram

Package outlines not drawn to scale.



68-Pin PLCC

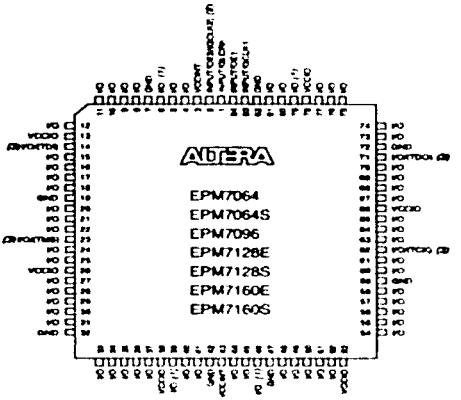
Notes:

(1) The pin functions shown in parenthesis are only available in MAX 7000E and MAX 7000S devices.

(2) JTAG ports are available in MAX 7000S devices only.

Figure 18. 84-Pin Package Pin-Out Diagram

Package outline not drawn to scale.

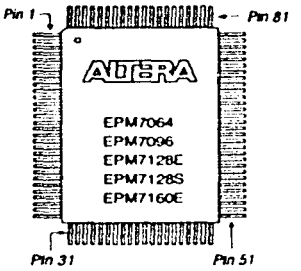


84-Pin PLCC

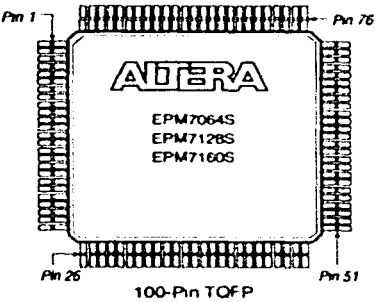
Notes:
(1) Pins 6, 39, 46, and 79 are no-connect (N.C.) pins on EPM7096, EPM7160E, and EPM7160S devices.
(2) The pin functions shown in parenthesis are only available in MAX 7000E and MAX 7000S devices.
(3) JTAG ports are available in MAX 7000S devices only.

Figure 19. 100-Pin Package Pin-Out Diagram

Package outline not drawn to scale.



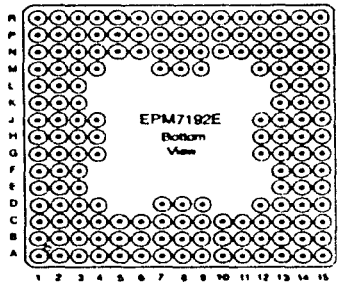
100-Pin PQFP



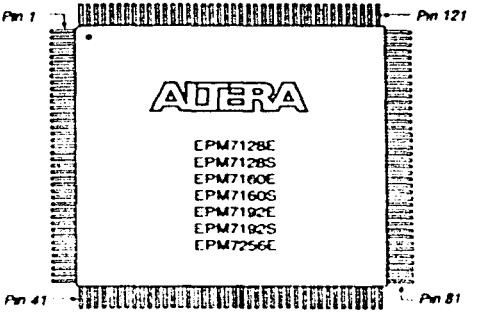
100-Pin TQFP

Figure 20. 160-Pin Package Pin-Out Diagram

Package outline not drawn to scale.



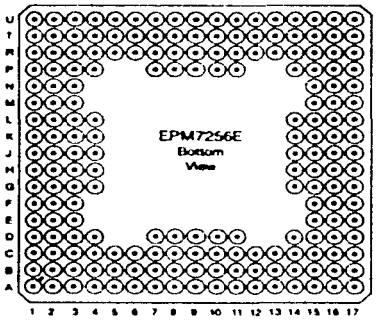
160-Pin PGA



160-Pin PQFP

Figure 21. 192-Pin Package Pin-Out Diagram

Package outline not drawn to scale



192-Pin PGA

Figure 22. 208-Pin Package Pin-Out Diagram

Package outline not drawn to scale

208-Pin PQFP/RQFP

Revision History

The information contained in the *MAX 7000 Programmable Logic Device Family Data Sheet* version 6.3 supersedes information published in previous versions. The following changes were made in the *MAX 3000A Programmable Logic Device Family Data Sheet* version 6.3: Updated the "Open-Drain Output Option (MAX 7000S Devices Only)" section on page 18.

101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
http://www.altera.com
Applications Hotline
(800) 800-EPLD
Customer Marketing
(408) 544-7104
Literature Services
lit_req@altera.com

APÉNDICE F

Glosario de Términos Astronómicos

Aberración Cromática.- Es el fenómeno óptico en el que se dan tantos focos como número de radiaciones monocromáticas haya, debido a la dispersión de la luz.

Aberración Esférica.- Es el fenómeno óptico que se da cuando los rayos procedentes de un punto del eje de un espejo forma con éste ángulos demasiado grandes y, por este motivo, no convergen en un punto, sino que envuelven una superficie.

Aberración en cabellera o coma - Es el fenómeno óptico que aparece cuando los rayos que parten de un punto exterior al eje óptico envuelven una superficie muy compleja, causada por una aberración esférica. Esto ocasiona que los rayos de luz proyecten una figura que recuerda a la cola de un cometa.

Apertura.- Es el diámetro del lente-objetivo del telescopio. A mayor diámetro, mejor el poder de captación luminosa del telescopio, por lo que capta a astros cuya magnitud sea más débil.

Ascensión Recta (AR).- Es la distancia angular entre el equinoccio de marzo y un punto particular sobre el ecuador celeste, cuya equivalente terrestre es la longitud.

Azimut.- Es la distancia angular de la proyección de un astro sobre el horizonte terrestre desde el polo norte celeste.

Bóveda Celeste.- Esfera envolvente imaginaria, inmóvil, proyectada hacia el infinito desde la Tierra, donde se encuentran "fijos" los astros celestes. En realidad, estos astros están en continuo movimiento pero, al estar a tan grandes distancias de la Tierra, aparecen como objetos fijos en el firmamento. Igualmente, se dice que esta esfera "gira" durante la noche aunque, en realidad, la que gira es la Tierra debido a su movimiento de rotación.

Cenit.- Es el punto de la bóveda celeste situado exactamente en la vertical del observador.

Declinación (DEC).- Es la distancia angular, ya sea norte o sur, entre el ecuador celeste y un punto particular en la bóveda celeste, cuya equivalente terrestre es la latitud.

Distancia Focal.- Es la distancia que existe entre el espejo del telescopio y su foco.

Distancia Focal Efectiva.- Cuando se manejan espejos cuyos radios de curvatura son grandes, no es necesario proceder con mucho cuidado para escoger el punto a lo largo del eje del espejo a partir del cual medir la distancia focal, bastando medir, para este caso, desde el punto en que el espejo intersecta con el eje. Sin embargo, en sistemas como el telescopio reflector tipo Gregoriano o del tipo Cassegrainiano, donde los radios de curvatura de los espejos son pronunciados, es necesario llevar un análisis óptico más complejo que arroja la distancia focal efectiva.

Eclíptica.- Es el círculo sobre la bóveda celeste que marca la trayectoria anual aparente del Sol. Sobre la proyección de este mismo círculo orbitan la mayoría de los planetas y la Luna, con excepción de Plutón y de Charonte.

Ecuador Celeste.- Es la proyección del ecuador terrestre en la esfera celeste. Encima de éste la declinación es positiva, mientras que debajo de él es negativa.

Equinoccio.- Cualquiera de las dos ocasiones durante el año en el que el Sol cruza el ecuador celeste (marzo 21 y septiembre 21).

Foco.- Es el punto en el que se reúnen los rayos procedentes de un punto objeto localizado en el infinito, para formar una imagen.

Frecuencia Sideral.- Es la frecuencia determinada por la duración de la rotación de un punto específico sobre la Tierra. La bóveda celeste girará, en su desplazamiento aparente, con respecto a ese punto específico, a esta frecuencia. Su unidad es el Hertz Sideral y varía, principalmente, según la latitud en el que se encuentre el observador.

Hora Sidérea.- La hora sidérea indica el punto de la bóveda celeste, ubicado por sus coordenadas de ascensión recta, que se encuentra en el cenit en una localización geográfica determinada, en un momento determinado.

Magnitud.- Es la medida del brillo aparente de un astro según una escala predeterminada. En esta escala, mientras más pequeño es el índice de magnitud, mayor será el brillo del astro, existiendo incluso magnitudes negativas. De esta manera, el planeta Venus tiene una magnitud de –4.5, mientras que la estrella polar tiene una magnitud de +2. El ser humano, a simple vista, puede observar astros de una magnitud de hasta +6 mientras que, los mayores telescopios del mundo, alcanzan a apreciar astros de magnitud de +26.

Movimiento de Carrera.- Es el mecanismo mediante el cual el ocular de un telescopio puede ser alejado o acercado al espejo objetivo para conseguir un correcto enfoque de las imágenes observadas. El movimiento de carrera se realiza, normalmente, con la ayuda de un tornillo que desplaza al ocular en la misma trayectoria que el haz de luz procedente de la imagen.

OAN.- Abreviatura de Observatorio Astronómico Nacional. Si un Observatorio ostenta este título, significa que es operado por el Instituto de Astronomía de la UNAM.

Polo Norte Celeste.- Es el punto imaginario del cielo en donde el eje de rotación de la Tierra, proyectado al infinito, tocaría la bóveda celeste.

Rotación.- Es el movimiento de la Tierra sobre su propio eje. Este movimiento ocasiona el movimiento aparente de la bóveda celeste durante la noche.

Traslación.- Es el movimiento de la Tierra en su órbita alrededor del Sol. Este movimiento ocasiona que, durante este recorrido, se pueda apreciar desde la Tierra distintos fragmentos de la bóveda celeste.