UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA DE INGENIERIA ELECTRICA

TRABAJO DE GRADUACION:

"Diseño y Construcción de Interface Programador de Arreglos Lógicos Programables (PAL´s) para computadoras IBM o Compatibles"

PRESENTADO POR:

HECTOR ENRIQUE OJEDA

PARA OPTAR AL TUTULO DE:

INGENIERO ELECTRICISTA

CIUDAD UNIVERSITARIA, MARZO DE 1998

UNIVERSIDAD DE EL SALVADOR

RECTOR:

DR. BENJAMIN LÓPEZ GUILLEN

SECRETARIO GENERAL:

LIC.ENNIO LUNA

FACULTAD DE INGENIERIA Y ARQUITECTURA

DECANO:

ING. JOAQUIN ALBERTO VANEGAS AGUILAR

SECRETARIO:

ING. JOSÉ RIGOBERTO MURILLO CAMPOS

ESCUELA DE INGENIERIA ELÉCTRICA

DIRECTOR:

ING. JOSÉ ROBERTO RAMOS LÓPEZ

UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA DE INGENIERIA ELECTRICA

Trabajo de graduación previa a la opción al grado de:
INGENIERO ELECTRICISTA

Titulo : **"Diseño y Construcción de Interface Programador de Arreglos Lógicos Programables (PAL's) para computadoras IBM o Compatibles"**

Presentado por:

Héctor Enrique Ojeda

Trabajo de Graduación aprobado por:

Coordinador :

Ing, Ricardo Ernesto Cortéz

Asesores:

Ing. Javier Elías Guillén Hénriquez
Ing. Julio Quijano

San Salvado, Marzo de 1998

Trabajo de Graduación aprobado por:

Héctor Enrique Ojeda

Coordinador y Asesor:

ESCUELA DE INGENIERIA ELECTRICA
FACULTAD DE INGENIERIA
Y ARQUITECTURA
Universidad de El Salvador

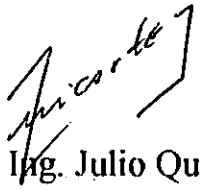
Ing, Ricardo Ernesto Cortéz

Asesor:

REGISTRO NACIONAL DE ARQUITECTOS E INGENIEROS
JAVIER ELIAS GUILLEN HENRIQUEZ
INGENIERO ELECTRICISTA-REGISTRO No. IE 0248
MOP-VMVDU EL SALVADOR, C. A.

Ing. Javier Elías Guillén Henríquez

Asesor:



Ing. Julio Quijano

ACTA DE CONSTANCIA DE NOTA Y DEFENSA FINAL

En esta fecha, 16 de marzo de 1998 en el local de la Sala de Lectura de la Escuela de Ingeniería Eléctrica, a las diecisiete horas, en presencia de las siguientes autoridades de la Escuela de Ingeniería Eléctrica de la Universidad de El Salvador:

Firma

1- Ing. José Roberto Ramos López
Director
2- Ing. Gerardo Marvin Jorge Hernández
Secretario

ESCUELA DE INGENIERIA ELECTRICA
FACULTAD DE INGENIERIA
Y ARQUITECTURA
Universidad de El Salvador

Y con el Honorable Jurado de evaluación integrado por las personas siguientes:

Firma

1- Ing. César González

2- Ing. Eduin Ruyé Mendoza Maldonado

Se efectuó la defensa final reglamentaria del Trabajo de Graduación:

"Diseño y Construcción de Interface Programador de Arreglos Lógicos Programables (PAL's) para Computadoras IBM o Compatibles"

A cargo del Bachiller:

OJEDA HECTOR ENRIQUE

Habiendo obtenido el presente trabajo una nota final, global de 8.9
(Ocho Punto Nueve.)

AGRADECIMIENTOS

-Mis amigos Luis Escobar, Hugo Colato, y demás que me alentarón a seguir adelante en este proyecto.

-Ing, Ricado Ernesto Cortéz, por su cooperación y las oportunas observaciones en el desarrollo de este trabajo

- Ing. Javier Elías Guillén Hénriquez, por demostrar su amistad, ayuda incondicional en el desarrollo de este proyecto.

Ing. Julio Quijano, por su cooperación y disposición incondicional mostrada en el desarrollo de este proyecto.

DEDICATORIA

MI FAMILIA: Por su compañía en todo el proceso de formación hasta coronar mi carrera.

MI FUTURA ESPOSA : Maura por su comprensión y cooperación para mi carrera.

MI HIJO : Marlón Alexander, por estar conmigo.

PREFACIO

Actualmente en la Escuela de Ingeniería Eléctrica en aplicaciones de circuitos combinacionales y/o secuenciales que se desarrollan en las materias de electrónica digital han llegado a utilizar solamente componentes integrados de productos estándar las series 74xxxx ó 54xxxx, componentes discretos, con combinación de otros tipos de integrados para diseñar Interfaces particulares, produciendo diseños de interfaces de gran tamaño.

En este trabajo que consiste en diseñar e implementar un sistema, que tenga la capacidad de realizar la programación de integrados de arquitectura programable, para sustituir todos los integrados de productos estándar que se utilicen en diseños digitales.

Los integrados que en este trabajo se podrá realizar la programación de su arquitectura son los Arreglos lógicos Programables PAL's, y con ello se inicia una nueva alternativa de implementar circuitos digitales, con costos menores.

Los integrados a programar con el sistema implementado en este trabajo son los fabricados por Texas Instruments, y Advanced Micro Device (AMD), los primeros con las familias de 24 y 20 pines, mientras que los segundos solamente con la familia de 20 pines.

PREFACIO

Actualmente en la Escuela de Ingeniería Eléctrica en aplicaciones de circuitos combinacionales y/o secuenciales que se desarrollan en las materias de electrónica digital han llegado a utilizar solamente componentes integrados de productos estándar las series 74xxxx ó 54xxxx, componentes discretos, con combinación de otros tipos de integrados para diseñar Interfaces particulares, produciendo diseños de interfaces de gran tamaño.

En este trabajo que consiste en diseñar e implementar un sistema, que tenga la capacidad de realizar la programación de integrados de arquitectura programable, para sustituir todos los integrados de productos estándar que se utilicen en diseños digitales.

Los integrados que en este trabajo se podrá realizar la programación de su arquitectura son los Arreglos lógicos Programables PAL's, y con ello se inicia una nueva alternativa de implementar circuitos digitales, con costos menores.

Los integrados a programar con el sistema implementado en este trabajo son los fabricados por Texas Instruments, y Advanced Micro Device (AMD), los primeros con las familias de 24 y 20 pines, mientras que los segundos solamente con la familia de 20 pines.

RESUMEN DEL TRABAJO

El trabajo desarrollado consistió en diseñar e implementar un sistema que permita realizar la programación de la arquitectura de los Arreglos Lógicos Programables PAL's, para lo cual se fracciono en los siguientes tres capítulos.

En el capitulo I se realiza una documentación de los Arreglos de Lógica Programables PAL's de manera generalizada. Para comprenderlos y así comprender su arquitectura interna y también las curvas de programación necesarias para programarlos.

En el capitulo II se realiza el diseño e implementación del hardware necesario para realizar la programación, haciendo en dos etapas : hardware de control y hardware de fuentes conmutables.

En el capitulo III se diseña e implementa el software necesario (drivers) que manejara el hardware para realizar la programación de fusibles de la arquitectura interna de los PAL's.

TABLA DE CONTENIDOS

Capítulo **Página**

CAPITULO I

DOCUMENTACIÒN DE LAS ALTERNATIVAS PARA DISEÑOS E IMPLEMENTACIÒN DE CIRCUITOS DE LÒGICA DIGITAL.

Introducción.

En éste capítulo se ha puesto especial enfásis en la documentación y conceptualización de la clasificación en las distintas alternativas de los IC("Integrated Circuit"), que se encuentran disponibles en el mercado, para diseñar y/o implementar circuitos de lógica digital.

La documentación de las distintas alternativas de IC se realiza a nivel general para plantear la ubicación de cada una de ellas, en las familias lógicas, y optar por cualquiera de ellas, a ser aplicada en un diseño. Esta clasificación se lleva a cabo en base a la arquitectura interna de los dispositivos, que conforman cada alternativa.

Entre estas alternativas, se encuentra la de lógica programable de fusibles, que es la que nos ocupa en esté trabajo de graduación. Ya que se construye un **programador de Lógica de Arreglo Programable ("Logic Array Programmable PAL"),** por lo tanto, se realiza la documentación de la lógica programables de fusibles de forma concreta a tal nivel que se discuten las curvas de programación de los PAL, similarmente la arquitectura interna de ellos.

1.1.0 Objetivos

1.-Presentar y documentar las Familias Lógicas Programables, y sus diferentes alternativas para realizar implementación de circuitos Digitales.

2.- Presentar y documentar los Arreglos Lógicos Programable PAL's, de los fabricantes Texas Instruments, Advenced Micro Device AMD. Con sus respectivas gráficas de tiempo de programación

1.1.1 Fundamentos de las Familias Lógicas Programables

En los dispositivos productos estándar, la arquitectura es definida por el fabricante de los IC, y no puede ser en ningún momento alterada por el usuario, Ejemplos son la serie 54XXX,74XXX,etc

En la actualidad los diseñadores de circuitos electrónicos pueden escoger entre una amplia variedad de alternativas de IC, para construir sistemas digitales. Esta variedad ha venido desarrollándose aceleradamente desde la invención de los Ics, en la década de los años sesenta, y más concretamente en 1970°, cuando Monolithib Memories, (ahora parte de Advanced Micro Divaces "AMD"), fue el primer pionero como fabricante en introducir los dispositivos de Lógica de Arreglos Programables ("PLA's"), y posteriormente en 1978 AMD es primero en fabricar los PAL en grandes volúmenes para el mercado de mandado por los usuarios. Esta variedad de IC puede clasificarse en tres categorías básicas los cuales se presentan a continuación.(ver figura 1.1)

1.- Productos estándar ("StandardProducts").
2.- Dispositivos de Lógica Personalizada ("Custom Logic")
3.- Lógica de Fusible Programable ("Programmable Logic")

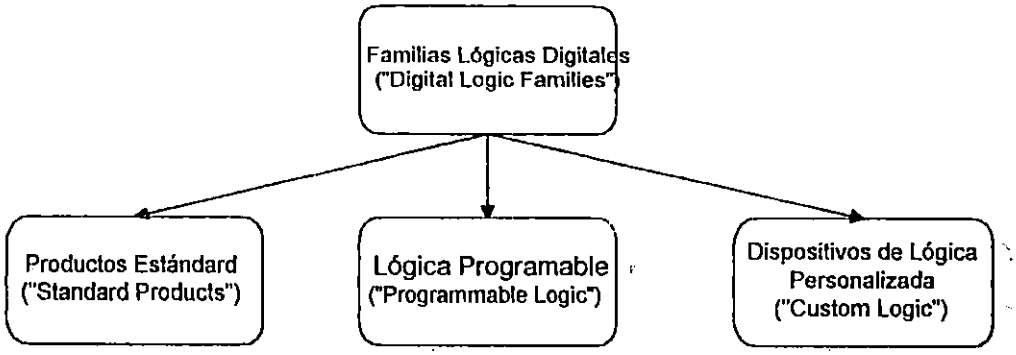


Fig.1Categorias básicas de lógica digital

1.1 Productos Estándar ("Standard Productcs").

En los dispositivos productos estándar, la arquitectura es definida por el fabricante de los IC, y no puede ser en ningún momento alterada por el usuario, Ejemplos son la serie 54XXX,74XXX,etc. sobresaliendo las siguientes características de los productos estándar que a continuación se presentan.

1. Por parte del usuario se necesita poco conocimiento para el uso en aplicaciones de los IC.
2. Bajo costo por cada IC individualmente.
3. Producción de los IC en grandes volúmenes.
4. Arquitectura fija por el fabricante no modificable por el usuario.

5. No optimiza, las funciones lógicas son grandes y no se pueden simplificar más de lo permiten las herramientas de diseño.
6. Altos costos económicos por sistemas.(Gran número de IC por sistema).

El nivel de conocimiento necesario para el uso de los IC producto estándar en aplicaciones es mínimo en comparación con el nivel de conocimiento requerido en las aplicaciones desarrolladas con circuitos de lógica personalizada o lógica programable.

La producción de IC de productos estándar es realizada en grandes volúmenes, el costo de producción de IC es reducido. Y así es bajo el precio por unidad de los productos estándar. Igual efecto ocasiona la producción de los mismos productos por la competencia de productos estándar.

La arquitectura interna de los productos estándar se presenta estática con respecto a las modificaciones futuras de fondo en el tiempo o es mínima, comparado con las modificaciones que se producen en la arquitectura de lógica personalizadas o lógica programable.

La responsabilidad del fabricante de IC es total en los siguientes campos.
El diseño de la arquitectura interna de los IC productos estándar.
Pruebas de los IC productos estándar.
Forma de los IC productos estándar.

Por que, el circuito integrado es fabricado sobre una larga escala de integración **("Large Scale Integration","LSI")**, el proceso es eficiente.

La utilización de los Ics en los equipos electrónicos digitales ha ido en aumento, inicialmente se utilizaron los circuitos integrados de **función lógica fija, ("off the Shelf")**, realizados en pequeña escala de integración ("**Small Scale Integration** SSI") y media escala de integración ("**Medium Scale Integration** MSI"). Ejemplos de estos son las series 74XXX, 54XXX para TTL, la 40XXX de CMOS y los microprocesadores, microprogamables construidos con bloques de dispositivos de gran escala de integración **("Large Scale Integration** LSI") y dispositivos de lógica Transistor Transistor ("TTL") y dispositivos ("CMOS") SSI/MSI.

1.2 Dispositivos de Lógica Personalizada ("Custom Logic Devices")

La lógica personalizada, se encuentra disponible predominantemente en forma de arreglos de compuerta. Ofreciendo al diseñador importante ventajas sobre los productos estándar. Comparados en las implementaciones SSI\MSI. La lógica personalizada requiere que el diseñador especifique el chip a ser diseñado, desde los garabatos de inicio hasta las funciones necesitadas a realizar. La intención es proveer la solución dada por el diseñador, que es la necesaria para la aplicación en cuestión, ni más ni menos.

Como la lógica personalizada permite al diseñador realizar su propia arquitectura exactamente, cualquier diseño de lógica personalizada esta libre de realizar soluciones innovadoras a las aplicaciones del problema, lo que le significa competitivas ventajas a estos productos.
Para los dispositivos de lógica personalizada se pueden hacer referencia a las siguientes principales desventajas las cuales son:

1. Alto conocimiento del IC por parte del usuario para su uso.

2. Alto costo por unidad individual de dispositivos.
3. Escasez de apoyo en alto nivel en la forma de: Software, desarrollos de sistemas, notas de aplicaciones o libros para diseños de lógica personalizados, por que cada dispositivo es diferente.
4. Bajo densidad comparativa a estándar SLI .

El tiempo de trabajo invertido en un diseño con dispositivos de matrices de compuertas puede incrementar significativamente el costo de un sistema diseñado. Pero los dispositivos no sólo deben ser diseñados, sino que debe de ser depurado antes de la puesta en producción. En la producción de dispositivos de lógica personalizada tenemos dispositivos tales como las Matrices de Compuertas(**"Gate Arraays"**), y Celdas Estándar **("Standard Cells").**

Los diseñadores deberían documentar el diseño completamente para lograr utilizar el dispositivo correctamente.

1.3 Dispositivos de Lógica Programable ("Programmable Logic Devices")

La Lógica Programable combina las ventajas de la arquitectura flexible de los dispositivos personalizados con el **"Off The Shelf"** de los productos estándar, está combinación conlleva a la reducción de, el tiempo necesario para realizar un diseño implementado con esté tipo de lógica, y el costo del dispositivo. La lógica programable presenta en la implementación de un diseño un ciclo de tiempo menor, que el requerido en lógica personalizado que puede llegar a ser desde meses hasta años. Un elemento de lógica programable puede ser definido por programación de fusibles. El proceso toma solo unos pocos segundos, esto representa una alternativa revolucionaria para ser aprovechada por los diseñadores de sistemas de circuitos electrónicos, con esta alternativa el ingeniero diseñador puede censar las salidas con entradas en la nueva arquitectura y evaluar rápidamente, si no le parece una nueva idea puede ser definida, programada y estar lista para la evaluación en horas.

La velocidad con la cuál un nuevo diseño puede ser accesado, explorado y evaluado se presenta como una nueva innovación de diseño para los diseñadores.

El nivel de conocimiento, y tiempo necesario para diseñar, medir, depurar y poner en producción un dispositivo de lógica programable, es mucho mayor que el empleado en los productos estándar, pero sustancialmente menos que los dispositivos personalizados. Los productos de lógica programable proveen de las herramientas de Software necesarias para disminuir los gastos considerablemente, esto permite a los diseñadores definir sus diseños en términos de las funciones lógicas, especificación formal de las entradas, y de salidas.

Uno de los problemas que puede enfrentar un diseñador es la necesidad de utilizar gran cantidad de circuitos integrados, con varias tarjetas de circuito impreso de tamaño considerable, y en consecuencia, muchas conexiones. El hecho de disminuir el número de integrados empleados ofrece interesantes ventajas

A continuación se presentan un esquema general(ver fig. 1.2) de las diferentes alternativas de los dispositivos de lógica de fusibles programable o conocidos como

Dispositivos de Lógica Programable (**"Programmable Logic Devices, PLD's"**), disponibles en el mercado para implementación de circuitos electrónicos.

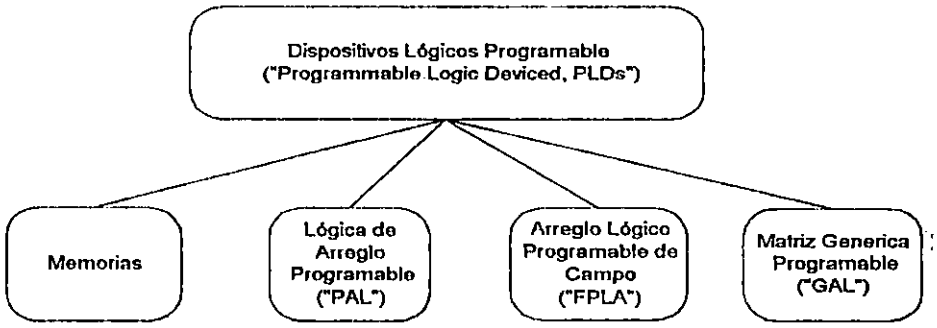


fig1.2 Clasificación de Dispositivos Lógicos Programable PLD's

1. Memorias
2. FPLA: Field Programmable Array Logic.
3. PAL: Programmable Array Logic.
4. GAL Generic Array Logic

La mayoría de los dispositivos listados arriba corresponden a arreglos programables de compuertas AND, OR o ambos, en donde lo que hace es eliminar fusibles en los arreglos a programar. Excepto el GAL que contiene celdas programables. Su forma de acceso es aleatoria y son no volátiles, en lo que respecta a la permanencia de información.

En general, un PLD puede sustituir, en algunas aplicaciones, desde unos pocos hasta decenas de integrados de función fija.

Muchos de los dispositivos de lógica programable pueden incluir los siguientes componentes como parte de su arquitectura: Flip Flops, Registros Básicos, Registros de entrada y salida en donde el modo de operación de estos dispositivos es programable. Todos estos dispositivos de lógica programable por fusibles permiten en la medida de su arquitectura interna implementar el diseño de tipo lógico secuencial o lógico combinacional.

1.4 Dispositivos Lógicos Programables "PLD's"

La documentación siguiente es en primer lugar presentar los conceptos básicos de la tecnología de Lógica Programable. El término de Dispositivos Lógico Programable **("Programmable Logic Device PLDs")**, se refiere a un dispositivo cuya arquitectura interna presenta un arreglo lógico, en el cual el diseñador de circuitos electrónicos, programará las funciones para la aplicación especifica. La estructura básica de los Dispositivos Lógicos Programable en general se puede representar , a nivel, de bloques tal como se muestra en la fig. 1.3 y la función lógica de salida estará como suma de

productos, la que es fácil de generar con los mapas de Karnaugh y los teoremas de D'morgan.

Fig.1.3 Estructura Básica de un PLD's de Arquitectura Programable

1.4.1 Ventajas de la lógica programable.

Los PLDs ofrecen muchas ventajas para los diseñadores de sistemas con circuitos electrónicos, las siguientes son ventajas atribuibles a la lógica programable:

- Se reduce el tamaño de la tarjeta del circuito impreso, debido a que se ocupa menos espacio.
- Se tiene menor demanda de potencia.
- El proceso de montaje es más rápido y menos costoso.
- El diseño presenta mayor confiabilidad, debido a que tienen menos conexiones externas o puentes, entre los integrados.
- Se aumenta la velocidad, debido a que el retardo de tiempo entre los integrados se reduce.
- Los procedimientos para la detección de fallas son más sencillos.
- Se reduce la cantidad de encapsulados ("Packages") cuando las funciones MSI/SSI son reemplazadas por PLDs.
- El ciclo de diseño es corto, cuando se compara con los productos personalizados.
- Protección del diseño(fusible de protección o seguridad): Circuitos que pueden ser protegidos de copias indebidas por la quema del fusible de seguridad.

1.4.2 Simbología para los PLD's

Los PLDs deben de estar enmarcados bajo el concepto revolucionario para realizar soluciones cortas en circuitos de lógica TTL discreta. En general, un PLD como se mencionó es un circuito que puede ser programado por el usuario para realizar funciones lógicas.

Para comprender y usar la arquitectura de los PLDs, ha adoptarse una convención para explicar la forma de representar cuando un fusible esta quemado o no. (Ver fig.1.4). Esta figura es la representación de una compuerta AND de tres entradas. Note que una sola línea es mostrada como entrada de la compuerta AND. Esta línea es comúnmente referida a todas las líneas productos que se encuentran formando la arquitectura interna del PLD. Las entradas son mostradas como líneas verticales, y la intersección de esas líneas son los fusibles a programar. **Una "X" representa un fusible intacto,** y forme

parte del termino producto; un punto ". " Representa una conexión fija, del término producto. Y **una no "X" representa un fusible quemado** Esto hace que la entrada se aparte del termino producto. La representación de ningún punto "." O " **X**" representa un fusible fundido y que no existe conexión eléctrica alguna entre el término producto y las entradas del arreglo.

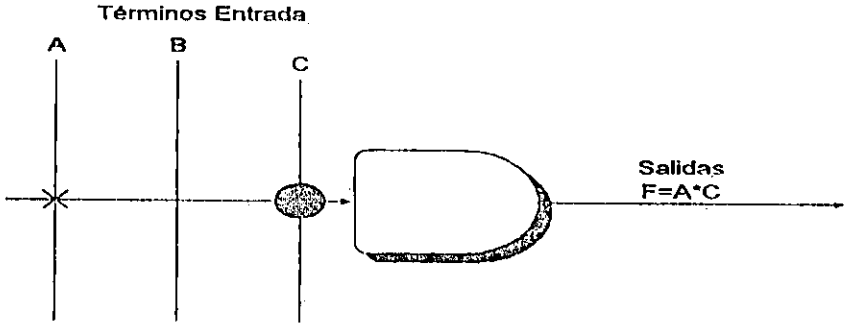


fig 1.4 Simbología básica de la arquitectura de fusibles programables

Esta simbología podría extenderla a un desarrollo simple de 2-entradas al arreglo AND programable que alimente a una compuerta OR. (Fig1.5). Note que los buffer están alimentando los valores de verdad y complemento de las variables de entrada. La intersección de líneas de términos de entrada forma un arreglo de 4-entradas y 3-compuertas AND programables y la función de salida en la compuerta OR será F=A(B)'+B(A)'. De acuerdo a lo expresado en la fig.1.4.

Concluyendo, los PLD, son un circuito integrado que contiene un número muy grande de compuertas básicas (AND, OR, NOT y/o EXOR), y en muchos casos flip-flops o registros que están conectados entre sí, dentro del integrado. Dicho circuito es programable, es decir, la función específica a desarrollar viene determinada por la ruptura selectiva de conexiones tipo fusible que contiene, y al mismo tiempo, dejando otras conexiones intactas.

En otras palabras, son dispositivos de lógica de fusibles programables construidos en bloques, capaces de permitir la generación de funciones complejas y de alto rendimiento, las cuales combinan la flexibilidad de la arquitectura proveniente de un diseño propio, con la disponibilidad instantánea, junto con la creación de arreglos a bajo costo y que no estén en existencia en el mercado.

Estos dispositivos programables tienen un fusible de seguridad que al quemarlo, impiden la lectura de lo programado, protegiendo así el diseño que se hizo. Este proceso de programación (quema de fusibles) puede ser realizado por el fabricante con instrucciones del cliente (diseñador), o por el mismo cliente (cuando este posee o tiene disponible el equipo de programación del dispositivo de lógica programable).

1.4.3 Memorias.

Las memorias son dispositivos de almacenamiento de datos binarios de largo plazo a corto plazo. Los principales tipos de memorias son: Semiconductoras, Magnéticas, y Opticas. Las memorias semiconductoras están formadas por matrices de

elementos de almacenamiento que pueden ser latches, condensadores o cualquier otro elemento de almacenamiento de carga. Y las dos memorias semiconductora son las Memorias de acceso aleatorio("Ramdom Access Memory RAM"), Memoria Sólo de lectura ("Read Only Memory ROM") que es un tipo de memoria donde se almacenan datos en forma permanente.

Las Familias de las memorias se fabrican con tecnología bipolar(TTL), o con tecnología MOS (metal-óxido de metal)(ver fig. 1.6). Se presenta la clasificación de las memorias ROM. La ROM de máscara es un tipo de memoria en la que los datos se almacenan permanentemente en la memoria durante el proceso de fabricación. La PROM, o ROM programable, es aquel tipo de ROM en la que el usuario, con ayuda de equipo especializados, almacena eléctricamente los datos. Es decir tanto la ROM con mascara como la PROM pueden ser de cualquier tecnología. La EPROM, o memoria borrable (erasable PROM) es exclusivamente un dispositivo MOS.

La UV EPROM puede ser programada eléctricamente por el usuario, pero los datos almacenados deben borrarse mediante la exposición a la luz ultravioleta durante un periodo de varios minutos. La PROM borrable eléctricamente (EEPROM Electrically Erasable) se puede borrar y programar mediante pulsos eléctricos, ya que se pueden reprogramar dentro del propio circuito final, lo que permite reconfigurar cualquier sistema. El proceso sólo dura unos pocos milisegundos.

Las memorias FLASH son memorias de lectura/escritura de alta densidad(alta densidad se refiere a gran capacidad de almacenamiento de bits) que no son volátiles, lo que significa que los datos se pueden almacenar indefinidamente sin necesidad de alimentación. Alta densidad significa que se pueden empaquetar en una pequeña superficie del chip gran cantidad de celdas; es decir, en cuanto mayor sea la densidad mayor cantidad de bits se pueden almacenar en un chip de tamaño determinado.

La arquitectura interna de las memorias se presentan de manera general en la fig. 1.6a y la fig.1.6b. Las memorias ópticas y magnéticas no se contemplan en este trabajo por no estar dentro del conjunto de los PLDs.

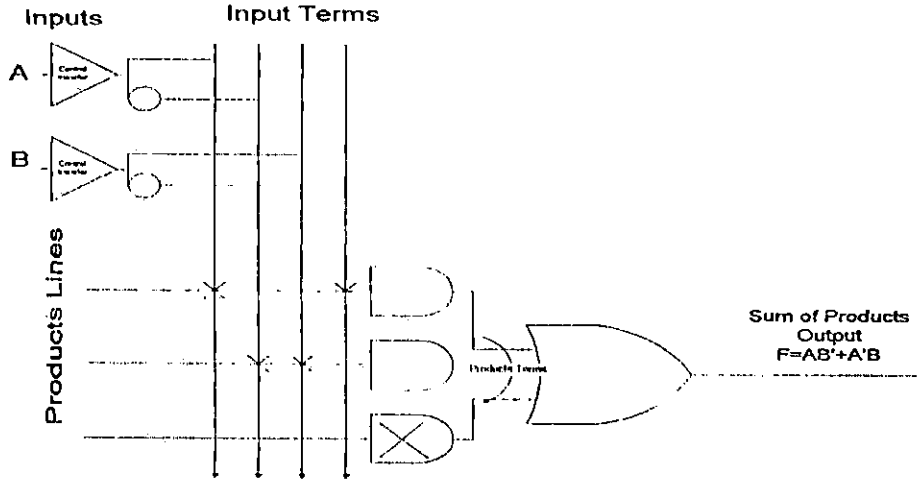


Fig 1.5 simbología básica de los PLD's

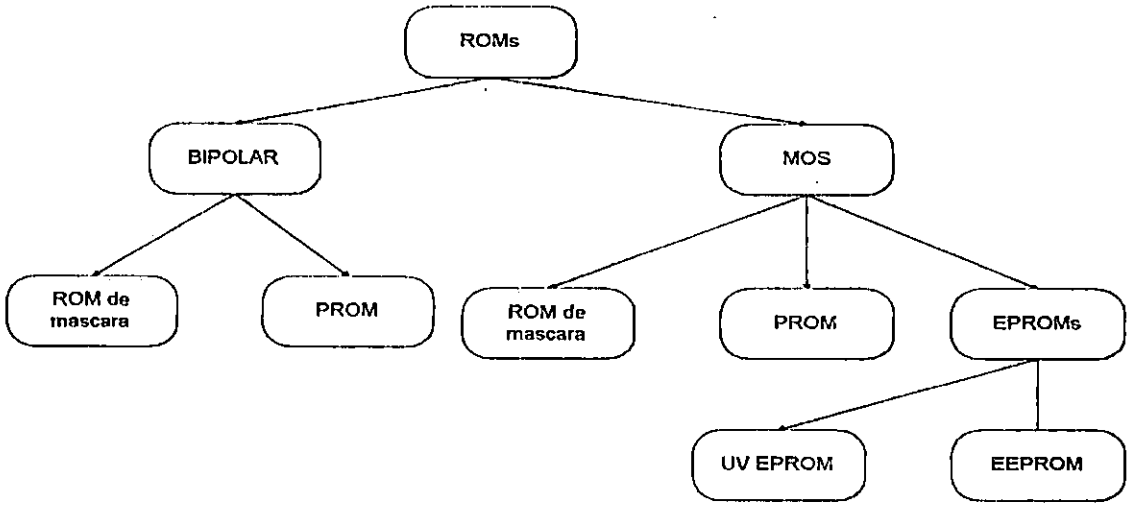


fig1 6 Familia de las memorias ROM semiconductoras

fig1.6a Memoria se sólo lectura PROM's

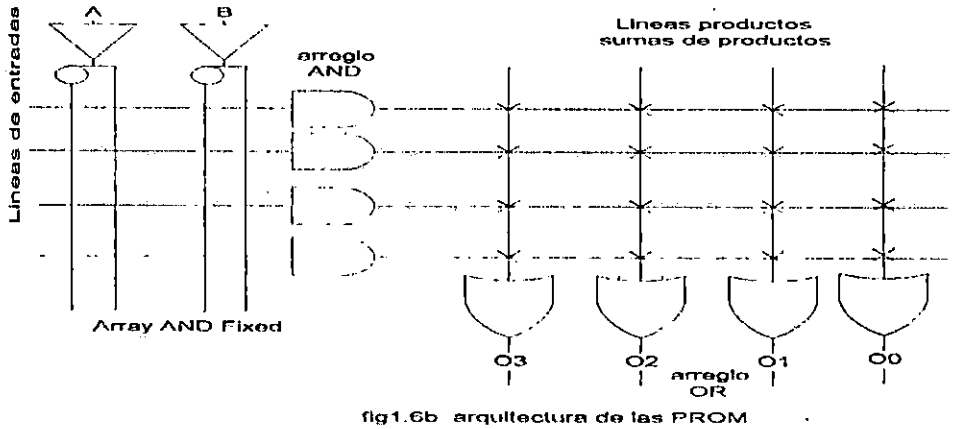


fig1.6b arquitectura de las PROM

1.4.4 Arreglo Lógico Programable de Campo ("Field Programmable Array Logic, FPLA")

Un arreglo FPLA, es un dispositivo de lógica de fusibles programables, en donde tanto el arreglo AND como el arreglo OR, son programables (fig. 1.7). Es decir que son programables las conexiones de las líneas de entradas hacia el arreglo AND, y las líneas de producto AND hacia las entradas de la compuerta OR. Estas características los convierten en uno de los más versátiles PLDs. Sin embargo, presenta la desventaja de que por tener dos conjuntos de conexiones tipo fusible, es más difícil de programar y verificar, que una PROM o PAL.

fig1.7a Arreglo de Lógica Programable

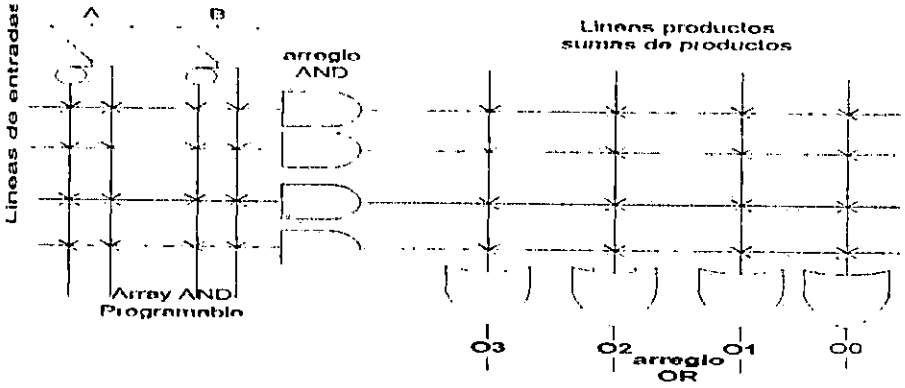


fig1.7b arquitectura de las FPLA

1.4.5 Lógica de Arreglo Programable ("Programmable Array Logic PAL")

La estructura básica de un PAL es exactamente el opuesto de una PROM; el arreglo AND es el programable y el arreglo OR es fijo, en las PLAs se dispone de una cantidad de términos productos mayor que en las PROM, (la arquitectura interna de las PROM presenta el número de "n" entradas es igual al número de 2 a la "n" compuertas AND en el arreglo interno), así con el PALs un diseñador de circuitos electrónicos, dispone de los términos productos y entradas necesarias para desarrollar su aplicación.

El PAL's es dispositivo de lógica de fusibles programables, construidos en bloques, capaces de almacenar funciones complejas de alto rendimiento, las cuales combinan la flexibilidad de la arquitectura proveniente de un diseño propio con la disponibilidad instantánea y a bajo costo de los productos estándar.

Un PAL es un PLD en el que las conexiones de las líneas de entradas hacia el arreglo AND son programables (conexiones tipo fusible a quemar), mientras que las conexiones de las líneas de producto AND hacia las entradas de las compuertas OR son fijas(alambres). Las funciones generadas se obtienen como sumas de productos estándar(MINTERMOS).

El factor por el cual el arreglo AND es el programable, por que la arquitectura interna del PALs ya viene diseña por los fabricantes que los fusibles a romper se encuentran disponibles en el arreglo AND. Consecutivamente mayor cantidad de términos productos, permitiendo desarrollar e implementar complejas aplicaciones por diseñadores experimentados.

La limitación primordial del PALs consiste en: **El número de compuertas AND requeridas por un diseñador de circuitos electrónicos para generar una función lógica, en una aplicación especifica, no puede exceder las disponibles por la arquitectura interna del PALs.**

Los dispositivos PALs contienen muchos dispositivos adicionales desde la fabricación tales como: registros, buffer de tres estados, multiplexores, entre otros. Los dispositivos PAL tienen muchas características en común con las memorias programables de sólo lectura (PROM) y los arreglos lógicos programables (PLA). Los tres comparten la estructura básica interna AND-OR, pero se diferencian en la programabilidad.

En la figura 1.8 se muestra la estructura básica de los dispositivos programables .

La estructura tiene dos niveles; el primero es el arreglo AND el cual recibe las entradas, ejecuta las funciones AND deseadas sobre las entradas, y aplica estas funciones al segundo nivel, constituido por un arreglo OR; este combina varias funciones AND, produciendo las salidas deseadas. La estructura antes descrita hace a los dispositivos programables ideales para el desarrollo de la lógica Booleana, en forma de sumas de productos, las cuales son fácilmente generadas por técnicas de diseño digital como el mapa de Karnaugh. Y se puede implementar cualquier diseño combinacional o secuencial (ver fig.1.9), esquema de diseño secuencial en el ámbito de bloques.

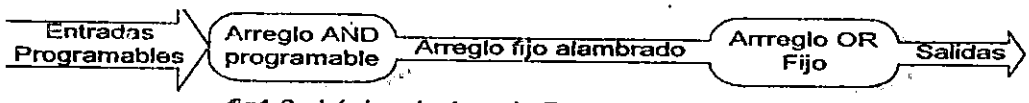


fig1.8a Lógica de Arreglo Programable(PAL)

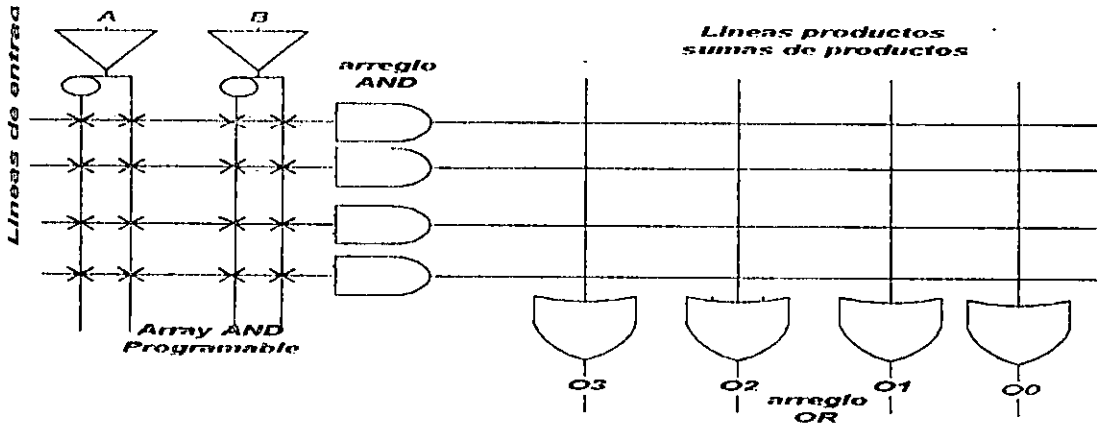


fig1.8b arquitectura de las PAL

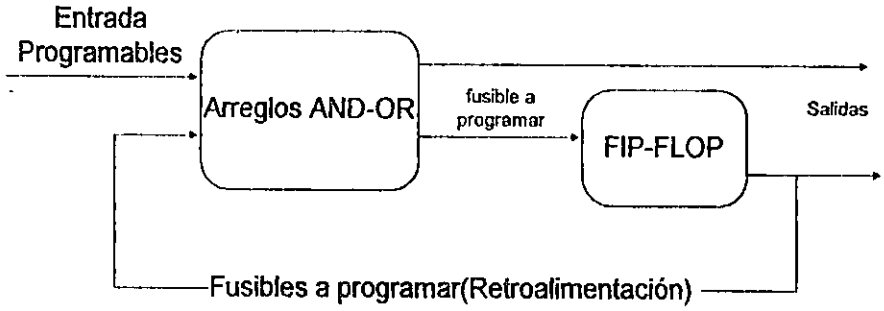


Fig. 1.9Dispositivos de Lógica Programable Secuencial

1.4.6 Arreglo Lógico Generalizado (" Generic Logic Array GAL")

El desarrollo más reciente de los PLDs es el GAL. Al igual que el PAL, se forma con una matriz AND programable y una matriz OR fija, con una salida lógica programable. Las dos principales diferencias entre los dispositivos GAL y PAL son:

1. GAL es reprogramable
2. GAL tiene configuraciones de salida programables

El GAL se puede reprogramar una y otra vez, por que se usa la tecnología CMOS Borrables Eléctricamente ("Electrically Erasable EECMOS") celdas programables y reprogramables, en lugar de tecnología bipolar y fusibles.

En la fig. 1.9 se ilustra la estructura básica de una GAL con dos variables de entrada y una salida, aunque la mayoría de las GALs se pueden tener muchas entradas y muchas salidas. La matriz reprogramable es esencialmente una red de conductores ordenados en filas y columnas, con una celda CMOS eléctricamente borrable (EECMOS) en cada punto de intersección, en lugar de un fusible como en el caso del PALs. En la fig. 1.9, estas celdas se indican como bloques.

Cada fila está conectada a la entrada de una puerta AND, y cada columna a una variable de entrada o a su complemento. Mediante la programación se activa o desactiva cada celda EECMOS, y se puede aplicar cualquier combinación de variables de entrada, o sus complementos, a una puerta AND para generar cualquier operación producto que se desee. Una celda activa conecta de forma efectiva fila y columna, y una celda desactivada desconecta la fila y la columna. Las celdas se pueden borrar y reprogramar eléctricamente. Una celda EECMOS típica puede mantener el estado en que se ha programado durante 20 años o más.

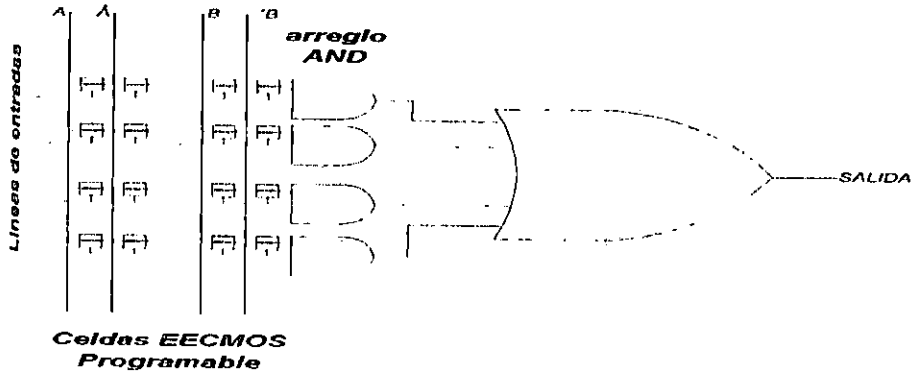


fig1.9 arquitectura de las GAL

Los dispositivos de lógica programable recientemente se han ido sustituyendo casi completamente por los dispositivos de **Lógica de Arreglo Generalizado("Generalized Array Logic, GAL")**, que son capaces de reemplazar una **"suma de productos"** Booleanos. En una celda de salida se puede realizar la programación como Combinacional o registrada, dándoles una enorme versatilidad. Para el uso de -- diseñadores de circuitos electrónicos la aplicación de los PLDs no hacen obsoleta la teoría de lógica digital en general, sino al contrario para programar un GAL es necesario conocer cómo funcionan los chips digitales internamente(arquitectura interna de los GALs), y así hacer una adecuada selección de las compuertas a usar, cuidar la manera de excitar cada Flip-Flop (tipo D), que se encuentra en la celda de salida, eliminar Glitches y así poder aprovechar al máximo el recurso del dispositivo GAL.

Para realizar una recapitulación de lo expuesto hasta ahora sobre el universo de los PLD's se presenta el siguiente cuadro Sinóptico de los dispositivos de lógica programable (ver Tabla S.C1.1)

1.5 Tipos de PAL's estándar para los fabricantes AMD("Advanced Micro Device") y TI ("Texas Instrumemts")

Los dispositivos PALs, como ya se documentan ellos contiene en su arquitectura interna circuitos lógicos, los que se pueden configurar como entrada o salida combinacionales, o como salida secuenciales. En este modo secuencial la salida proviene

de un Flip flop. Y para ilustrar mejor los tipos* de salidas que se pueden obtener en el PALs en estudio se presenta abajo un detalle de la clasificación de salida. (ver fig. 1.10)

PROM	EPROM	EEPROM	PAL	FPLA	GAL
Dispositivo de lógica de fusible programable Tecnología MOS y bipolar.	Dispositivo de Lógica de fusible programable tecnología MOS.	Dispositivo de lógica de fusible programable Tecnología MOS.	Dispositivo de lógica de fusible programable.	Dispositivos de lógica de fusibles programables.	Dispositivos de lógica de celdas tipo EECMOS programables
Programado en el proceso de fabricación.	Puede ser programado Por el usuario.	Puede ser programado por el usuario	Puede ser programado por el usuario	Puede ser programado por el usuario	Puede ser programado por el usuario
Arreglo AND fijo Arreglo OR programable	Arreglo AND fijo arreglo OR programable	Arreglo AND fijo arreglo OR programable	Arreglo AND programable y arreglo OR fijo	Arreglo AND-OR programable	Celdas programables
No Borrables cuando son ROM con mascara	Borrable totalmente con exposición a luz Ultravioleta.	Puede ser barrada parcialmente (palabras), eléctricamente	No Borrables	No Borrables	Borrables y reprogramables
No volátil	No volátil	No volátil	No volátil	No volátil	No volátil

C.S1.1 Tabla Comparativos entre los Dispositivos de Fusibles Programables

Fig1.10 Clasificación básaa en la arquitectura interna de los PALs estándard

* Programador de Arreglos lógicos tesis UCA, Sept,1987 pag.8

1.5.1 Lógica de Arreglo Programable que posee registros en la salida.

Estos poseen un registro tipo D a la salida, ya que puede ser realimentado internamente y permitir mantener información de estados lógicos aún cuando la salida no esté habilitada. Además, la

Salida puede controlarse por medio de un habilitador programable que es común a todos los registros (fig. 1.11). La posibilidad de controlar la salida hace del PALs con registros el dispositivo ideal para aplicaciones en sistemas de buses de datos, registrarlos u operar con ellos, colocándolos en el bus de datos al habilitar el buffer de salida.

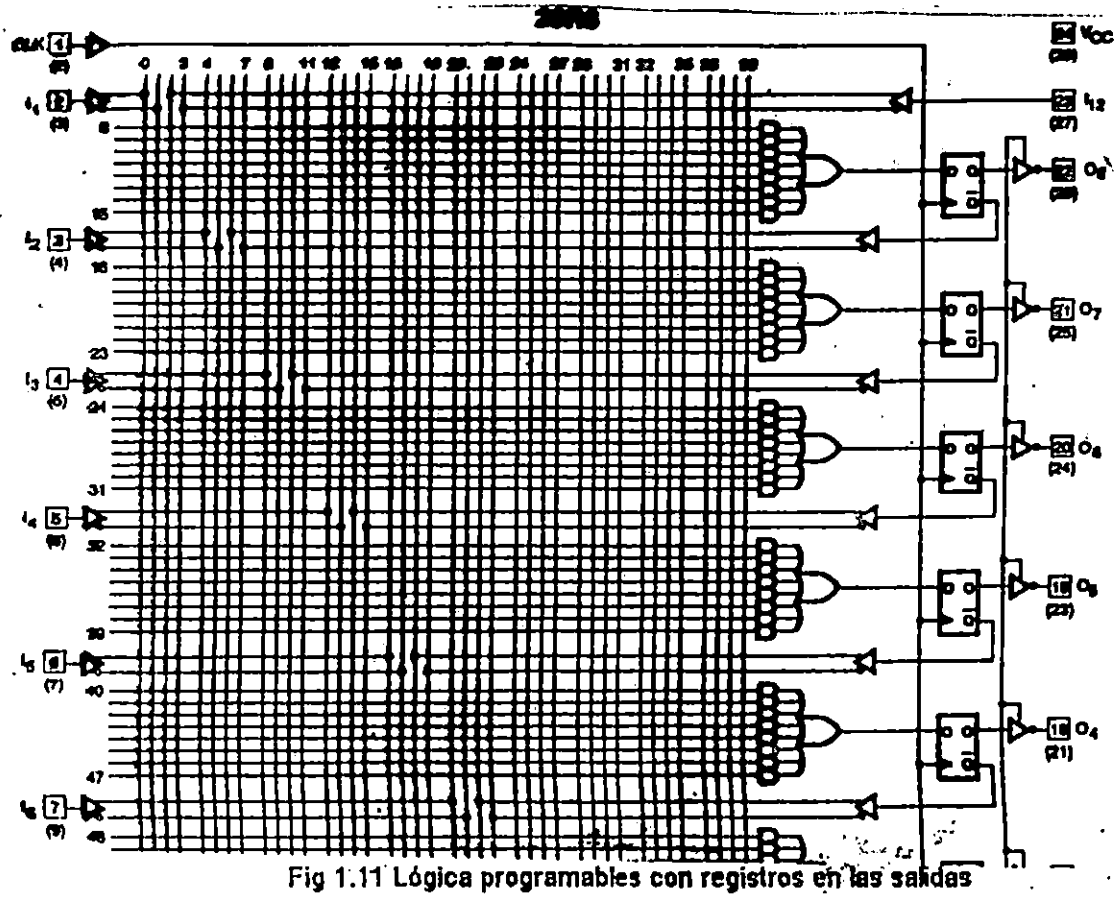


Fig 1.11 Lógica programables con registros en las salidas

Fig. 1.11 Lógica Programable con registros en las salidas

1.5.2 Lógica de Arreglo Programable que poseen pines de Entradas/Salidas* Programables.

Los diagramas lógicos para un PAL con estructura de salida bidireccional. Una de las características de este tipo de PAL es la habilidad de programar el habilitador (enable) de salida como una función de una compuerta AND en el arreglo. El "Buffer" de salida puede ser programado de tres formas siguientes:

1. **Con una salida Asignada;** en este caso el buffer de salida esta siempre habilitado y la función lógica sé realimenta al arreglo AND, permitiendo el desarrollo de funciones lógicas más complejas utilizando dos o más niveles de compuertas AND-OR.
2. **Como una entrada asignada;** no utiliza la compuerta AND-OR asociada al pin Enable, con lo que se crea una entrada extra. Esta capacidad de intercambiar salidas por entradas es una de las grandes ventajas del PALs sobre otros dispositivos programables.
3. **Como una Entrada/Salida dinámicamente controlable;** haciendo una combinación lógica de una o más entradas Habilitado/Deshabilitado. El habilitador(Enable) puede ser utilizado como una entrada, así como también, reteniendo la completa capacidad lógica de las compuertas AND-OR.

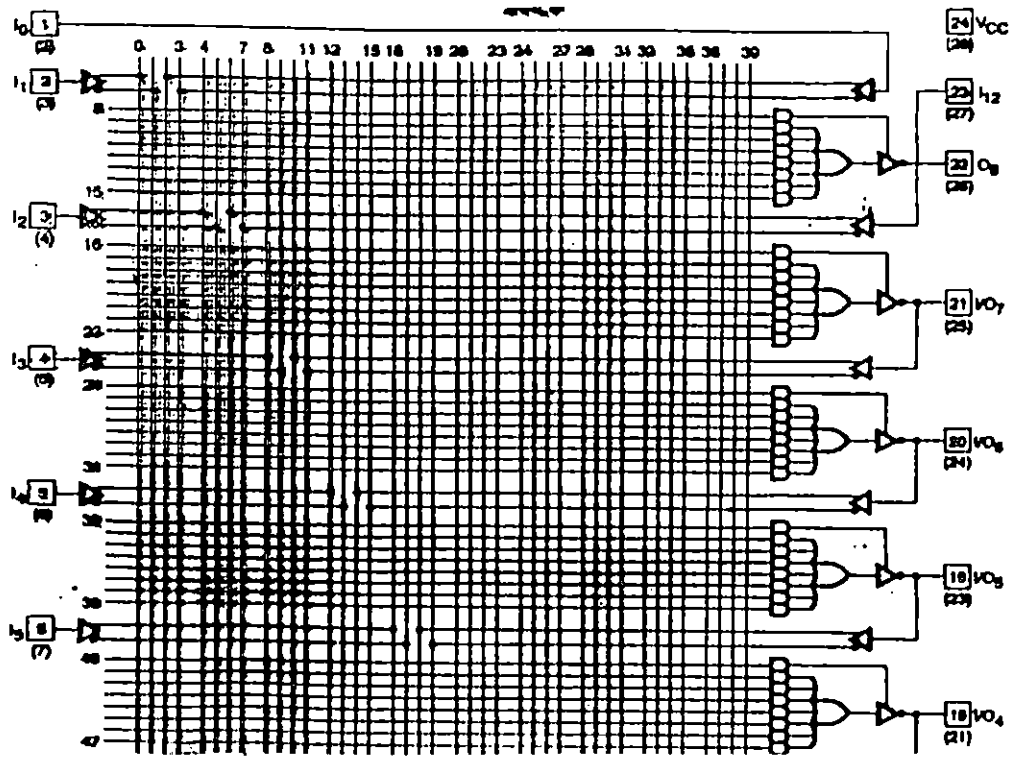


Fig. 1.11 Lógica Programable con registros en las salidas

* Para mayor información refiérase al libro "Advanced Micro Devices Programable Array Logic Handbook". Cap. Y pag 17 Edi.1984

1.5.3 Lógica de Arreglo Programable con salidas activo alto/ activo bajo.

Se presenta (fig. 1.13), a continuación las salidas disponibles en activo alto/activo bajo del PAL de salidas asignadas; las salidas siempre están habilitadas.

La compuerta AND(en otros tipos utilizada para la función de habilitación) provee un término lógico AND extra en este tipo de PAL. Esto eleva el número de compuertas AND por salida a ocho; el lazo de realimentación de la salida a la entrada aún está presente, permitiendo la implementación de lógica de varios niveles.

La compuerta AND extra hace a las salidas ideales para reemplazo de lógica no orientada a buses de datos, generación de señales de control especialmente complejas, codificación y decodificación.

1.5.4 Sistemas de Codificación y Instrucciones para Ordenar el PALs

Los fabricantes de semiconductores de PLDs han estandarizado la codificación para que el usuario les sea fácil realizar una orden de compra de esté tipos de dispositivos, con la cual se puede identificar a cada chip; obteniendo así las condiciones de operación, y otra información necesaria pero general del dispositivo; y así cualquier diseñador de circuitos electrónicos pueda conocer una idea general de las características del chips y sus propiedades, y así saber si es el PAL apropiado a su aplicación.

Pero en esté trabajo sólo se contempla los fabricantes de semiconductores siguientes:

- **Advanced Micro Device AMD**
- **Texas Instruments TI**

1.5.4.1Codificación en Advanced Micro Device AMD.

Los productos PALs de AMD para aplicaciones comerciales están facilitados con una variedad de opciones (determinadas por su arquitectura interna), para ordenar el PAL estándar que en este caso se programaran de 20 pines. Se presenta(ver fig. 1.14) la siguiente codificación de ellos.

Para realizar la solicitud de cualquier dispositivo de lógica de arreglo programable por fusibles se presentan los diferentes campos que los conforman.

1. El campo **"PAL"** representa el tipo de familia a la que pertenece el dispositivo de Lógica de Arreglo Programable.
2. El campo especifica el número de entradas en el arreglo de lógica programable.

3. El campo representa el número de salidas con las que cuenta la lógica de arreglo programable.
4. El campo informa de la cantidad de salidas
5. El campo de la codificación especifica la velocidad del dispositivo a usar en una aplicación especifica.
6. El campo informa del tipo de revisión.
7. El campo informa sobre lo relacionado al tipo de encapsulado
8. El campo nos facilita la información del rango de temperatura en la que puede operar el dispositivo de lógica de arreglo programable.
9. El campo informa el tipo de proceso realizado por el integrado.

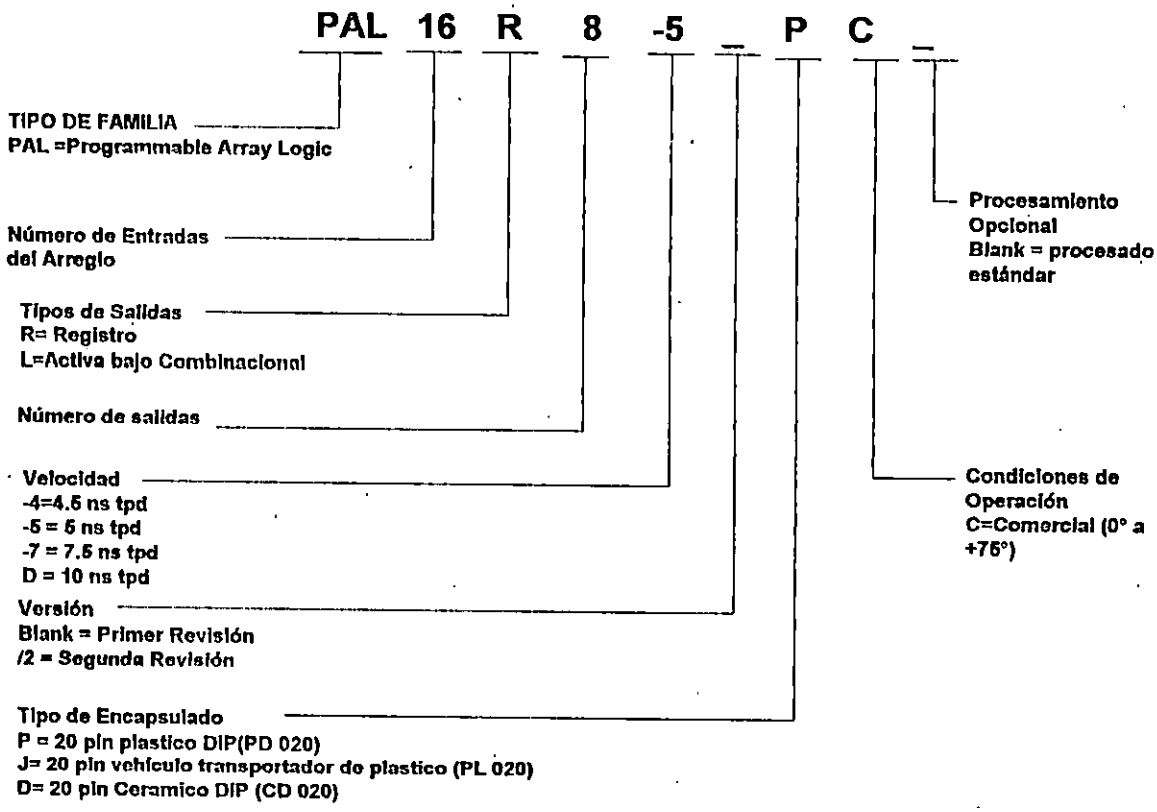


Fig 1.14 Codificación para identificar los PAL's en AMD

1. 5.4.2 Codificación en Texas Instruments TI.

El fabricante Texas Instruments usa la siguiente codificación (ver fig. 1.15) para identificar los dispositivos de Lógicas de Arreglo Programable PALs.

En el PALs de Texas Instruments está conformado la codificación por nueve campos que a continuación se detallan:

1. En el primero campo se describe la tecnología utilizada para la construcción del dispositivo.

2. En el segundo campo se hace referencia a la familia a que pertenece el dispositivo.
3. En el tercero campo sé especifica el tipo de registro de entrada.
4. En el cuarto campos de informa la cantidad de entrada al arreglo.
5. En el quinto campo sé especifica la configuración de salida.
6. En el sexto campo se lee el número de salida.
7. En el séptimo campo sé especifica la potencia y la velocidad
8. En el octavo campo especifica el rango de temperaturà de operación de los chips.
9. En el noveno campo se describe el tipo de encapsulado del integrado.

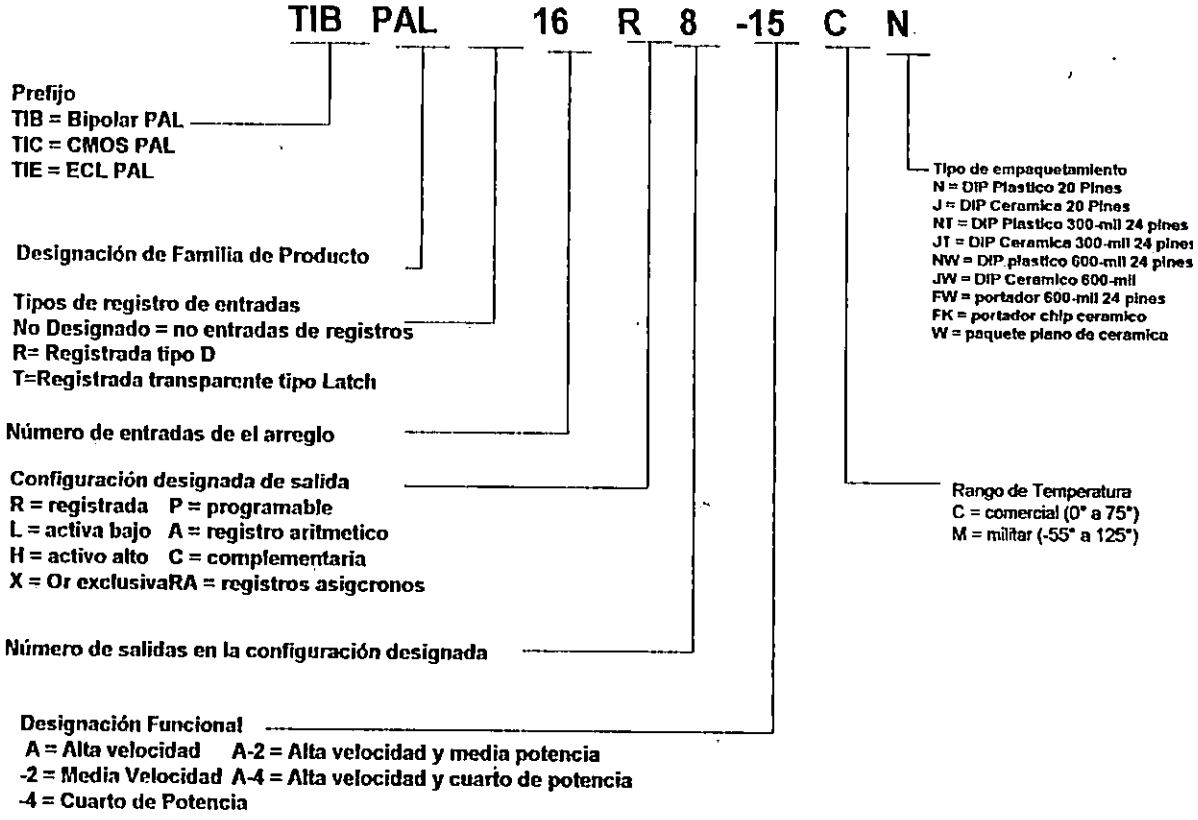


Fig 1.15 codificación de los dispositivos de lógica programable de Texas Instruments

1.5.5 Secuencia de Programación del PALs estándar de los fabricantes Texas Instruments y Advanced Micro Device

Como se ha descrito en lo que lleva del trabajo todo tipo PAL esta compuesto por compuertas lógicas y matrices de fusibles, donde inicialmente todos están intactos; el proceso de programación consiste en la eliminación de fusibles que corresponden a los términos no deseados en la función lógica de salida en el diseño. En la fig. 1.16 se

observan los pasos a seguir en proceso de la programación de los dispositivos de lógica programable.

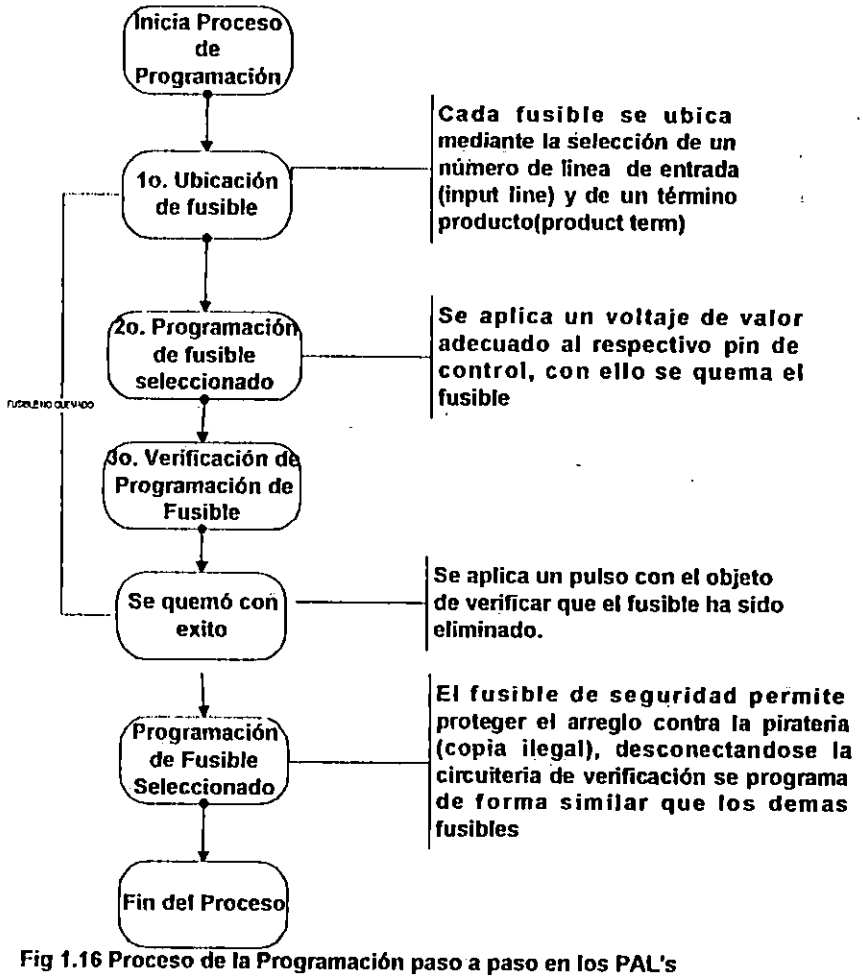


Fig 1.16 Proceso de la Programación paso a paso en los PAL's

En términos generales, el proceso de programación de los PLDs sigue una secuencia de pasos determinados los cuales se presentan en la fig. 1.16, y a continuación se describen las etapas:

1. Se ubica el fusible a programar (quemar o eliminar) mediante la selección de número de **Línea de Entrada (Input Line),** que corresponde al arreglo de filas, y de un **Término de Producto (Product Terms),** que corresponde al arreglo de columnas y de salidas, una a la vez.
2. El fusible a eliminar se programa aplicando una secuencia de voltajes a los respectivos pines de control. El nivel de voltaje necesario a aplicar depende de fabricante por ejemplo para Texas Instruments son 5v, 10.5v, 21v y alta impedancia. Mientras que los de Advenced Micro Device aplican 5v,21v y alta impedancia. Por esta diferencia de niveles de voltajes no se puede generalizar la secuencia de programación.
3. Se verifica que el fusible halla sido programado(es decir eliminado), introduciendo un pulso al pin de verificación, y observando el nivel de la señal en el respectivo pin de salida, si el nivel de voltaje es mayor que cero entonces el fusible no se quemó, y si es cero el quemado fue un éxito.

4. Se programa el fusible de seguridad, de forma similar que en los pasos anteriores. **El fusible de seguridad sirve para proteger el arreglo contra la piratería, previniendo su copia.** Una vez quemado este fusible, la circuiteria de verificación y precarga de los registros de salida queda separada permanentemente la circuiteria par realizar la lectura del PAL.

1.5.6 Proceso de Programación de los PAL's en Texas Instruments y los Advanced Micro Device.

El PAL al igual que todos los dispositivos de lógica programable, necesitan de herramientas de software y hardware para realizar la programación de sus fusibles, y así implementar sustituciones de circuitos lógicos de MSI/SSI. Este sistema consta básicamente de tres elementos: el primer es el software, el cual proporciona la secuencia de pasos necesarios para la ejecución de dicho algoritmo, el segundo es el ordenador necesario, y el tercero es el hardware necesario para garantizar los diferentes niveles de programación y verificación, en el proceso de programación.

Debido a que cada fabricante proporciona su propia secuencia de pasos, no es posible generalizar que las herramientas de software sea útil para todos los fabricantes de PALs.

Es así como en este trabajo sólo se toma en cuenta el PALs de 20 pines y de 24 pines de Texas Instruments, y solamente los de 20 pines de AMD. De estos últimos por no poder tener acceso a las formas de onda de programación del PALs de 24 pines.

1.5.6.1 Proceso en Texas Instruments.

El PALs estándar de 20 pines y de 24 pines no presenta la opción de **precarga.** La salida puede ser activo bajo para algunos de ellos o con registros de tipo D. Los cuales son los siguientes:

TIBPAL 16R8	PAL16R8A	PAL20R8A
TIBPAL16R6	PAL16R6A	PAL20R6A
TIBPAL16R4	PAL16R4A	PAL20R4A
TIBPAL16RL8	PAL16L8A	PAL20L8A

En las figuras 1.17,1.18,1.19 se presentan las configuraciones de los pines de los PALs estándar del fabricante Texas Instruments que se deben de respetar para la programación. En cada uno de los pines de estas configuraciones le corresponde un determinado nivel de voltaje en el proceso de programación.

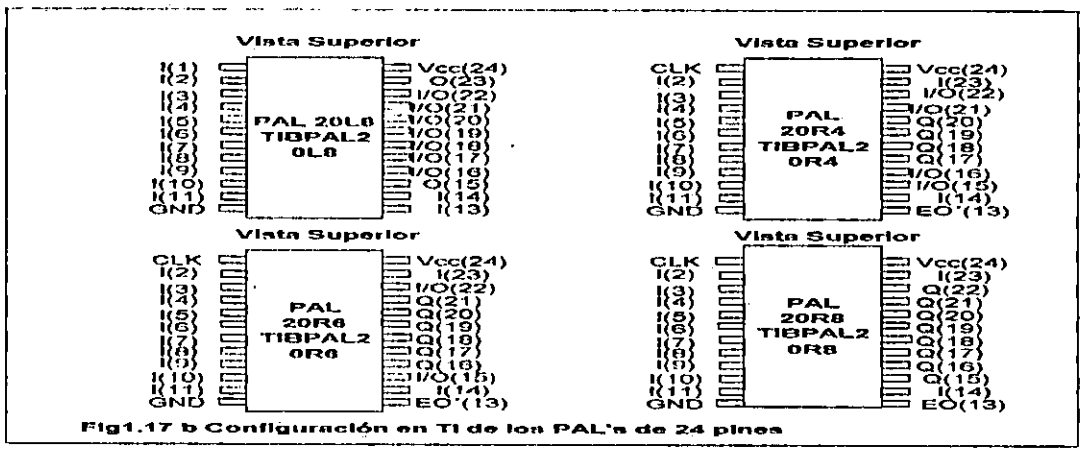


Fig1.17 b Configuración en TI de lon PAL's de 24 pines

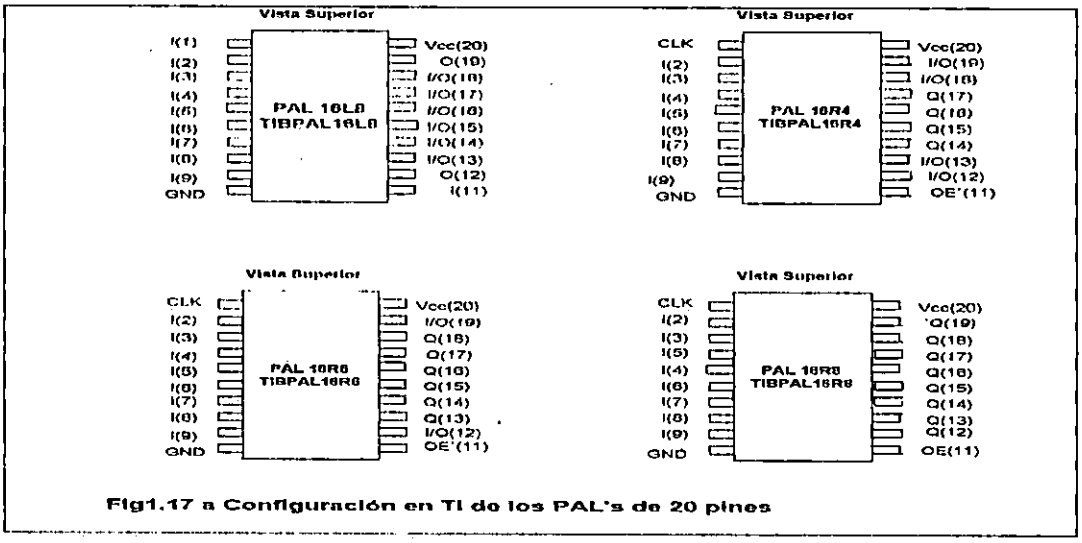


Fig1.17 a Configuración en TI de los PAL's de 20 pines

En la fig. 1.17 se presenta la configuración de los pines para las líneas productos para 20 pines, además en la fig. 1.17 se presenta la configuración del PALs de 24 pines que se tiene que respetar para realizar la programación de cualquier fusible entre las posiciones de las líneas producto desde 0 a la 63. A continuación se documenta la manera de como se selecciona las líneas de entrada y producto. Colocando en los respectivos pines los valores de voltajes respectivos que se especifican en las tablas 1.1,tabla 1.2 y tabla 1.3 siguientes.

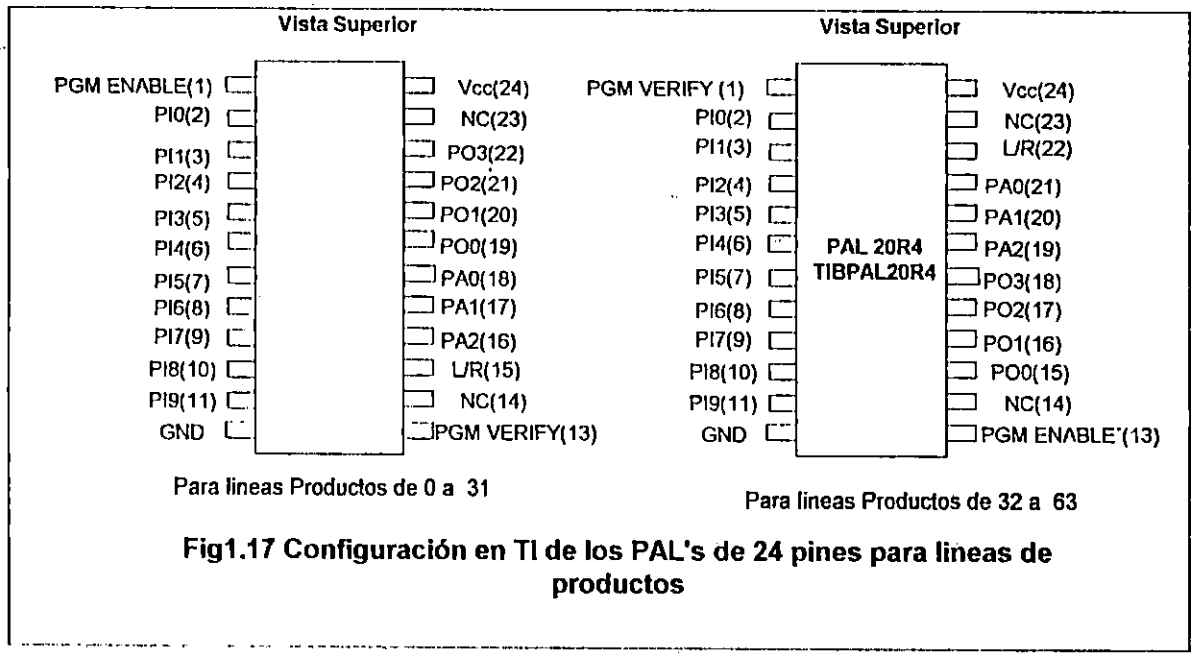


Fig1.17 Configuración en TI de los PAL's de 24 pines para lineas de productos

1.5.6.1.1 Selección de las líneas de entradas y productos

Los arreglos de fusibles se seleccionan mediante la ubicación de las **líneas de entrada (Input Lines) y Líneas Producto (Product Lines).** El PALs de 20 pines disponen de 32 líneas de entrada y 64 líneas de producto; y el PALs de 24 pines, tiene 40 líneas de entrada y 64 líneas de producto. Cada fusible puede ser abierto por medio de la selección apropiada de una línea de entrada y luego eligiendo la línea de producto correcta.

Para seleccionar la línea producto (Product Line), existen dos configuraciones, tanto para el PALs de 20 o el PALs de 24 pines las cuales se presentan a continuación.

1. **Para determinar los términos de producto:** (Product Terms) del 0 al 31. Los niveles de voltaje se aplican en los pines PO y PA (ver fig. 1.18), siendo éstos, los valores indicados en la tabla 1.1 para los PAL's de 20 pines y la tabla 1.3 para los PAL's de 24 pines.

2. **Para determinar términos de producto: (Product Terms)** del 32 al 63, los voltajes para PO y PA(ver fig. 1.18) se indican en las tablas 1.2 para los PAL's de 20 pines y Tabla 1.3 para los PAL's de 24 pines respectivamente.

TABLA 1.1 Pines y voltajes para seleccionar líneas de entrada (20 pines)

Línea de Entrada	Nombre del Pin								
Número	PI7	PI6	PI5	PI4	PI3	PI2	PI1	PI0	L/R
0	*HH	HH	HH	HH	HH	HH	HH	L	Z
1	HH	HH	HH	HH	HH	HH	HH	H	Z
2	HH	HH	HH	HH	HH	HH	HH	L	HH
3	HH	HH	HH	HH	HH	HH	HH	H	HH
4	HH	HH	HH	HH	HH	HH	L	HH	Z
5	HH	HH	HH	HH	HH	HH	H	HH	Z
6	HH	HH	HH	HH	HH	HH	L	HH	HH
7	HH	HH	HH	HH	HH	HH	H	HH	HH
8	HH	HH	HH	HH	HH	L	HH	HH	Z
9	HH	HH	HH	HH	HH	H	HH	HH	Z
10	HH	HH	HH	HH	HH	L	HH	HH	HH
11	HH	HH	HH	HH	HH	H	HH	HH	HH
12	HH	HH	HH	HH	L	HH	HH	HH	Z
13	HH	HH	HH	HH	H	HH	HH	HH	Z
14	HH	HH	HH	HH	L	HH	HH	HH	HH
15	HH	HH	HH	HH	H	HH	HH[1]	HH	HH
16	HH	HH	HH	L	HH	HH	HH	HH	Z
17	HH	HH	HH	H	HH	HH	HH	HH	Z
18	HH	HH	HH	L	HH	HH	HH	HH	HH
19	HH	HH	HH	H	HH	HH	HH	HH	HH
20	HH	HH	L	HH	HH	HH	HH	HH	Z
21	HH	HH	H	HH	HH	HH	HH	HH	Z
22	HH	HH	L	HH	HH	HH	HH	HH	HH
23	HH	HH	H	HH	HH	HH	HH	HH	HH
24	HH	L	HH	HH	HH	HH	HH	HH	Z
25	HH	H	HH	HH	HH	HH	HH	HH	Z
26	HH	L	HH	HH	HH	HH	HH	HH	HH
27	HH	H	HH	HH	HH	HH	HH	HH	HH
28	L*	HH	HH	HH	HH	HH	HH	HH	Z
29	H*	HH	HH	HH	HH	HH	HH	HH	Z
30	L	HH	HH	HH	HH	HH	HH	HH	HH
31	H	HH	HH	HH	HH	HH	HH	HH	HH

Tabla 1.2 Pines y voltajes para seleccionar líneas de producto (24 y 20 pines)

Línea de Producto	Nombre del Pin						
Número	PO0	PO1	PO2	PO3	PA2	PA1	PA0
0,32	Z	Z	Z	HH	Z	Z	Z
1,33	Z	Z	Z	HH	Z	Z	HH
2,34	Z	Z	Z	HH	Z	HH	Z
3,35	Z	Z	Z	HH	Z	HH	HH
4,36	Z	Z	Z	HH	HH	Z	Z
5,37	Z	Z	Z	HH	HH	Z	HH
6,38	Z	Z	Z	HH	HH	HH	Z
7,39	Z	Z	Z	HH	HH	HH	HH
8,40	Z	Z	HH	Z	Z	Z	Z
9,41	Z	Z	HH	Z	Z	Z	HH
10,42	Z	Z	HH	Z	Z	HH	Z
11,43	Z	Z	HH	Z	Z	HH	HH
12,44	Z	Z	HH	Z	HH	Z	Z
13,45	Z	Z	HH	Z	HH	Z	HH
14,46	Z	Z	HH	Z	HH	HH	Z
15,47	Z	Z	HH	Z	HH	HH	HH
16,48	Z	HH	Z	Z	Z	Z	Z
17,49	Z	HH	Z	Z	Z	Z	HH
18,50	Z	HH	Z	Z	Z	HH	Z
19,51	Z	HH	Z	Z	Z	HH	HH
20,52	Z	HH	Z	Z	HH	Z	Z
21,53	Z	HH	Z	Z	HH	Z	HH
22,54	Z	HH	Z	Z	HH	HH	Z
23,55	Z	HH	Z	Z	HH	HH	HH
24,56	HH	Z	Z	Z	Z	Z	Z
25,57	HH	Z	Z	Z	Z	Z	HH
26,58	HH	Z	Z	Z	Z	HH	Z
27,59	HH	Z	Z	Z	Z	HH	HH
28,60	HH	Z	Z	Z	HH	Z	Z
29,61	HH	Z	Z	Z	HH	Z	HH
30,62	HH	Z	Z	Z	HH	HH	Z
31,63	HH	Z	Z	Z	HH	HH	HH

[1]

* IIH = V_{IH} = 10.5 voltios II = V_{II} = 5.0 voltios L = V_{IL} = 0 voltios (0.8 máximo)
V_{OH} = 2.4 voltios máximo V_{OL} = 0.4 voltios (máximo) Z = 10K a 5.0v

1.5.6.1.2 Secuencia de Programación

Los pasos a seguir para programar un PAL's de 20 pines y 24 pines se presenta a continuación para los PAL's estándar de Texas Instrument, todos los pasos se deben ir confirmando con la fig1.19:

Pasos:

1. Se eleva el pin habilitador de programación (PGM ENABLE) a un nivel VIHH[91]
2. Se selecciona la línea de entrada (INPUT LINE) deseada, aplicando niveles de voltaje apropiados a los pines L\R y PI según se muestra en las tablas 1.1 y 1.3.
3. Inicialmente la selección de la línea de salida (OUTPUT LINE) colocando los respectivos niveles de voltaje a los pines observar tabla 1.1 y 1.3.
4. El voltaje de alimentación del pin Vcc se eleva a VIHH
5. Se quema el fusible seleccionado, elevando el voltaje de los pines PO apropiados a VIHH (Observar las tablas 1.2), en donde se pueden observar los valores de PO.
6. Se reduce el voltaje en el pin Vcc a su nivel normal (+5v). A continuación se le envía un pulso al pin PGM VERIFY, con el objeto de verificar la efectividad de la programación. Para comprobar qué fusible seleccionado (PASO 5) ha sido programado, el nivel de voltaje en el correspondiente pin PO será inferior a VOL; en caso contrario, el fusible no ha sido quemado. La verificación sólo es posible si el fusible de seguridad se encuentra sin programar. Si la verificación indica que el fusible no quemó, los pasos anteriores (1 al 6) pueden repetirse con un máximo de cuatro veces.

A continuación mostramos el diagrama de [9]temporización para Programar el PALs de 20 y 24 pines.

En la fig. 1.19 se muestras las formas de ondas de programación del PALs de Texas Instrument, estas formas de onda son reflejo de los valores de voltajes que se encuentran en las tablas 1.1,1.2, 1.3. Como se puede observar que el primer nivel de voltaje aplicado es en el pin Vcc a 5v, y seguidamente en el pin ENABLE a 10.5V, y seguidamente se coloca la combinación de voltajes en los pines de entradas(2 al 11 para 24 pines, y 2 al 9 para 20 pines), según valores especificados por la tabla 1.1(para 20 pines) y tabla 1.3 (para 24 pines). En ese mismo orden la tercera ondas de programación es el del pin Vcc que tiene que llevarse desde 5v a 10.5v, posteriormente cuando todos los valores anteriores de voltajes se encuentran en los niveles deseados se procede a realizar la selección de los fusibles a programar con los pines PO y PA, colocando en los pines del 12 al 19 (para 20 pines) o de los pines 15 al 22 (para 24 pines) los valores correspondientes en esos pines se muestran en la tabla 1.2. Es en este momento cuando se

Tabla 1.3 Pines y voltajes para seleccionar la línea de entrada

Línea de Entrada	Nombre del Pin										
Número	PI9	PI8	PI7	PI6	PI5	PI4	PI3	PI2	PI1	PI0	L/R
0	*HH	HH	HH	HH	HH	HH	HH	HH	HH	H	Z
1	HH	HH	HH	HH	HH	HH	HH	HH	HH	L	Z
2	HH	HH	HH	HH	HH	HH	HH	HH	HH	H	HH
3	HH	HH	HH	HH	HH	HH	HH	HH	HH	L	HH
4	HH	HH	HH	HH	HH	HH	HH	HH	L	HH	Z
5	HH	HH	HH	HH	HH	HH	HH	HH	H	HH	Z
6	HH	HH	HH	HH	HH	HH	HH	HH	L	HH	HH
7	HH	HH	HH	HH	HH	HH	HH	HH	H	HH	HH
8	HH	HH	HH	HH	HH	HH	HH	L	HH	HH	Z
9	HH	HH	HH	HH	HH	HH	HH	H	HH	HH	Z
10	HH	HH	HH	HH	HH	HH	HH	L	HH	HH	HH
11	HH	HH	HH	HH	HH	HH	HH	H	HH	HH	HH
12	HH	HH	HH	HH	HH	HH	L	HH	HH	HH	Z
13	HH	HH	HH	HH	HH	HH	H	HH	HH	HH	Z
14	HH	HH	HH	HH	HH	HH	L	HH	HH	HH	HH
15	HH	HH	HH	HH	HH	HH	H	HH	HH	HH	HH
16	HH	HH	HH	HH	HH	L	HH	HH	HH	HH	Z
17	HH	HH	HH	HH	HH	H	HH	HH	HH	HH	Z
18	HH	HH	HH	HH	HH	L	HH	HH	HH	HH	HH
19	HH	HH	HH	HH	HH	H	HH	HH	HH	HH	HH
20	HH	HH	HH	HH	L	HH	HH	HH	HH	HH	Z
21	HH	HH	HH	HH	H	HH	HH	HH	HH	HH	Z
22	HH	HH	HH	HH	L	HH	HH	HH	HH	HH	HH
23	HH	HH	HH	HH	H	HH	HH	HH	HH	HH	HH
24	HH	HH	HH	L	HH	HH	HH	HH	HH	HH	Z
25	HH	HH	HH	H	HH	HH	HH	HH	HH	HH	Z
26	HH	HH	HH	L	HH	HH	HH	HH	HH	HH	HH
27	HH	HH	HH	H	HH	HH	HH	HH	HH	HH	HH
28	HH	HH	L	HH	HH	HH	HH	HH	HH	HH	Z
29	HH	HH	H	HH	HH	HH	HH	HH	HH	HH	Z
30	HH	HH	L	HH	HH	HH	HH	HH	HH	HH	HH
31	HH	HH	H	HH	HH	HH	HH	HH	HH	HH	HH
32	HH	L	HH	HH	HH	HH	HH	HH	HH	HH	Z
33	HH	H	HH	HH	HH	HH	HH	HH	HH	HH	HH
34	HH	L	HH	HH	HH	HH	HH	HH	HH	HH	Z
35	HH	H	HH	HH	HH	HH	HH	HH	HH	HH	HH
36	L*	HH	HH	HH	HH	HH	HH	HH	HH	HH	Z
37	H*	HH	HH	HH	HH	HH	HH	HH	HH	HH	HH
38	L	HH	HH	HH	HH	HH	HH	HH	HH	HH	Z
39	H	HH	HH	HH	HH	HH	HH	HH	HH	HH	HH

realiza la programación del fusible, con la orden que manda el nivel de voltaje en el pin PO. En todo este tiempo transcurrido el pin PGM VERIFY esta a 0V.

Es aquí cuando comienza el período de verificación del fusible quemado. Se retira el nivel de voltaje en el pin PO seleccionado, y aplicando un pulso de voltaje en el pin PGM VERIFY de 5v, y el pin el nivel en el pin Vcc a 5v, y seguidamente se procede a realizar la lectura de voltaje en el pin PO seleccionado anteriormente, si el nivel es mayor que cero entonces no se quemó el fusible, pero si el voltaje es cero entonces si se quemó con éxito el fusible. Si todo fue un éxito se procede[1] a retirar los niveles de voltajes de los pines de entradas, Enable, Vcc, PO, PA, verify. Y seleccionar otro fusible.

[1] HH = V_{HH} = 10.5 voltios H = V_{IH} = 5.0 voltios L = V_{IL} = 0 voltios (0.8 máximo)
V_{OH} = 2.4 voltios máximo V_{OL} = 0.4 voltios (máximo) Z = 10K a 5.0v

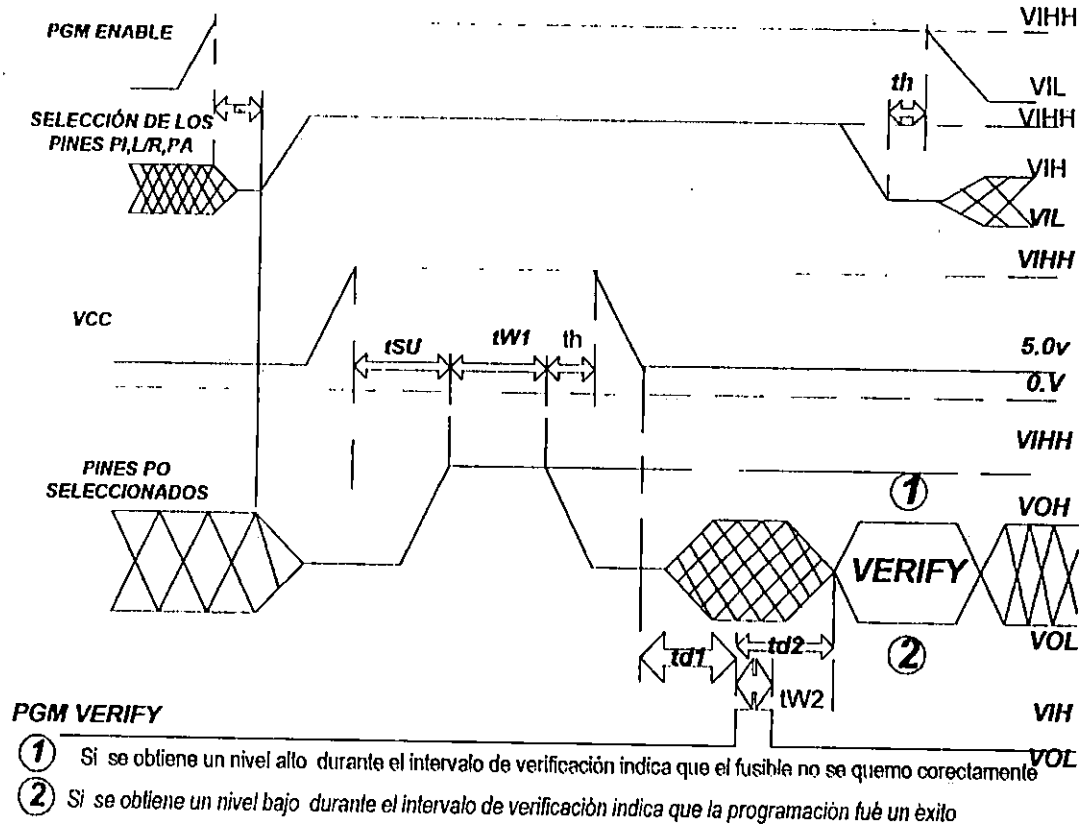


Fig. 1.19 Forma de onda de programación de los PALs Estándar en Texas Instruments

Para realizar la programación del fusible de seguridad, si es de 20 pinés el pin VCC se lleva a cero "0" voltios. Mientras se tenga en ese valor, se aplican pulsos al pin 1, primeramente, y luego al pin 11. en la fig. 1.20 se muestra el diagrama de temporización.

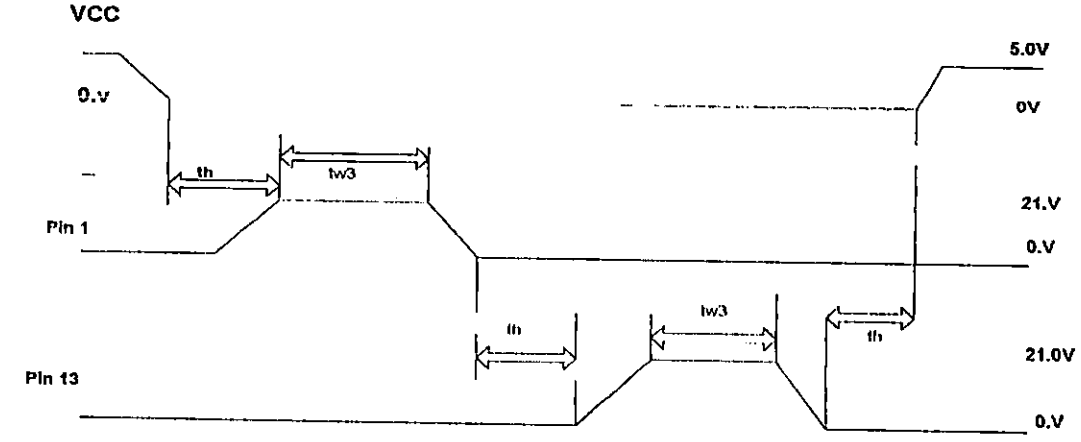


Fig1.20 Diagrama de formas de ondas para programar el fusible de seguridad de los PALs de 20 pines

Por el contrario, si el PAL es de 24 pines, para el fusible de seguridad, Vcc se lleva a 5v y siempre se aplican pulsos pero ahora a los pines 1 y 13. los pulsos de niveles de voltajes que son aplicados son de 21 voltios en los pines respectivos. En la fig. 1.21 se muestra la temporalización del quemado del fusible de seguridad del PALs de 24 pines,

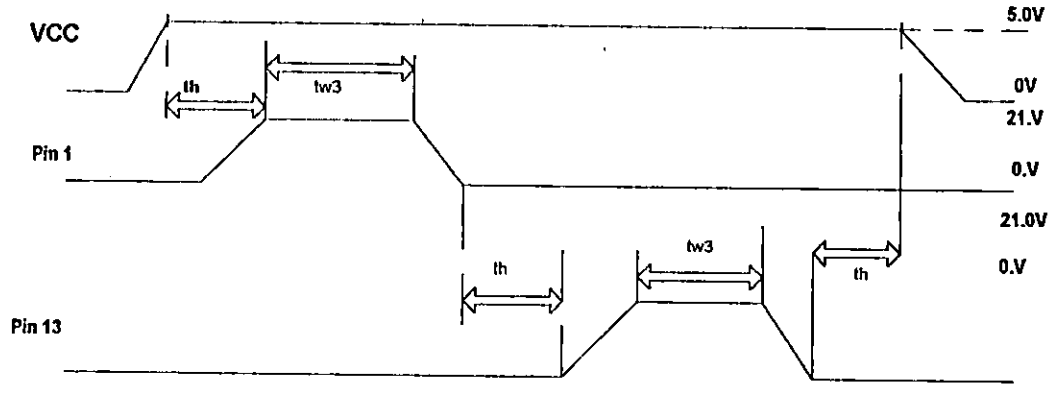


Fig1.21 Diagrama de formas de ondas para programar el fusible de seguridad de los PALs de 24 pines

1.5.6.2 Proceso de programación en AMD Advanced Micro Devices para PALs de 20 pines

Cada fusible es programado con una simple secuencia de voltajes aplicados a dos pines de control (pin 1 y pin 11), y un pulso de programación aplicado a la salida seleccionada en programación. El direccionamiento de los 2048 fusibles de la arquitectura interna del PALs, es realizado con niveles de voltajes TTL normales en ocho pines de entrada (cinco seleccionan el número de líneas de entradas y tres seleccionan el número de líneas de término de producto). Vcc es mantenido a un nivel normal durante toda la programación y en el ciclo de verificación no se requiere niveles altos.

La secuencia necesaria de niveles de voltajes para la programación es mostrada en el diagrama de tiempos (fig.1.22). La dirección de cada fusible se realiza con la selección de cada uno de los términos de número de líneas de entradas y el número del término de producto, los cuales están definidos por las tablas de direccionamiento de fusible (tablas 1.4 y 1.5) Los parámetros de corriente, voltaje y requerimiento de tiempo para cada pin son especificados en los anexos. Después de que toda la programación a sido terminada, el arreglo completo deberá ser verificado. Esto se inicia con el PAL en el pin Vcc a 5v. La verificación puede ser realizada en la lectura de cualquiera de las ocho salidas seleccionada en la programación, a continuación se explicará la secuencia de pasos necesarios para poder programar cualquier de los 2048 fusibles de la arquitectura interna de un PAL de la familia AMD de veinte pines.

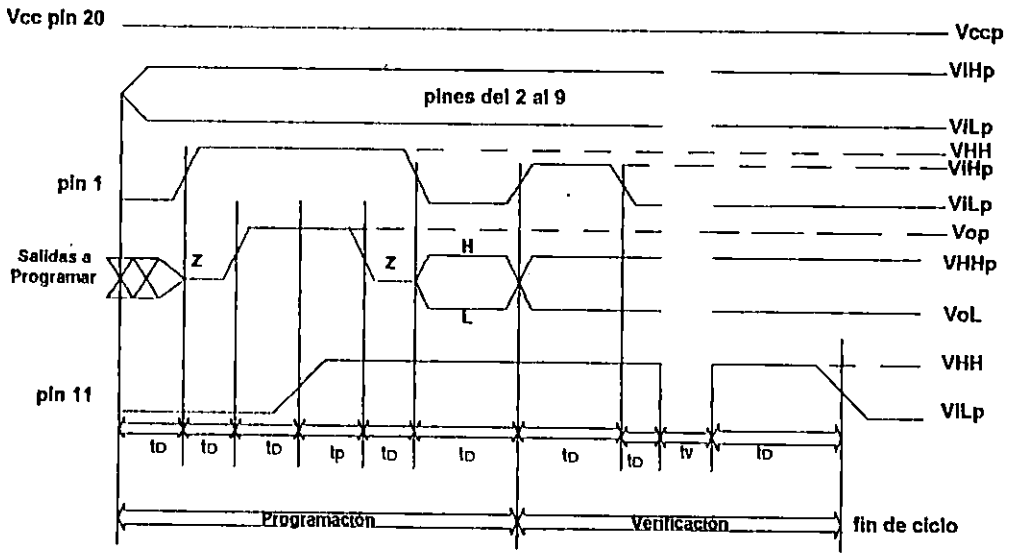


Fig. 1.22 Diagrama de tiempo de programación de los PALs de 20 pines de AMD

El primer paso es llevar la alimentación al circuito integrado, es decir, colocar el pin veinte a Vccp[V] . Luego, se direcciona el fusible a quemar. EL direccionamiento es realizado por medio de ocho pines, cinco de los cuales (9,8,7,6 y 5) seleccionan la columna (número de líneas de entrada) donde se encuentra localizado el fusible a quemar. Cada una de las 32 columnas es seleccionada por la combinación binaria respectiva en las entradas de los pines descritos con anterioridad (tabla 1.4).

Los tres pines restantes (2, 3 y 4) seleccionan una fila (número de líneas del término producto) de cada uno de los ocho arreglos; es decir, el fusible a quemar estará ubicado en cualquier de las ocho intersecciones(cada intersección representa un fusible), formada por la columna escogida y las ocho filas seleccionadas en los respectivos arreglos. Cada una de las ocho filas de cada arreglo es seleccionada por la combinación binaria respectiva en las entradas de los pines mencionados. Luego que el direccionamiento se ha realizado, se prosigue a deshabilitar las salidas de los ocho arreglos y convertirlas en entradas de programación, esto se realiza llevando el pin 1 a VHH. En estos momentos las salidas, de los ocho arreglos (pines 12 al 19) están listas para aceptar el voltaje de programación (Vop). Sólo la salida del arreglo donde se encuentra el fusible a quemar, se llevará hasta el voltaje de programación durante un tiempo de 140us.

Ahora todo está listo para realizar el quemado del fusible deseado, sólo falta proporcionar la orden, esto se realiza llevando a pin 11 a VHH.

Luego de realizado el quemado del fusible, el voltaje de programación es eliminado y las salidas son de nuevo habilitadas, llevando el Pin 1 al nivel de voltaje de VIHp. Es acá cuando termina el ciclo de programación y comienza el de verificación.

La verificación se realiza, poniendo el voltaje VIHp en el pin 1 durante un tiempo mínimo TD y llevando nuevamente el pin 1 al nivel de voltaje de VILp. En este instante estamos listos para leer en las salidas de los arreglos y comprobar si el fusible ha sido

[V] Vccp=5.0 VIHP= 5.0v VOP= 21.0v VILP= 0.0v H=5.0v, L=0v

programado con éxito. Si el PAL que sé esta programando es activo en bajo y si el fusible ha sido exitosamente quemado, un nivel bajo deberá leerse en las salidas. Si por el contrario, el PAL es activo en alto y el fusible ha sido exitosamente quemado un nivel alto deberá leerse.

Si el fusible no ha sido exitosamente quemado, todo el proceso se repetirá pero con un tiempo mayor de programación del fusible(tiempo de fusión). En resumen la secuencia de pasos de la siguiente manera:

Tabla 1.4 Direccionamiento del Término Producto

Número de líneas del término del producto								Pines de Direccionamiento del Término Producto		
pin 19	pin 18	pin 17	Pin 16	pin 15	pin 14	pin 13	pin 12	4	3	2
0	8	16	24	32	40	48	56	L	L	L
1	9	17	25	33	41	49	57	L	L	H
2	10	18	26	34	42	50	58	L	H	L
3	11	19	27	35	43	51	59	L	H	H
4	12	20	28	36	44	52	60	H	L	L
5	13	21	29	37	45	53	61	H	L	H
6	14	22	30	38	46	54	62	H	H	L
7	15	23	31	39	47	55	63	H	H	H
Acceso a los pines de Programación y Verificación										

Pasos:

1. Vcc es aplicado al circuito integrado
2. La dirección del fusible a ser programado, es seleccionada por niveles TTL en los pines de direccionamiento apropiados.
3. Las salidas son desahabilitadas
4. El voltaje de programación es aplicado a la salida apropiada.
5. Una habilitación de fusión se lleva a cabo, por la subida de una estrada a un nivel arriba del voltaje de operación TTL normal(el Pin 11 es usado para esto). Esta acción hace que la corriente fluya a través del propio fusible, resulta en un fusible abierto en pocos microsegundos.
6. El voltaje de programación en la salida de 21 v, es disminuido y luego eliminado
7. El dispositivo es habilitado y sincronizado si se requiere. El estado a la salida indica entonces si la programación se ha realizado con éxito. Si la programación no ha ocurrido, una secuencia de pulsos mucho más larga debe de ser aplicada, hasta que la programación ocurra.

La secuencia de los pasos del 2 al 7 se repiten por cada fusible que deba ser programado.

Un fusible adicional se proporciona en cada circuito PAL AMD para prevenir copias no autorizadas del original. Programando el fusible de seguridad se bloquea la entrada al modo de verificación, la siguiente secuencia es la realizada para quemar el fusible de seguridad.

Pasos.

1. Llevar el circuito integrado a 5v
2. Llevar al Pin 5 a 10.5v
3. Aplicar un pulso en el pin 11 desde 0v a 21v durante 50 us.
4. Ejecutar un normal ciclo de verificación al final de la programación

Todas las localidades de fusibles deberán ser censadas como quemadas en el PALs de 20 pines, si el fusible de seguridad ha sido exitosamente quemado.

TABLA 1.5 DIRECCIONAMIENTO DE LINEA DE ENTRADA

Línea de Entrada	Estado de los Pines de direccionamiento del número de líneas de entrada				
Número	9	8	7	6	5
0	L	L	L	L	L
1	L	L	L	L	H
2	L	L	L	H	L
3	L	L	L	H	H
4	L	L	H	L	L
5	L	L	H	L	H
6	L	L	H	H	L
7	L	L	H	H	H
8	L	H	L	L	L
9	L	H	L	L	H
10	L	H	L	H	L
11	L	H	L	H	H
12	L	H	H	L	L
13	L	H	H	L	H
14	L	H	H	H	L
15	L	H	H	H	H
16	L	L	L	L	L
17	H	L	L	L	H
18	H	L	L	H	L
19	H	L	L	H	H
20	H	L	H	L	L
21	H	L	H	L	H
22	H	L	H	H	L
23	H	L	H	H	H
24	H	H	L	L	L
25	H	H	L	L	H
26	H	H	L	H	L
27	H	H	L	H	H
28	H	H	H	L	L
29	H	H	H	L	H
30	H	H	H	H	L
31	H	H	H	H	H

CONCLUSIONES

1.- Para realizar la Programación de los Fusibles en la arquitectura con que cuenta los Arreglos Lógicos Programables, se debe de tener pleno conocimiento de ella.

2.-Las diferencias de las curvas de programación entre los fabricantes de Arreglos Lógicos Programables es esencialmente en los niveles de voltaje requeridos y las altas impedancias que presentas valores distintos para la programación de los fusibles.

3.- El fusibles de seguridad es quemado solamente con la participación de tres pines, y una vez quemado el pin de seguridad no se puede accesar a la información grabada en el arreglo de fusibles, limitandose solamente a utilizarlo como herramienta.

4.- Cuando los fusibles de seguridad se quema permite garantizar que nadie copiara el diseño quemado y solo se utilizara.

5.- Las aplicaciones son variadas pero una de inmediata es que se puede utilizar como key de software, y así proteger las licencias de los software.

RECOMENDACIONES

1. -Recomiendo que en los programas de materias de sistemas Digitales se realice por parte del catedrático la introducción obligatoria de diseños de circuitos digitales utilizando Arreglos de lógica programable PAL's.

2. -Qué sé de inicio al incentivo hacia los educandos para que realicen proyectos de aplicaciones de electrónica Digital con la ayuda del PAL's.

REFERENCIAS BIBLIOGRAFICAS

-Tocci, Ronald J. Sistemas Digitales, Principios y Aplicaciones. México Prentice Hall Hispanoamericana S.A , 5ª. Edición 1993.

-Texas Instruments Enginneering staff. TTL Data Book For Design Engeneers. Dallas, Texas, Texas Instruments Incorporated. 2da. De. 1976

-Texas Instruments General Information Funcional Index Field Programmable Lógic USA, Texas Instruments Incorporated 1985

-Texas Instruments Programmable Lógic Data Book, Dallas Texas, Texas Instruments Incorporated. 1990

- Trabajo de Graduación Programador de Arreglos Lógicos Programables (PALs) El Salvador, UCA 1990

-Bolestar, R y Nashelsky, L. Electronica, Teoría de circuitos. México, Edit. Prentice Hall 4ª. Edit. 1993

-Trabajo de Graduación Diseño y Construcción de un Programador Inteligente de Circuitos de Arreglo Lógico Programable (PAL) Basado en el Microprocesador 8085. El Salvador (Universidad Politecnica) 1996

-Advanced Micro Devices Programmable Array Logic Handbook, 1984

-Trabajo de graduación Diseño e Implementación de un programador de PAL operado por un computador, El Salvador (Unversidad Politecnica) 1987

-Advanced Micro Devices Programmable Array Logic Handbook, 1997

CAPITULO II

DISEÑO Y CONSTRUCCION DE LA INTERFACE DE MANEJO DE LAS FUENTES CONMUTABLES PARA REALIZAR LA PROGRAMACION DE LOS PALs.

Introducción

En este capitulo abordamos el diseño y construcción de la interface controladora para una PC AT compatible IBM, que operará con el puerto paralelo, en el cual se controlará la secuencia de programación de los Dispositivos de Lógica Programable "PAL's" de los fabricantes **TEXAS INSTRUMENTS y ADVANCED MICRO DEVICE**, cuyas características eléctricas entre ellos sean compatibles con los niveles de voltajes disponibles en el programador. El corazón de este circuito es un chip llamado PPI (Interface programable de periféricos), una interface diseñada con este chip hace necesaria una arquitectura de funcionamiento basada más en el software, además del Hardware necesario para generar la conmutación de niveles de voltajes requeridos en los diversos pines de la base ZIF. Todas las funciones básicas del prototipo deberán programarse. Para realizar la ruptura o programación de un fusible en particular, verificación del mismo, hasta la ruptura del fusible de seguridad.

En este capitulo se muestra también procede a mostrar como se hace la construcción de los circuitos impresos, previamente diseñados y verificando su funcionamiento adecuadamente con lo requerido en Proto Bord para las fuentes conmutables que se necesitarán en el programador de PAL's.

Se pretende hacer un antecedente de la manera como se logro realizar los impresos de doble cara, sin hacer referencia a los fracasos, sino que solo al resultado exitoso de fabricación de impresos en tabletas de baño de cobre.

Además se presenta un enfoque del Software utilizado **("EAGLE")** para la generación de los circuitos impresos. Ya que no se encuentra documentación disponible para la elaboración de impresos con la utilización de este Software. Que muy poca gente lo puede utilizar para realizar un trabajo de impresos de doble cara con calidad, confiabilidad, y garantía.

Objetivos

1. Diseño de la interface de control para el programador de PALs
2. Diseño de las interfaces para generar los diferentes voltajes para la programación del PALs. (fuentes conmutables)
3. Presentar una alternativa para el diseño de circuitos impresos de doble cara con el Software Eagle.

2.1.6.1 Descripción del funcionamiento de las señales del sistema de bus de la PC IBM

SD7 - SD0: Data system (E/S)

Son los ocho bits menos significativos del bus de datos del sistema. En transferencias de 8 bits son los únicos utilizados, SD0 es el bit LSB y SD7 es el bit MSB

SA19 - SA0: Address system (E/S)

Representa el bus de direcciones de la computadora y son usados para direccionar memoria y dispositivos de Entrada/Salida dentro del sistema. SA19-SA0 permiten direccionar hasta 1Mbyte de memoria RAM. SA0 es el bit menos significativo y SA19 es el bit más significativo.

CLK: Clock (S)

Es la señal de reloj del sistema; debe usarse solamente para efectos de sincronización.

RESET DRV: Restet Drive (S)

Inicializa la lógica del sistema al encender la máquina o durante un bajo voltaje. Esta señal es sincronizada en el flanco de caída de la señal OSC.

BALE: Buffered Address Latch Enable (S)

Esta señal proviene del controlador del bus y es utilizada para enclavar direcciones válidas del microprocesador. Se utiliza como indicador de direcciones válidas del DMA ó del CPU.

I/O CH CK: I/O channel check (E)

Esta es una señal baja activa y proporciona al sistema información de error de paridad en memoria ó de dispositivos en los canales de entrada/salida.

IRQ7 - IRQ2: Interrupt Request (E)

Son señales de solicitud de interrupción, IRQ7 es la de menor prioridad e IRQ2 es la de mayor.

IOR: Input/Output read (E/S)

Esta señal le indica a un periférico que maneje sus datos a través del bus de datos. La señal es controlada por el microprocesador, el controlador de DMA o similares, presentes en el canal entrada/salida. Es activa en bajo.

IOW: Imputa/Output Write (E/S)

Esta señal le indica a un periférico que lea los datos presentes en el bus de datos. Es controlada en igual forma que IOR. Es activa en bajo.

SMEMR: (S)

Esta señal le indica a dispositivos de memoria, manejar sus datos a través del bus de datos. Es controlada por la CPU o el controlador de DMA. Es activa solo para rangos menores a 1 MByte de memoria.

SMEMW: (S)

Esta señal le indica a los dispositivos de memoria almacenar los datos presentes en el bus de datos. Es controlada por la CPU o por el controlador de DMA. Es activa solo para rangos menores a 1 Mbyte de memoria.

DRQ1, DRQ2, DRQ3: DMA request (E)

Estas líneas de entradas son asincronas y son usadas para solicitud de transferencia de DMA. DRQ1 es la de mayor prioridad y DRQ3 tiene la prioridad más baja; esta señal se mantiene en alto hasta que el correspondiente DAK se activa a un nivel bajo. Note que DRQ0 no se encuentra disponible sobre el bus, esta es utilizada para refrescar la memoria dinámica del sistema.

4. Presentar un procedimiento de construcción de tarjetas impresas por métodos artesanales, cumpliendo con resultados de confiabilidad, seguridad, y garantía en el diseño.

2.0 Fuentes Programables y sus Controles

Para programar un PAL se necesitan diversos niveles de voltaje (refiérase al anexo A y B para ver gráficas de programación), tales como +5v, +10.5v, +21v y 0v, así también, se requiere un estado de alta impedancia (10 kΩ a 5.0 v para Texas Instruments y para Advanced Micro Device 2kΩ a 5.0 v) en el proceso de programación, para los siguientes fabricantes Texas Instruments y Advanced Micro Devices,

Para obtener esos niveles de voltaje y estados de alta impedancia, se utilizan fuentes conmutables, que son arreglos de compuertas lógicas, latches (Flip-flop tipo D) con resistores, diodos y transistores que, dependiendo de los niveles lógicos colocados a sus entradas (que pueden ser dos o tres según el tipo), así será su nivel de salida, para controlar las entradas de las fuentes conmutables, a continuación se presentan las diferentes modalidades de fuentes conmutables, desarrolladas en este proyecto.

2.1 Circuito de Conmutación de Voltajes.

En este apartado se presentan los circuitos desarrollados con los cuales se generan los diferentes niveles de voltajes, y estados de alta impedancia para ambos fabricantes.

Inicialmente se muestra la vista de los Ics de fabricación Texas en los pines PI 2 al PI11 y el pin de VCC se necesitan niveles de 5.0 v y 10.5 v, y en los pines PO14 al PO23, y PI1 se requieren los niveles de 10.5 y alta impedancia (10K a 5.0v). Y similarmente son las distribuciones para los Ics de AMD, con la diferencia que en los pines del PI2 al PI9, y pin Vcc se necesitan solamente los niveles de 0.0v y 5.0 v, mientras que en los pines del PO12 al PO19 se necesitan los niveles de 21.0v y alta impedancia (2k a 5.0v) , además en los pines PI1 y PI11 sean los niveles de 12.0 v como máximos.

Iniciamos con el circuito de la etapa de lectura, que se utilizará para verificar si sé la programación de los fusibles fue exitosamente realizada.

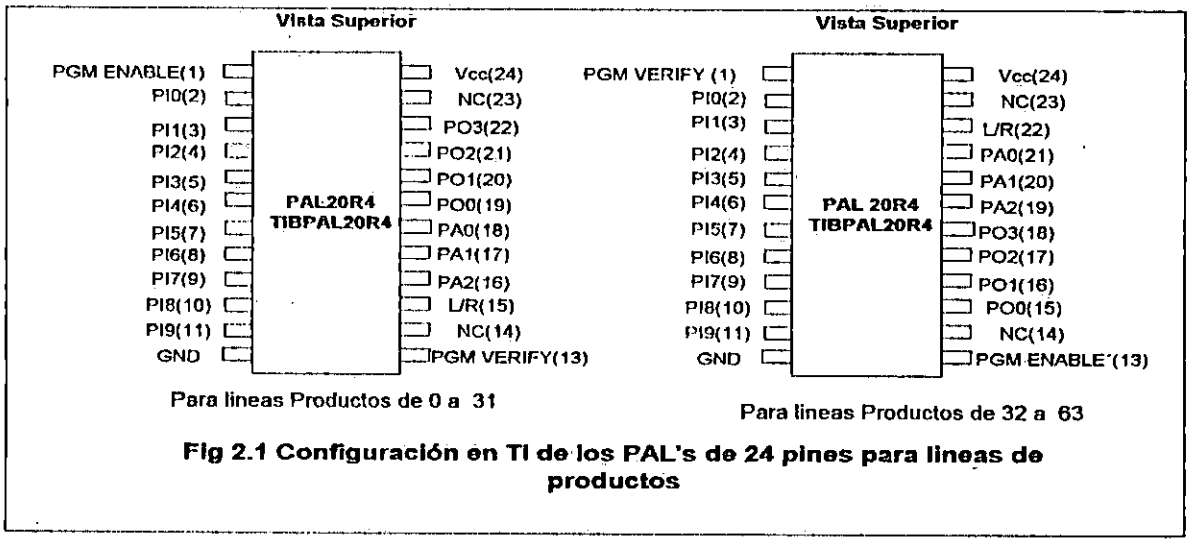


Fig 2.1 Configuración en TI de los PAL's de 24 pines para lineas de productos

2.1.1 Circuito de Lectura Tipo A.

El circuito de conmutación de tipo A utilizado para realizar la lectura de la salida de los pines PO, PA Y L/R, siempre y cuando las entradas en los pines PI se mantengan para poder realizar la verificación de que se rompió o no el fusible seleccionado.

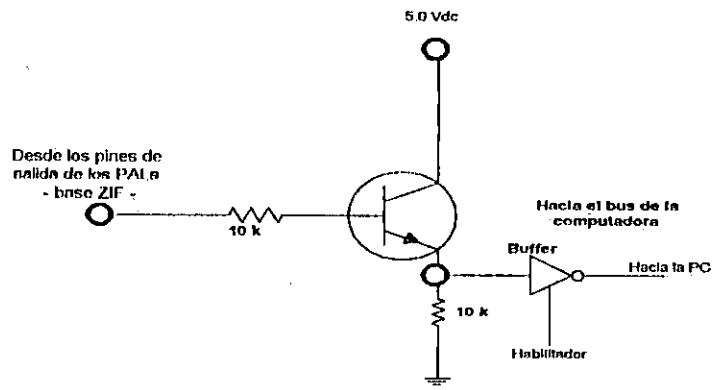


Figura 2.2 Circuito de Conmutación de Voltajes Tipo A.

En la fig. 2.2 se observa la configuración del circuito tipo A que sirve para realizar la comunicación entre las salidas del PALs, y el bus de datos de la computadora y así realizar procesar la información captada.

2.1.1.1 Descripción del funcionamiento del circuito de Lectura Tipo A.

El funcionamiento del circuito tipo A se describe de la siguiente manera:

Cuando se selecciona un fusible en el arreglo, se necesita colocar una combinación de voltaje en los pines PI del PALs y colocar el PO, PA, y L/R en el estado de alta impedancia, a excepción de aquel pin al que se le coloca 10.5 v para Texas Instruments y 21.0 v para Advanced Micro Device. Posteriormente se retiran los niveles de voltajes de todos los pines PO, PA, y L/R para leer si el fusible ha sido programado o quemado exitosamente. Es en ese momento que se genera un nivel de voltaje generado por el PALs en programación en el pin de salida seleccionado, siempre y cuando se mantengan los niveles de entrada en los pines PI del PALs.

Esa lectura se realiza a través de la resistencia de 10K que coloca un nivel de voltaje en la base del transistor 2N2222 y conduce colocando el nivel de voltaje de 5.0v en la entrada del buffer 74LS244, en espera de que sea habilitado por la señal de control desde la PC. Para colocar el nivel de voltaje en uno de las 8 líneas del bus de datos de la PC IBM. Y así se realice el procesamiento de la información con las herramientas de Software.

2.1.2Circuito de Conmutación para los pines PI2 al PI11 Tipo B.

El circuito mostrado en la fig.2.3 se observa que están constituidos por diodos, resistencias, compuertas 74LS06, y Latch. Todos en conjunto constituyen el circuito tipo B, descrito a continuación.

Los circuitos integrados 74LS373 son los Latch que recibirán las palabras binarias desde la computadora el bus de datos y además recibirán una señal de control desde la PC. La cual definirá cual de todos los Latch deberá de ser habilitado para tomar la palabra que en ese momento se encuentre en el bus de datos. Es en ese momento cuando la combinación recibida aparecerá en la entrada de la compuerta de los Ics 74LS06, esto producirá un valor de terminado de voltaje de 5.0v ó 10.5v de salida en el pin de la base ZIF, del programador de PALs.

En ese momento se produce una salida determinada en función de la combinación en la entrada de los Ics 74LS06 son utilizados como parte sobresaliente de las fuentes conmutables de 5.0v a 10.5 v por ser salidas de colector abierto, los cuales se les coloca un conjunto de diodo-resistencia o solamente un resistor a la salida para generar la conmutación de los voltajes en la salida ya que los 74LS06 pueden soportar hasta 30 voltios de salida perfectamente sin sufrir ningún daño ellos ni el diseño.

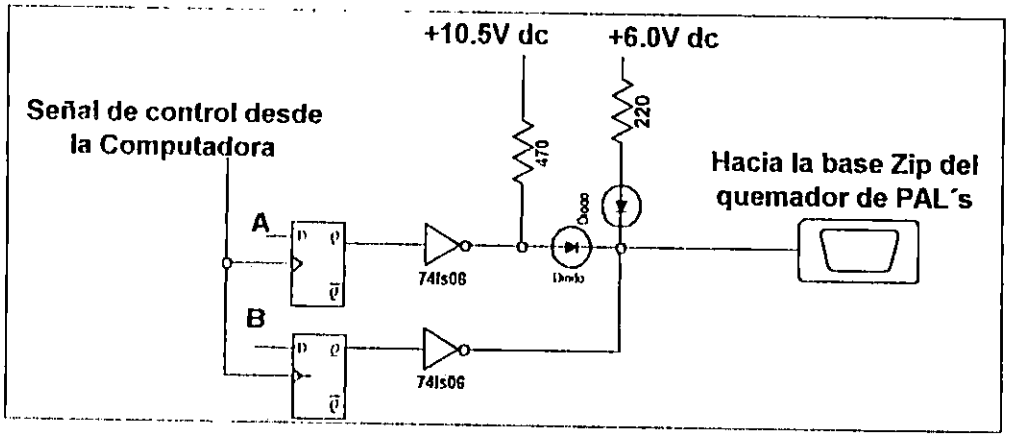


Figura 2.3 Circuito de Conmutación de Voltajes Tipo B.

El circuito tipo B es para los pines del PI2, PI4,PI5,PI6,PI7, PI8, PI9, PI10, y el P11 en el programador. Ya que en estos pines según gráficas de programación de los fabricantes de Texas Instruments los niveles de voltajes son de 10.5 v, 5.0 v y Advanced Micro Device se necesitan, 0v y 5, 0v.
Este tipo de circuito no se define el pin PI3 por carecer del nivel de 21.0v que en este pin se necesita cuando se programan PALs de 20 pines de fabricación TEXAS INSTRUMENTS. Lo cual si se hace en el circuito tipo C que a continuación se define.

2.1.2.1 Descripción del circuito tipo B.

Cuando las entradas "a" y "b" están en 0, **D1** está polarizado en directa y **D2** en inversa, con lo que el voltaje de 10.5v se refleja en la salida **Vo** (ver anexos A y B para curvas de programación según fabricantes).

Si **a=1 y b= 0, D1** se polariza en inversa(obsérvese que tiene un mayor potencial que el ánodo), **D2** en directa, con lo que el voltaje a la salida **Vo** se aproxima a 5.0v.
Si a=b=1 a la salida **Vo=0.0v.** como se puede observar en la tabla 2.1.

Tabla 2.1 Conmutación de Voltajes del circuito tipo B

A	B	Vo	DESCRIPCION
0	0	10.5	D1 DIRECTA, D2 INVERSA
0	1	5	D1 INVERSA, D2 DIRECTA
1	1	0	SALIDA DE LOS 74LS06 EN CERO

2.1.3 Circuito de Conmutación para los pines PI3, PI1, y PI13 Tipo C.

E l circuito de conmutación tipo C, es el utilizado en los pines siguientes:

En los pines PI1, y POI3 de la base ZIP se necesitan los niveles de voltajes de 21.0v, 10.5v 5.0v y 0.0v en el proceso de programación, cuando se estén programando los PAL´s de 20 y 24 pines, los cuales se generan con el circuito tipo C. Solo que el pin PI1 para los IC de 24 pines es el pin PI1 de la base ZIF, pero cuando se esta programando los PAL´s de 20 pines, el pin PI1 se encuentra ubicado en el pin PI3 de la base ZIP.

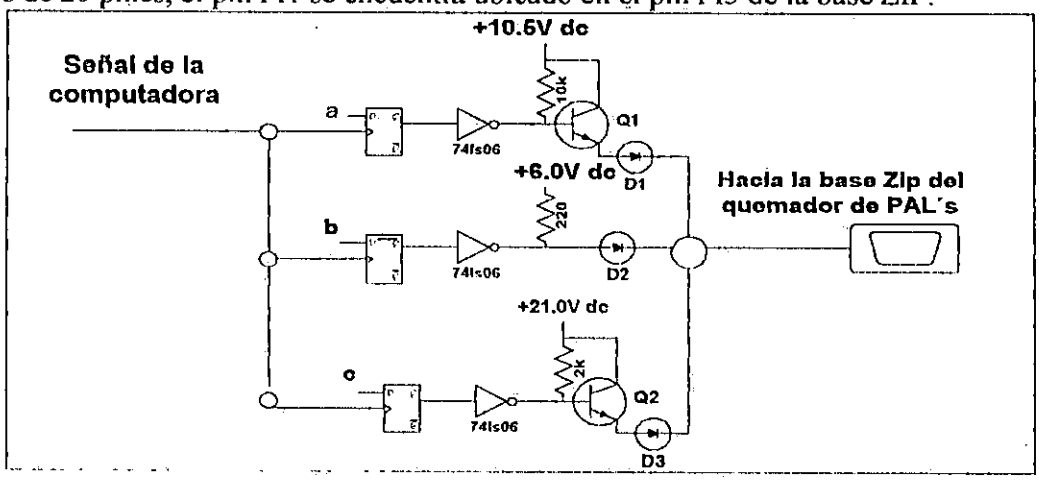


Figura 2.4 Circuito de Conmutación de Voltajes Tipo C.

2.1.3.1 Funcionamiento del Circuito de Conmutación para los pines PI3, PI1, y PI13 Tipo C.

En el circuito tipo C, cuando están en a=b=c=0, Q1, y Q2 se encuentran en saturación y todos los diodos D1=D2=D3 están en directa y se produce superposición de voltajes a la salida dando como resultado Vo =21.0v. (transistores -NPN)

Cuando a=b=0 y c=1 se encuentra Q2 en saturación y Q1 en corte y polariza a D1 y D2 en inversa también esta polarizado D3 en directa y se produce una salida de a la salida Vo=10.5v.

Cuando a=c=1 y b=0, Q1=Q2 se encuentran en corte no conducen y el diodo D3 se encuentra polarizado en directa y la salida es Vo=5.0V. Mientras que cuando a=b=c=1 la salida Vo =0.0v porque los Q1=Q2 se encuentran en corte y D3 se encuentra en polarizado en reversa.

En la tabla 2.2 se muestra la salida de voltaje Vo del circuito tipo C, según la combinación de entrada de las inversores 74LS06.

Tabla 2.2 Conmutación de Voltajes del circuito tipo C

a	b	c	Vo	DESCRIPCION
0	0	0	21.0	D1 =D2=D3 en DIRECTA Q1=Q2 en saturación
0	0	1	10.5	D1 =D2 en DIRECTA y D3 en REVERSA Q1 en saturación, Q2 corte
1	0	1	5.0	Q1=Q2 en corte y D3 en directa
1	1	1	0.0	Q1=Q2 en corte y D3 en reversa

2.1.4 Circuito de Conmutación para los pines Vcc, PO, PA, y L/R2 Tipo D.

El circuito tipo D, es similar al circuito descrito en el apartado anterior con la diferencia de que contiene una etapa más y los estado de alta impedancia, para los fabricantes Texas Instruments y Advanced Micro Device, y se utiliza en los pines de la base ZIF de programador:

1. El pin de Vcc (pin 24 para 24 pines y pin 22 para 20 pines) se necesitan solo los siguientes niveles de voltajes 10.5,5.0 v, y 0.0v.
2. Los pines 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, en todos ellos en la base ZIF se necesitan los niveles de voltajes de 10.5 para Texas Instruments y alta impedancia de 10k a 5.0v, y para Advanced Micro Device 21.0 v y alta impedancia de 2.0K a 5.0v.

Estos diversos estados se logran con el circuito tipo D.

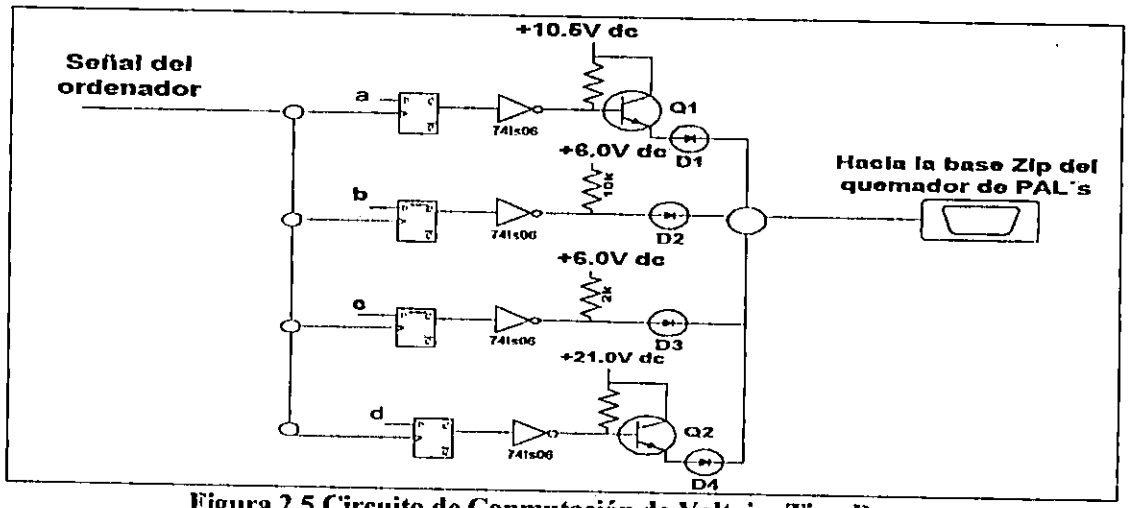


Figura 2.5 Circuito de Conmutación de Voltajes Tipo D.

El funcionamiento de este circuito tipo D es similar a explicado en el circuito tipo C, con diferencia de que cuenta con cuatro bits de control, en la tabla 2.3 se expone el funcionamiento del circuito de conmutación tipo D.

Tabla 2.3 Conmutación de Voltajes del circuito tipo D

A	b	c	D	Vo	DESCRIPCION
0	1	1	1	10.5	Q1 en corte D1 en directa D2=D3 =D4 en reversa Q2 en saturación
1	0	1	1	Z	Alta impedancia para Texas Instruments D1 =D3=D4 en reversa y D2 en directa
1	1	0	1	Z	Alta impedancia AMD D1=D2=D4= reversa Q1=Q2 en saturación y D3 en directa
1	1	1	0	21.0	Q1en saturación D1=D2=D3 en reversa Q2 en corte D4 en directa

2.1.5 Arquitectura del circuito interface

Este circuito esta basado en el chip 8255 (PPI). El lector puede referirse a las referencias bibliográficas citadas en este capitulo para estudiar el funcionamiento de este IC.

Es indispensable conocer la arquitectura de la tarjeta CPAL y las tarjetas de switcheos para las fuentes conmutables para realizar el diseño del Software que la controla adecuadamente. En este apartado se resume el diseño de esta tarjeta.

2.1.6 El sistema de bus de las computadoras personales IBM o compatibles

El sistema de bus IBM PC es una extensión del bus del microprocesador 8088 de la Intel, este tiene señales adicionales para soportar acceso directo a memoria (DMA), interrupciones y otras utilidades. Todas las señales sobre este bus responden a niveles TTL (lógica transistor - transistor); la fig. 2.6 muestra la asignación de pines para este bus.

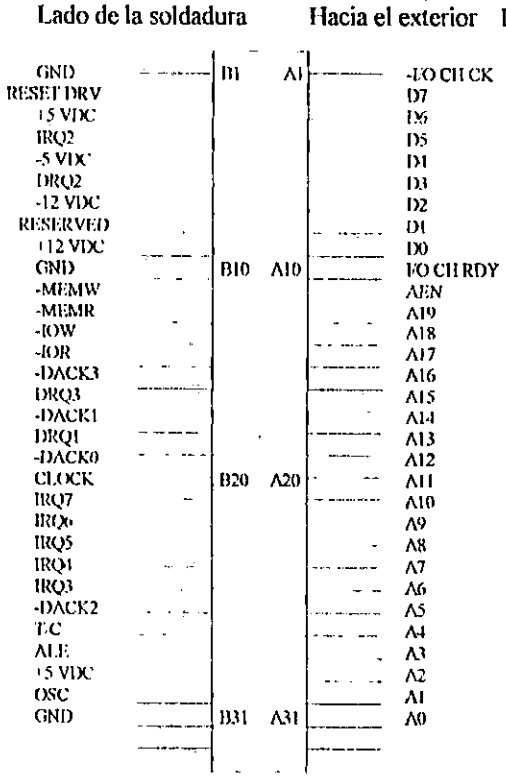


Fig. 2.6 Asignación de pines del sistema de bus de las PC IBM

DACK1, DAK2, DACK3: DMA acknowledge signals (S)

Estas líneas están activas en bajo, y sirven para accesar solicitudes de DMA.

AEN: Address enable (S)

Cuando esta señal se activa (alto), el CPU cede el control sobre el bus de direcciones y el bus de datos al controlador de DMA, permitiendo la transferencia de DMA.

REFRESH: (E/S)

Esta señal indica un ciclo de refrescamiento, es activa en bajo y puede controlar el microprocesador en el canal de entrada/salida.

T/C: Terminal count (S)

Proporciona un pulso alto cuando es alcanzada la cuenta final por cualquier canal de DMA.

OSC: Oscillator (S)

Señal de reloj no sincronizada de 14.311818 Mhz y 50% de duty cycle.

2.1.7 Mapa para dispositivos entrada / salida en una PC IBM y Compatibles

En orden a generar las direcciones correctas de selección de nuestra tarjeta, debemos estudiar el direccionamiento a puertos de I/O y el mapa de asignaciones en las PC IBM. El diseño de las PC es tal que provee 10 bits para direccionar puertos (AS9 - A0), para un total de 1024 direcciones de puertos. El mapa de direcciones de puertos I/O está dividido en dos partes. Las primeras 512 direcciones (0000H-01FFH) son asignadas a los componentes de la tarjeta madre del sistema. El espacio de direcciones desde 0200H a la 03FFH, están disponibles para direccionar tarjetas utilitarias sobre los slots. La fig. 2.6 muestra que las 512 direcciones de puertos más significativas son dispuestas para tarjetas para expansión del sistema. Los puertos disponibles para una tarjeta prototipo son desde la dirección 300H hasta la 031FH, lo cual corresponde solo a 32 direcciones de puertos.

2.2 Elementos asociados a buses

2.2.1 Amplificadores de bus

Son circuitos que permiten expandir la carga admisible ("fan-out") de los buses, respetando la polaridad de las señales o invirtiéndola. Las salidas normalmente son de tres estados, es decir que una determinada señal de control cuando es activa permite que las salidas adopten los valores binarios "0" o "1" con baja impedancia, dependiendo de los estados de las entradas; pero cuando la citada señal de control se desactiva, las salidas pasan a un tercer estado equivalente a la desconexión, presentando una elevada impedancia en serie, al margen de los estados lógicos de las entradas.

Ejemplos típicos pueden ser los Ics 74LS240, 241 y 244 (vea anexos), amplificadores de bus de 8 bits de 3 estados que requieren una corriente de entrada máxima de 0.2 mA presentando características de histerisis para mejorar la susceptibilidad a ruidos, y que a su

salida pueden entregar hasta 24 mA (amptos para hasta 15 entradas TTL o 60 TTL LS). La distinción entre estos 3 circuitos está en el carácter inversor o no de sus entradas, de tal manera que el 240 invierte el bus y activa las salidas cuando las entradas de control son "0"; el 241 no invierte el bus y activa las salidas con señal de control "1"; y el 244 no invierte el bus y activa las salidas con señal de control "0".

Fig. 2.6 Mapa de puertos de I/O en las PC IBM

Rango hexadecimal	Asignación	
000-00F	Chip DMA 8237A-5	
020-021	Chip de interrupción 8259A	
040-043	Temporizador 8253-5	
060-063	PPI 8255A-5	Asignados para los
080-083	Registros de paginas DMA	Componentes sobre
0Ax	Registro de mascara NMI	la tarjeta madre
0Cx	Reservado	
0Ex	Reservado	
100-1FF	No utilizable	
200-20F	Control de juegos	
210-217	Unidad de expansión	
220-24F	Reservado	
278-27F	Reservado	
2F0-2F7	Reservado	
2F8-2FF	Comunicación asincrona (2)	
300-31F	TARJETAS PROTOTIPOS	Asignados para las
320-32F	Disco fijo	tarjetas en los canales-
378-37F	Impresor	
380-38C	Comunicaciones SDLC	
380-389	Comunicación sincrona binaria (2)	
3A0-3A9	Comunicación sincrona binaria (1)	
380-3BF	display IBM monocromo/ impresor	
3C0-3CF	Reservado	
3D0-3DF	Color / Graficos	
3E0-3F7	Reservados	
3F0-3F7	Discos	
3F8-3FF	Comunicación asincrona (1)	

2.2.2 Transceptores

Los circuitos transceptores tienen una cierta similitud con los amplificadores de bus, de los que se distinguen por el hecho de poder amplificar las señales bidireccionalmente, lo que simplifica la conexión entre la mayoría de microprocesadores cuyo bus de datos es bidireccional y periféricos con características de dialogo.

Estos circuitos presentan generalmente las salidas con características de 3 estados y disponen de señales de control que permiten habilitar la amplificación en uno u otro sentido, bien sea por señales de control de salida, independiente para cada sentido y otras que autoriza las salidas.

Ejemplos típicos son los IC 74LS242 y 243 (vea anexos), que son transceptores de 4 bits con autorización de salida en un sentido por la señal GBA en "1" como mínimo de 3.65V. Presentan la peculiaridad de tener diferenciadas entrada y salida por un extremo, de tal modo que con conexión externa actúan como transceptores, y sin conexión entre ambas se convierten en doble amplificador de bus seleccionable.

Como muestra de transceptor de bus de 8 bits se puede tomar el IC 74LS245 (vea anexos) en el que el control de salidas se obtiene por la combinación de una señal de autorización y otra de determinación de la dirección de la señal en el bus.

2.3 Interfaz programable 8255A

El 8255A es un circuito programable de entradas/salidas de utilización general diseñado originalmente para ser acoplado a los microprocesadores INTEL 8080 y 8085. Dispone 24 terminales de E/S que pueden ser programados individualmente en dos grupos de doce y ser empleados en tres distintas modalidades principales. En la primera modalidad (Modo 0), cada grupo de doce terminales de entradas/salidas puede ser programado en grupos de cuatro para actuar como entradas o como salidas. En modo 1 (la segunda modalidad), cada grupo puede ser programado para disponer de 8 líneas de entrada o salida, mientras que tres de los restantes terminales, por cada grupo, son utilizados para sincronización de entradas/salidas sea por esperas o por interrupciones. La tercera modalidad de operación (Mode 2) adopta la configuración de bus bidireccional, empleando 8 líneas para datos y 5 para sincronización (tomando una del otro grupo).

La configuración funcional de cada puerto es programada por el sistema. En esencia el microprocesador emite una palabra de control al 8255, conteniendo información tal como: modalidad, activación de bits, restauración de bits, etc., con el fin de inicializar el circuito.

El registro de control solamente puede ser escrito, es decir que no es posible leer su estado en un momento determinado desde el microprocesador.

2.4 Propuesta para el circuito de interface de control CPAL

El circuito propuesto está basado en el chip 8255A (PPI) descrito anteriormente, en esta sección se describe discutimos en parte los temas anteriores, pues estos son considerados temas indispensables a conocer para el diseño efectivo de una tarjeta prototipo para las PC IBM o compatibles.

2.4.1 Decodificación de la tarjeta prototipo

En el diseño de la tarjeta se escogió utilizar las direcciones 031H-031FH las cuales se encuentran en el rango de direcciones validas para tarjetas prototipo.
Al encender la máquina el 8255A sé inicializa (por medio del pin RESET conectado a la línea RESET de la ranura de expansión) y sus tres puertos quedan configurados como puertos de entrada. Para configurar estos puertos se utiliza la palabra de control que consta de 8 bits y que son colocados en el registro de control, La interface solo responde a 4 direcciones, correspondientes a los puertos A, B, C y al registro de control de solo escritura.

La dirección del flujo de datos en la PPI, es controlada por los pines RD y WR, los que se conectan a las líneas IOR e IOW de las ranuras de expansión respectivamente.

La PPI posee además los pines de dirección A0 y A1 los cuales tienen la función de seleccionar entre uno de los tres puertos y el registro de control, la tabla 2. 4 muestra la función de los pines A1A0 de la PPI.

A0	A1	FUNCIÓN
0	0	Puerto A
0	1	Puerto B
1	0	Puerto C
1	1	Registro de control

Tabla 2.4 Direccionamiento de puertos y registro de control.

Para estos pines y el pin CS (chip select) se hace necesario construir un decodificador de direcciones, que habilite al chip con las direcciones correspondientes, dentro del rango I/O del mapa de memoria de la PC, la tabla 2.5 muestra las direcciones de los 3 puertos y el registro de control.

Tabla 2.5. Dirección de los tres puertos y el registro de control

	SA9	SA8	SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0	HEX
PUERTO A	1	1	0	0	0	1	1	1	0	0	31C
PUERTO B	1	1	0	0	0	1	1	1	0	1	31D
PUERTO C	1	1	0	0	0	1	1	1	1	0	31E
PALABRA DE CONTROL	1	1	0	0	0	1	1	1	1	1	31F

El decodificador de direcciones para esta interface se muestra en la figura 2.7

2.4.2 Operación de la PPI en el modo 0

A este modo es llamado entrada/salida elemental, en esta modalidad se configura a los tres puertos solamente como entrada/salida, presentando las siguientes características:

- Los puertos A y B de 8 bits cada uno y el puerto C dividido en dos grupos de 4 bits cada uno, los cuales pueden ser entrada o salida.
- Cualquier puerto puede ser entrada o salida
- Las salidas son enclavadas
- Existen 16 formas de configurar este modo.

En la tabla 2.5 se muestran las 16 posibilidades de configuración en el modo 0; es de tener muy en cuenta que cada vez que sé reconfiguren los puertos con este byte de control, los pines que se configuraron como salida son recolocados a cero.

Tabla 2.6. Configuración de puertos en el modo 0

Byte de control	Puerto A	Puerto B	4 MSB Puerto C	4 LSB Puerto C
80	salida	Salida	Salida	salida
81	salida	Salida	Salida	entrada
82	salida	Entrada	Salida	salida
83	salida	entrada	Salida	entrada
88	salida	salida	Entrada	Salida
89	salida	salida	Entrada	Entrada
8A	salida	Entrada	Entrada	Salida
8B	salida	Entrada	Entrada	Entrada
90	entrada	Salida	Salida	Salida
91	entrada	Salida	Salida	Entrada
92	entrada	Entrada	Salida	Salida
93	entrada	Entrada	Salida	Entrada
98	entrada	Salida	Entrada	Salida
99	entrada	Salida	Entrada	Entrada
9A	entrada	Entrada	Entrada	Salida
9B	entrada	Entrada	entrada	Entrada

En este decodificador observamos que la línea AEN de la ranura de expansión (A11) se utiliza para evitar que la interface sea habilitada durante una transferencia DMA. La línea -IOR se utiliza para manejar el sentido de flujo de datos, cuando esta línea es activa significa que el microprocesador está ejecutando una instrucción de lectura (IN) en este momento el sentido del transceptor es tal que las líneas A1-A8 son salidas y las líneas B1-B8 son entradas permitiendo de este modo la operación de lectura correctamente.

2.4.3Asignación de líneas de puertos PPI a las señales de CONTROL

En esta sección determinaremos las configuraciones de los puertos PPI, para asignarle a cada señal del bus su(s) correspondiente(s) líneas de los puertos.

- Para las líneas de datos (D1-D8), se utilizará el puerto A del 8255.
- Para las líneas de control se utilizan los 7 bits del puerto C
- Para las líneas de control se utilizan los 8 bits del puerto B.

Debido a que el bus de datos es bidireccional, los puertos de la PPI deben de configurarse cada vez que sea necesario. De la tabla 2.5 que describe el sentido de dirección de los puertos.

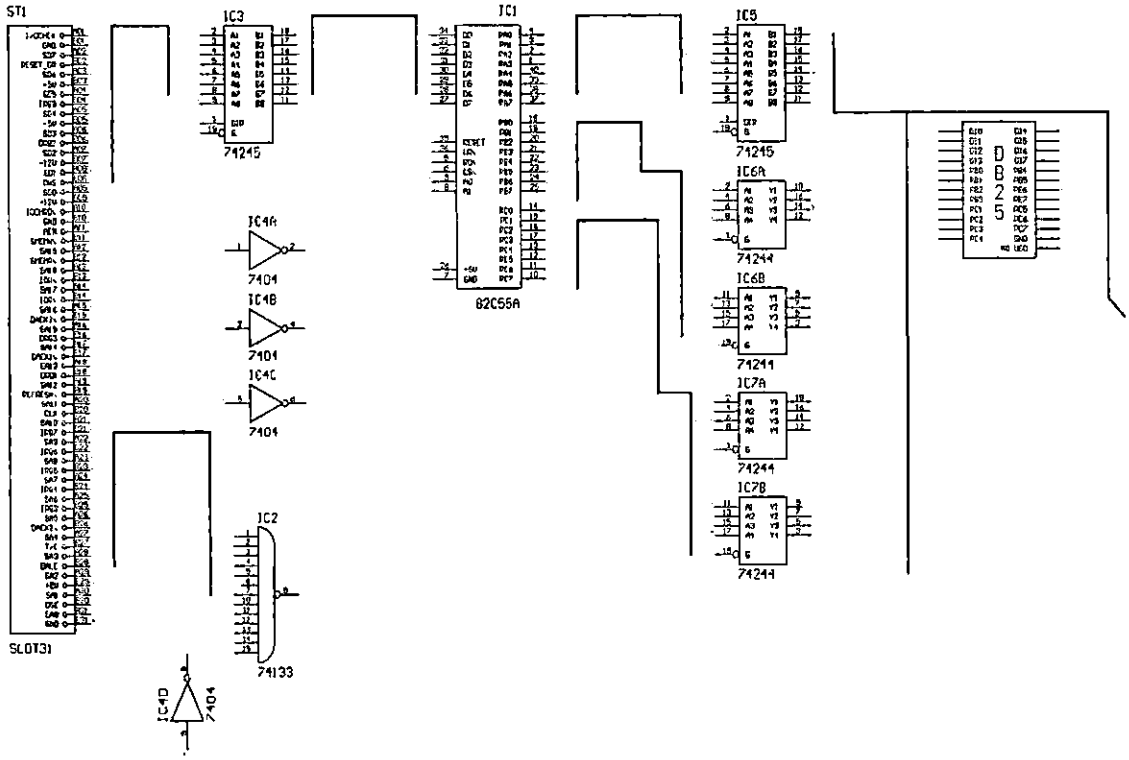


Fig. 2.7. Diagrama completo de CPAL Tarjeta de Control y su Decodificador

D0..7
D1..7
D2..7
BUSB0..7
IC1
IC2
IC3
74273
7406

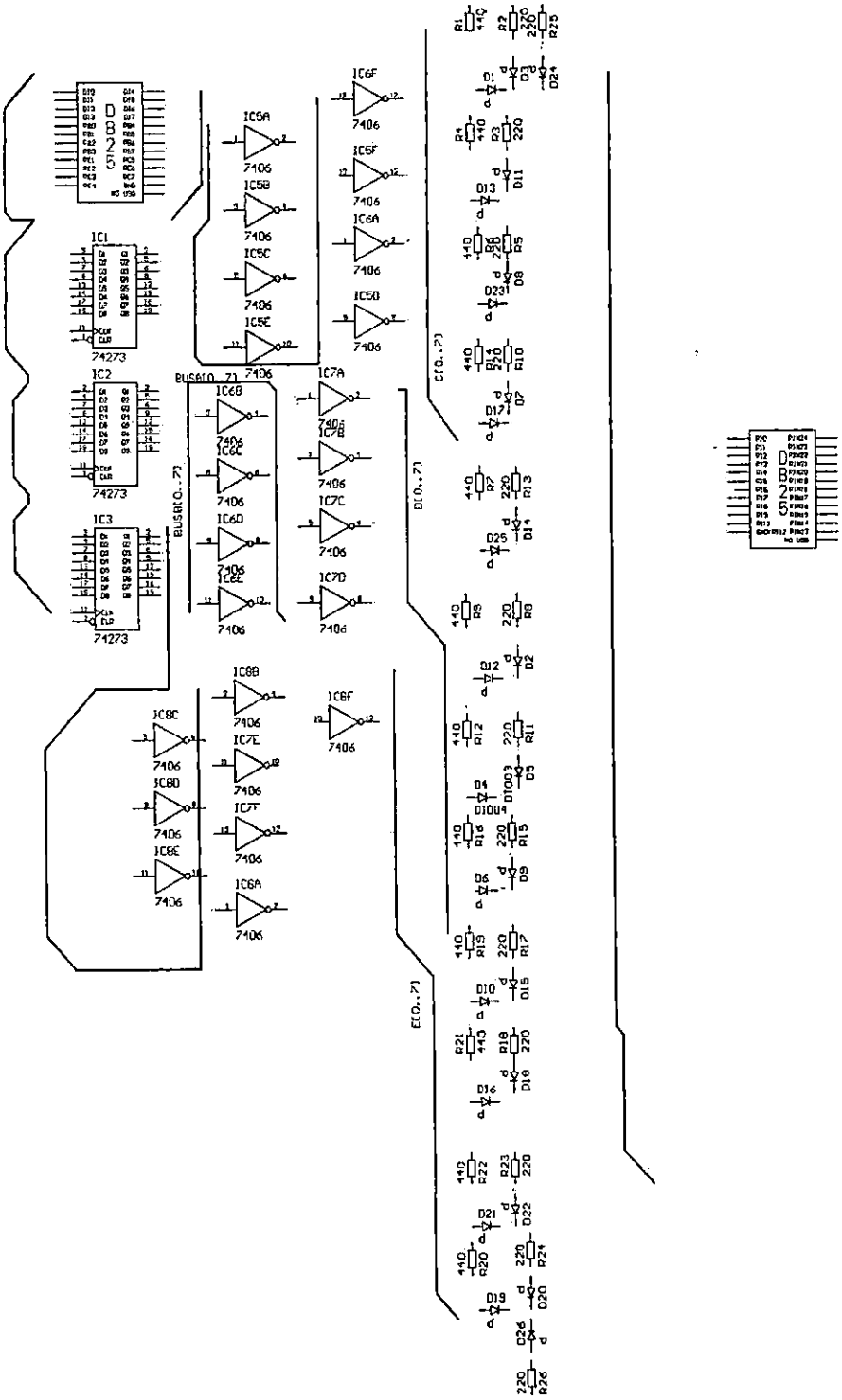


Fig. 2.9. Diagrama completo de fuentes conmutables Pines 13 al 24

2.5 Construcción del circuito impreso para la tarjeta CPAL

El circuito impreso fue elaborado en/por el paquete EAGLE, este es un software CAD (Diseño Auxiliado por Computadora) para ingeniería Eléctrica o carreras a fines. Este software es capaz de generar el circuito impreso a partir del circuito esquemático. No es objeto de esta sección describir a fondo este paquete, así se presenta en forma breve la utilización de las principales opciones que este paquete soporta, para la creación de un circuito impreso a partir de un circuito esquemático.

Aunque no es necesario tener el circuito esquemático para la creación del circuito impreso, la obtención de este a partir del esquema nos ahorra tiempo y trabajo.

2.5.1 Descripción de las funciones principales del software EAGLE

La siguiente tabla describe las funciones principales y muestra un ejemplo para estas.

Tabla 2.7. Funciones principales del software CAD EAGLE.

EJEMPLO	DESCRIPCION
Add 7400	Añade el símbolo para una compuerta NAND, al esquema actual, para ello debe tenerse en uso la librería 744xx, esto es en ambiente esquema (Schematic).
Copy	Duplica el objeto seleccionado con el MOUSE; en el ambiente esquema los símbolos no pueden ser duplicados.
Delete	Borra el objeto seleccionado con el MOUSE;
Group	Selecciona un conjunto de objetos, al encerrar estos en un recuadro con el MOUSE.
Junction	Une dos o más líneas seleccionadas con el MOUSE en el ambiente esquema. Puede usarse también para unir un pin y una línea de bus cuando esta línea no tiene el mismo nombre del pin.
Show	Ilumina y describe el objeto seleccionado, use para observar si las conexiones de líneas y pines están soldados.
Split	Fractura una línea o un alambre.
Text CPAL	Adiciona el texto CPAL en el punto seleccionado con el Mouse.
Bus	Use con el MOUSE para crear un bus en el ambiente esquema.
Label	Use para obtener el nombre de un bus o línea en forma de viñeta el ambiente esquema.
Move	Use con el MOUSE para mover el objeto seleccionado
Name	Use para cambiarle el nombre a un elemento (ej. E$1 por 7400)
Value	Use seleccionando el objeto al cual quiere cambiar su valor (ej. E$1 por IC1).
Window fit	Use para ver en pantalla todos los elementos colocados, puede utilizar un numero como factor de proporción para aumentar o disminuir el esquema en pantalla.

Continuación tabla 2.7.

Write CPAL	Guarda el archivo CPAL que esta en pantalla en disco, la extensión de este archivo varía de acuerdo al ambiente en que nos encontremos.
Edit CPAL	Carga el archivo CPAL que esta en disco a pantalla. Si el archivo EJGPIB no existe se tomará como un archivo nuevo. El archivo CPAL se tomará de acuerdo al ambiente actual del paquete, estos son: Board (ext. BRD), Esquema (ext. SCH).
OPEN MYLIB	Crea o abre la librería llamada MYLIB.
Close	Cierra la librería en uso, y retorna al ambiente por default.
Cut	Copia él (los) objeto(s) seleccionado(s) en el buffer de cortar.
Paste	Pega en el ambiente actual los objetos encontrados en buffer de cortar.
Undo	Permite restablecer acciones efectuadas en el archivo, si UNDO está activo (SET UNDO LOG ON)
Redo	Ejecuta la acción contraria a UNDO.
Dir	Muestra en pantalla los nombres de archivos en el directorio actual si nos encontramos en el ambiente BOARD o SCHEMATIC, y nos muestra los nombres de los objetos de una librería si esta se encuentra abierta.
Signal	Une dos pads en el ambiente BOARD.
Route	En ruta con alambre una señal (signal).
Vía	Coloca una vía en el ambiente BOARD, use para unir un alambre en la capa de componentes con la de soldadura.
Ratsnest	Use esta opción en ambiente BOARD para interconectar pads con signals correspondientes a las líneas que se encuentran en el archivo de esquemas.
Connect	Use cuando cree o modifique un dispositivo para interconectar los pads correspondientes a la macro asignada con los pines de este.
Package	Use cuando cree o modifique un dispositivo, para asignar una macro a un dispositivo.
Prefix	Use cuando cree o modifique un dispositivo, para asignar un prefijo a un dispositivo.
Pin	Use cuando cree o modifique un símbolo que no este en uso, para colocar un pin de conexión para el símbolo.
Arc	Use para crear un arco en la capa actual (layer).
Rect	Use para crear un rectangulo en la capa actual (layer).
Circle	Use para crear un circulo en la capa actual (layer).
Erc	Use en el ambiente esquema para verificar los errores que el esquema actual tenga. Esta función crea un archivo texto con el mismo nombre del esquema que se edita, con extensión ERC.
DRC	Use en el ambiente BOARD para verificar los errores que el impreso tenga.
Errors	Use en el ambiente BOARD después de haber ejecutado DRC, para observar los puntos que contienen error,
Pad	Use cuando cree o modifique una macro,

Continuación tabla 2.10.

Net	Use con el MOUSE para interconectar pines entre dispositivos.
Auto	Este comando se utiliza en el ambiente BOARD para en rutar señales (signals) de forma automática.
Board	Utilice este mandato para crear el circuito impreso a partir del circuito esquemático completo.
Change	Este comando tiene varias opciones, están cambian y funcionan diferente según el ambiente en que nos encontremos, se usa para hacer cambios de parámetros sobre componentes ya colocados o por colocar. Un ejemplo es: Change Size 0.4, cambia el grosor de los alambres a seleccionar con el MOUSE a 0.4 centímetros.
Display solder	Despliega la capa de soldadura si nos encontramos en el ambiente BOARD o si estamos modificando o creando una macro.
Grid on;	Use para observar en pantalla puntos o líneas, a una separación fija especificada por la opción Size de este comando. Seleccione las opciones con el MOUSE.
;	El punto y coma se utiliza para separar dos comandos, la selección de este indica el no progreso en ejecución de un comando.

2.5.1.1 Lógica de creación de elementos nuevos en el software EAGLE

El software EAGLE permite que Ud. cree sus propios elementos, para ello puede crear su propia librería o adicionar elementos a una librería existente.

Si el caso es crear una librería, siga los siguientes pasos:

a. En el punto de comando escriba OPEN MYLIB, donde MYLIB es el nombre del archivo de librería que Ud. quiere crear.

b. Utilice mandatos y opciones disponibles para crear las macros, símbolos y dispositivos para la librería.

C. Use el mandato WRITE para salvar todos los elementos creados.

D. Use el mandato CLOSE para cerrar la librería y retornar al ambiente por default. (Ambiente BOARD).

Si el caso es crear, adicionar o modificar elementos en una librería existente, siga los siguientes **pasos:**

a. Ejecute el mandato OPEN, luego escoja con el MOUSE la librería a la cual quiere adicionar el nuevo elemento.

b. Seleccione el comando EDIT y escoja la opción MACRO, luego elija la opción NEW; en este punto se le pedirá el nombre para el nuevo elemento, si quiere modificar una MACRO existente coloque el nombre de esta o selecciónelo con el MOUSE.

C. Utilice los comandos disponibles para la creación de macros, una macro no es mas que el encapsulado del elemento, este es el objeto que se colocará con el comando Add en el ambiente BOARD.

D. Escoja el mandato PAD para definir los puntos de conexión para la macro, utilice el mandato NAME para cambiar los nombres de los pads, estos son los elementos de conexión en el ambiente BOARD, estos nombres le permitirá conectar estos pads con los pines de su

respectivo símbolo; escoja la opción LAYER y luego PLACEPLAN del comando CHANGE, luego use el comando ARC (arco), WIRE (alambre), RECT (rectángulo), para dibujar la forma y tamaño del elemento, escoja el mandato TEXT y adicione los rótulos >NAME y >VALUE para identificar esta macro.

e. Una vez terminada la macro seleccione el mandato EDIT y escoja la opción SYMBOL, luego seleccione NEW, este le pedirá el nombre para el nuevo símbolo, coloque el mismo nombre que le puso a la macro si esta macro será solo para un símbolo, de lo contrario póngale un nombre que identifique correctamente a la macro; por ej. DIL14 que corresponde a una macro utilizada por los dispositivos que poseen 20 pines, si Ud. quiere modificar un símbolo existente coloque el nombre en NEW o selecciónelo con el MOUSE.

f. Utilice los comandos disponibles para la creación de símbolos, un símbolo es el elemento utilizado en el ambiente SCHEMATIC para la representación del elemento.

g. Escoja el mandato PIN para colocar los pines (elementos de interconexión en el ambiente SCHEMATIC), y el mandato NAME para cambiar los actuales nombres de estos por los nombres que identifiquen cada pin, una vez que se han colocado los pines escoja la opción FUNCTION del mandato CHANGE, luego seleccione entre las opciones NONE (ninguna), DOT (invertir), CLK (reloj) y DOTCLK (reloj invertido) para asignar la función que desempeñarán los pines del símbolo, luego seleccione el mandato DIRECTION del mandato CHANGE y seleccione entre NC (no conectar), IN (entrada), OUT (salida), I/O (entrada-salida), PWR (alimentación) etc.., para indicar la dirección de estos pines, una vez echo esto escoja la opción LAYER y luego SYMBOL del comando CHANGE para dibujar la forma y tamaño del elemento usando el mandato WIRE, ARC, RECT y NAME para crear y documentar el símbolo, luego con el mandato TEXT y habiendo seleccionado previamente la opción LAYER y luego NAMES coloque los rótulos >NAME y >VALUE para este símbolo.

h. Una vez terminado el símbolo escoja el mandato EDIT, seleccione la opción DEVICE y luego seleccione NEW, este le pedirá el nombre para el nuevo dispositivo, coloque el mismo nombre que le puso a la macro y al símbolo, si Ud. quiere modificar un dispositivo existente escriba el nombre o selecciónelo con el MOUSE.

i. Utilice los mandatos disponibles para dispositivos, aquí Ud. podrá unir varios símbolos del mismo nombre o diferentes, este espacio sirve para definir todos los elementos que el chip o dispositivo posee, como por ejemplo, definir el empaque (macro) y números de pines de acuerdo a la macro seleccionada.

j. Seleccione la opción ADDLEVEL y escoja NEXT.

k. Seleccione el mandato ADD y escoja el nombre del símbolo previamente creado.

l. Seleccione el mandato PACKAGE y escoja el nombre de la macro que creo, para este dispositivo.

m. Escriba el mandato CONNECT namepin(1) namepad(1) namepin(2) namepad(2) ... namepin(i-1) namepad(i-1) namepin(i) namepad(i); donde namepin(i) son los nombres de los pines (estos lo puso con el mandato NAME cuando creo este símbolo) y namepad(i) corresponden a los nombres de los pads (estos se los puso cuando creo la macro que ha asignado a este dispositivo). Este comando permitirá interconectar los pads con sus respectivos pines. Cuando se hace esto observara que sobre los pines se aparecen los nombres de sus respectivos pads.

n. Escoja el mandato NAME y nombre esta parte (compuerta) del dispositivo.

o. Escoja la opción ADDLEVEL y seleccione REQUEST del mandato CHANGE, luego con el MOUSE seleccione el símbolo al que acaba de conectarle los pads, vuelva a escoger ADDLEVEL y seleccione NEXT si desea adicionar un nuevo símbolo al dispositivo, esto nos permitirá seguir conectando los pines de este con sus respectivos pads.
p. Seleccione el mandato PREFIX y coloque el prefijo para este dispositivo, ej. IC.
q. Seleccione el mandato WRITE para guardar los cambios efectuados a la librería, y esta sea actualizada.
r. Seleccione el mandato CLOSE, para cerrar esta librería y retornar al ambiente por default (ambiente BOARD).

2.5.1.2 Lógica de creación del esquema para la tarjeta CPAL en el software EAGLE

Una vez todos los dispositivos que utilizaremos se encuentran disponibles en las librerías de EAGLE, y nos encontremos en el ambiente SCHEMATIC, utilizamos los mandatos ADD, NET, JUNCTION, BUS, NAME, LABEL, SHOW, SPLIT, TEXT, VALUE, USE, CHANGE, DELETE, MOVE, MIRROR, ROTATE, WRITE, UNDO y REDO para la creación de este esquema.

La lógica de creación es la siguiente:

a. Colocamos todos los dispositivos que utilizará en la pantalla para ello utilice el mandato USE para seleccionar la librería que contiene el dispositivo a colocar, luego se utilizó el mandato ADD para seleccionarlo. Cuando colocamos dispositivos tales como el 74LS133, se observó que las compuertas quedan identificadas con una letra y un prefijo tal como IC8F, donde 13 es él numero de IC y la letra M corresponde a la cuarta compuerta del IC, como se usan 13 compuertas, cada una se identifica con el nombre 74LS133 y con sus letras y números de pines correspondientes. Estos números de pines están dispuestos tal como está especificado en el manual TTL para dicho dispositivo. Si fue necesario reubicar dispositivos esto lo hice, seleccionando el mandato MOVE.

b. Con el mandato NET sé interconectarón los pines de estos elementos, de acuerdo al diagrama inicial de pin out.

c. Utilizamos el mandato BUS para colocar buses en aquellos lugares donde la cantidad de líneas es densa, tales como en el bus de datos y el bus de direcciones.

d. Con el Mandato JUNCTION colocamos uniones en aquellas líneas donde se bifurcaban dos o más del mismo punto, de manera tal que no queden dudas sobre las conexiones de estas.

d. Con el mandato SHOW se observó que todas las líneas unieran correctamente los pines correspondientes.

e. Se uso el mandato NAME para asignar nombres a las líneas, buses y dispositivos que lo ameritaban, de manera que puedan ser entendibles fácilmente. Con los nombres de las líneas debemos tuvimos cuidado al asignarle nombres, esto es si a una línea se le asigna un nombre de otra existente aunque estas no estén físicamente conectadas con NET estas quedarán conectadas a la hora de generar el circuito impreso a partir de este esquema.

f. Utilizamos el mandato LABEL para etiquetar los buses. Solamente se selecciona este mandato y luego con el MOUSE se selecciona el bus que se quiera etiquetar.

g. Una vez concluido el esquema utilizamos el mandato WRITE para guardar en disco este esquema, este mandato nos pide escribir el nombre del archivo, este no debe exceder de 8 caracteres de nombre; la extensión que coloca el software por default es BRD.

2.5.1.3 Lógica de creación del circuito impreso en el software EAGLE

En la sección anterior se detallo la lógica para crear el circuito esquemático, en esta sección se detallará la creación del circuito impreso a partir del circuito esquemático, para ello el esquema debe tenerse en pantalla y debidamente guardado en disco.

Pasos a seguir para generar el circuito impreso:

a. Utilice el mandato BOARD teniendo el esquema en pantalla.

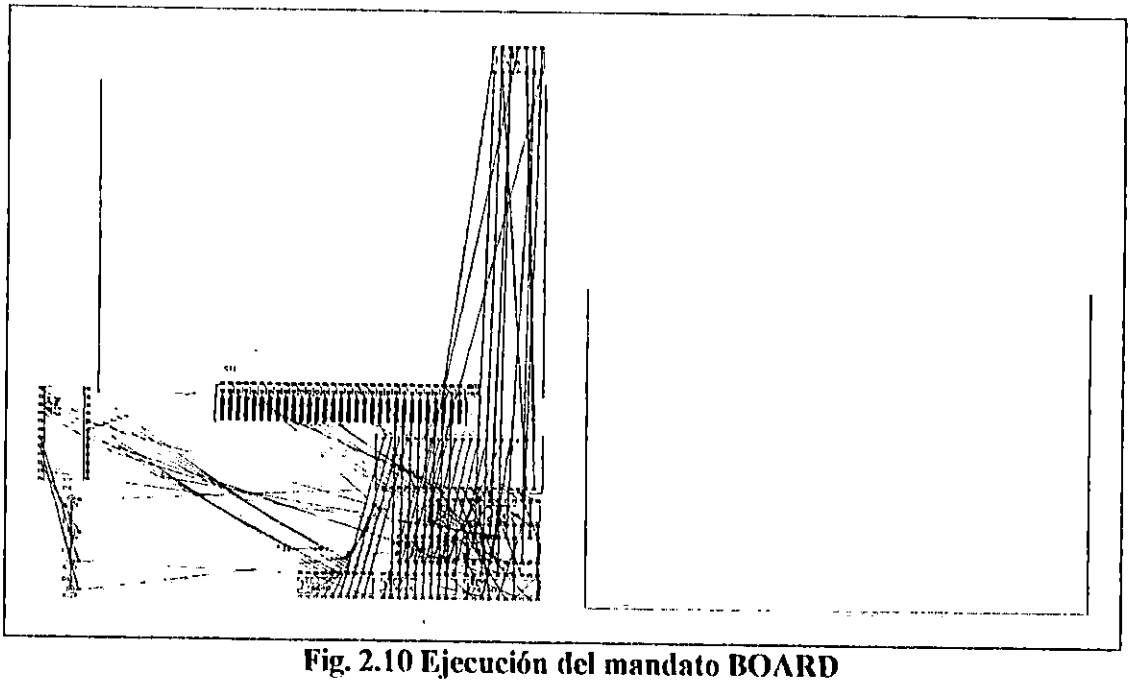


Fig. 2.10 Ejecución del mandato BOARD

Observe que las macros correspondientes a cada símbolo han sido colocados sobre la pantalla, estas macros se encuentran desordenadas y conectadas con segmentos de líneas llamadas signal, estas pueden ser enrutadas manualmente al utilizar el mandato ROUTE, estas líneas se hacen utilizando el mandato SIGNAL y solo pueden ser interconectadas sobre elementos pads y smds. También observe el recuadro que aparece al lado derecho de las macros este recuadro representa el área de pistas para la conexión de estas macros, este recuadro puede crearse al seleccionar la opción LAYER y luego escoger DIMENSION del mandato CHANGE, una vez echo esto, el recuadro se dibuja utilizando el mandato WIRE. Entre los cuidados que debemos tener es que este recuadro debe encontrarse en una región de la pantalla donde las coordenadas sean positivas, de lo contrario se generará una alarma de peligro a la hora de que el paquete intente auto en rutar.

También observe que los dispositivos han sido automáticamente conectados. Estas conexiones corresponden a las echas en el circuito esquemático con el mandato NET.

b. En este punto utilizamos el mandato MOVE para colocar todos las macros dentro del área de en rutado y debidamente posicionados, esto es debidamente distribuidos y colocados, la figura 2.6 muestra la distribución de estas macros.

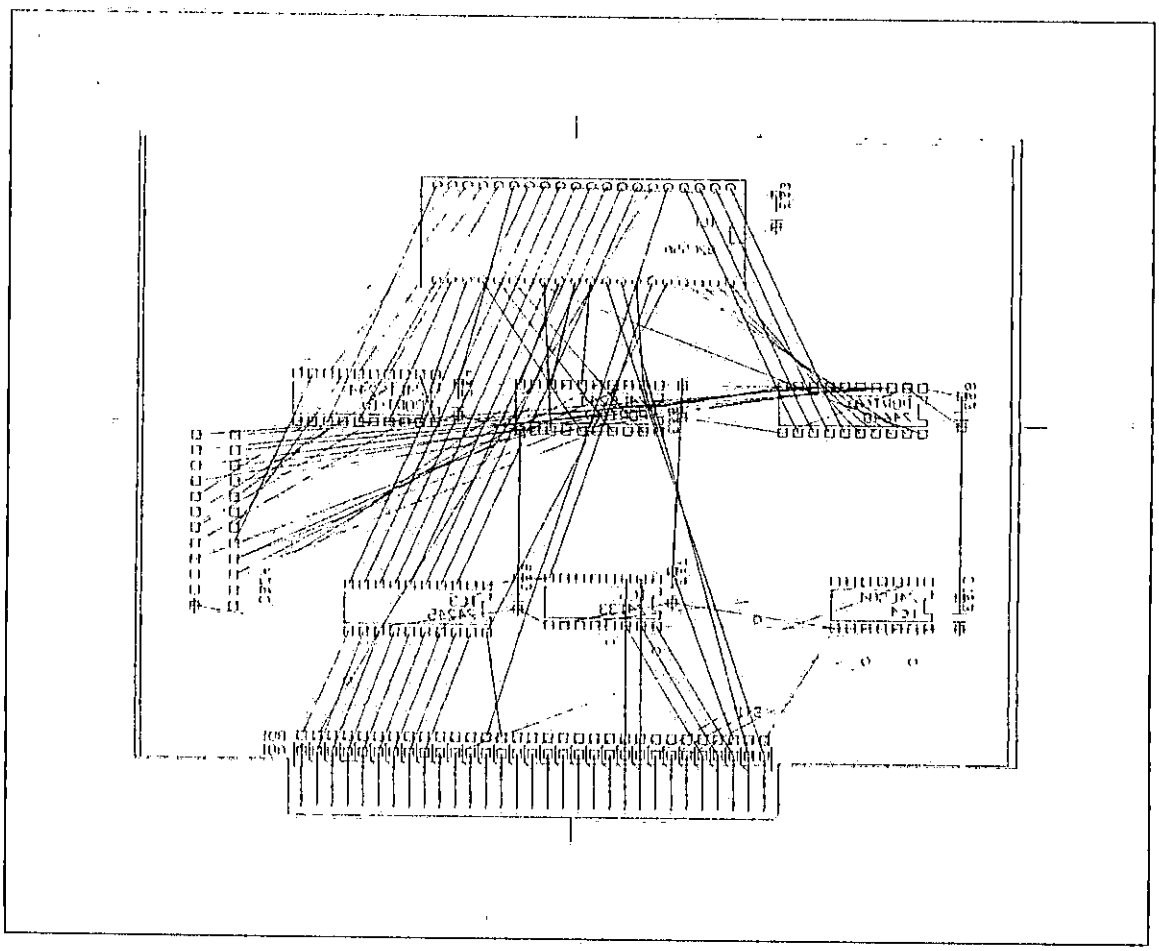


Fig. 2.11. Distribución de las macros sobre el área de en rutado

c. Utilizamos el mandato USE para abrir la librería Packages y añadir la macro correspondiente a la de un capacitor de pastilla para cada IC, esta macro se llama C5. Este capacitor sirve como un filtro, para las señales de alta frecuencia en las pistas de alimentación (señales de ruido)del IC, el valor de estos es de 1nF. Los pads de estas macros se interconectan a los pads de alimentación para cada IC con el mandato SIGNAL.

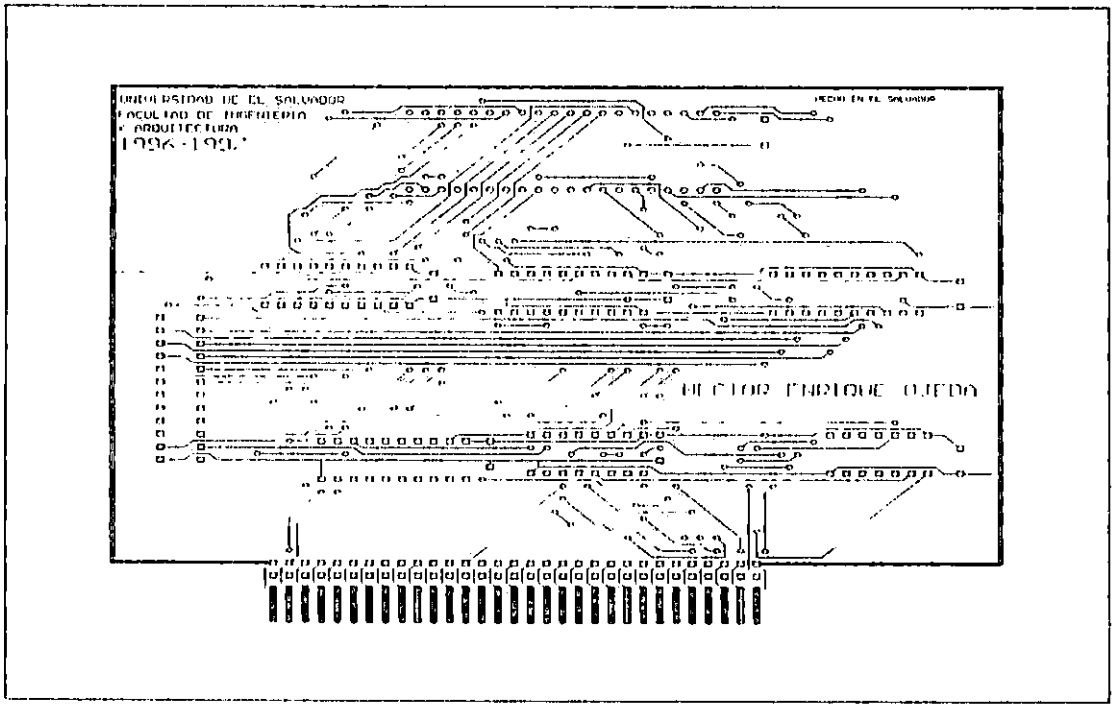
d. Utilizamos el mandato ROUTE para en rutar manualmente todas aquellas señales que no se desea que sean en rutadas por el paquete, si él en rutado lo queremos hacer sobre la cara de componentes la opción LAYER y luego la opción COMPONENT del mandato CHANGE debe seleccionarse antes de utilizar el mandato ROUTE, si se desea en la cara de soldadura envés de la opción COMPONENT seleccione la opción SOLDER. Una vez todos los elementos están situados y conectados ejecutamos el mandato SET AUTO_WIDTH 0.4064 para fijar el ancho de las pistas a 0.4064 cm, luego utilizamos el mandato AUTO para que el paquete en rute las señales restantes, la ejecución de este mandato toma unos minutos al ordenador, si el porcentaje de en rutado no es de 100% significa que aun existen señales que no pudieron ser enrutadas, si este es el caso utilizamos el mandato RIPUP en varias de las señales que ya están enrutadas para desenrutarlas (regresarlas a signal) en las cercanías de las que no pudieron ser enrutadas. Luego volvemos a utilizar el mandato AUTO, este procedimiento se puede repetir hasta que todas las señales han sido completamente enrutadas.

e. Utilizando el mandato MOVE movemos las pistas que se hallan cerca del lugar que se utilizará para fijar el conector DB24; utilizamos el mandato REC para dibujar rectángulos que delimiten la región donde se fijará el conector. Luego utilizamos el mandato SNAP para corregir aquellas pistas que no cumplen con las reglas eléctricas de en rutado.

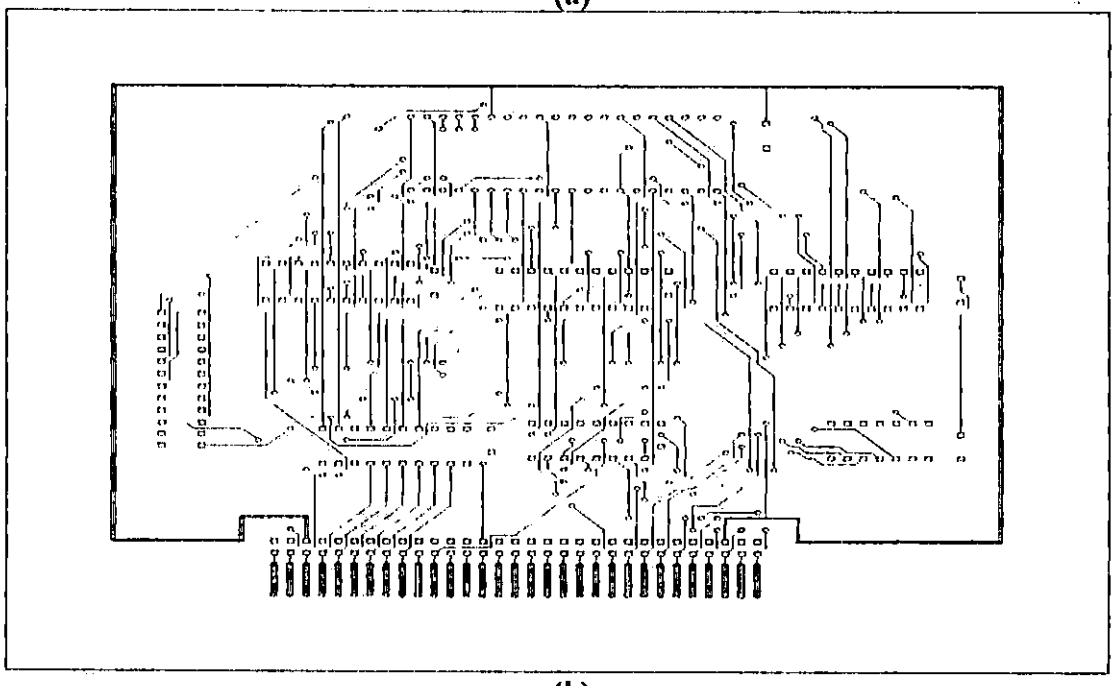
Efectuando todos los pasos anteriores y con un poco de imaginación y estética se termino el circuito impreso. La figura 2.7, muestra el esquema del circuito impreso para la cara de componentes y la cara de soldadura. Las dimensiones para la tarjeta son de 11.5 x 15 cm, estas medidas pueden ser variadas hasta un máximo de 11.5 x 24 cm.

Los siguientes esquemas se imprimieron en un impresor láser, utilizando el archivo XPLOT que es parte del software EAGLE. Estas impresiones se llevaron a un taller de serigrafía para que las grabaran sobre la superficie de una tableta cobreada doble cara.

Solamente se presentara el diseño para una tarjeta y las demás se presentan terminadas con este mismo procedimiento.

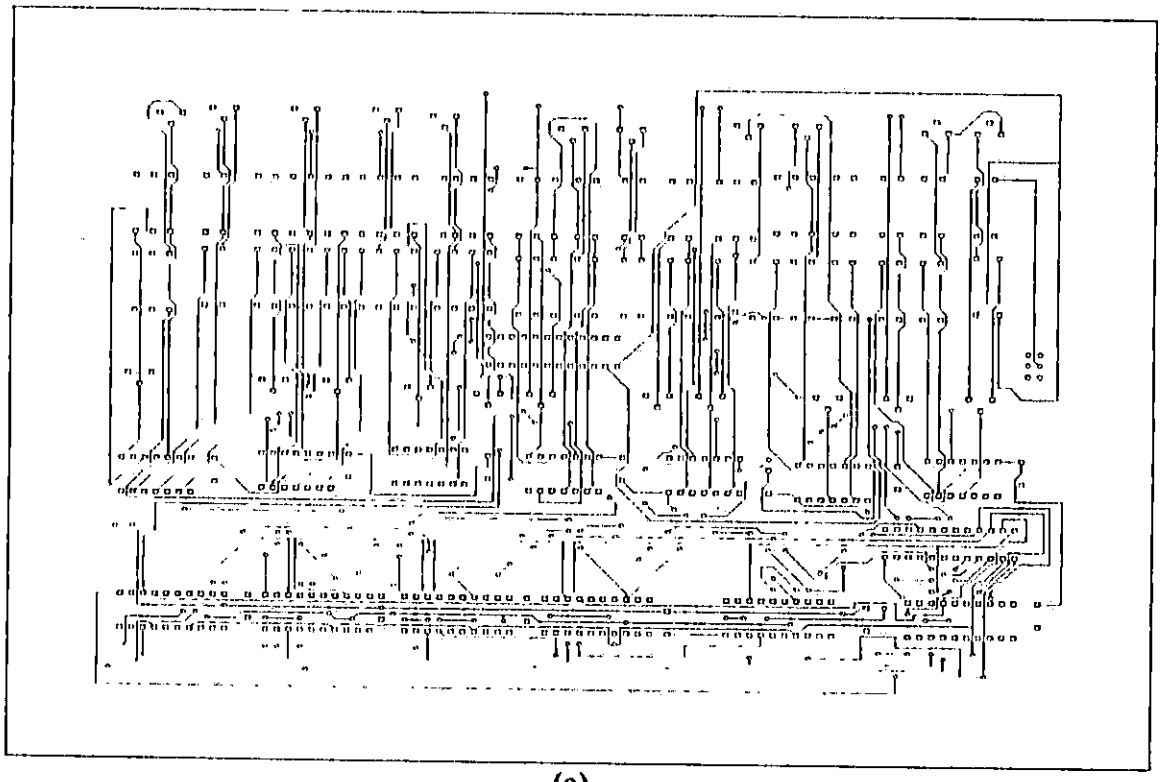


(a)

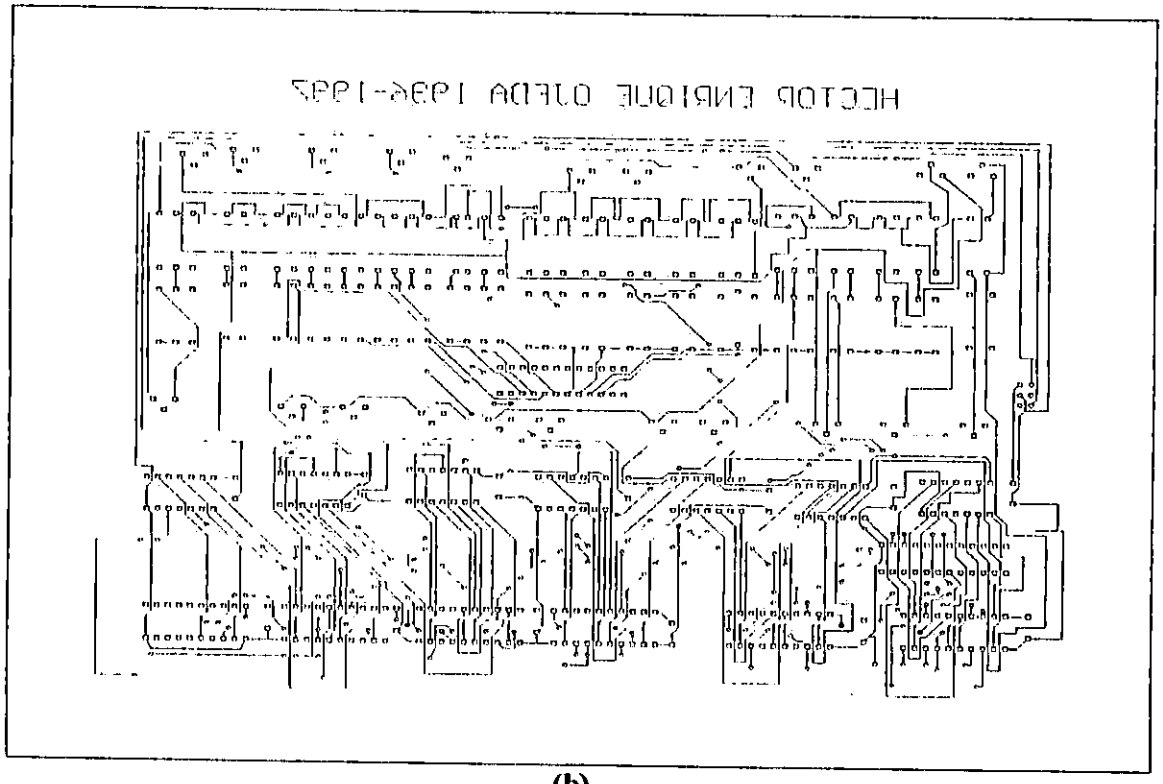


(b)

Fig. 2.12 Esquema de la tarjeta impresa, (a) Lado de los componentes (b) Lado de la soldadura

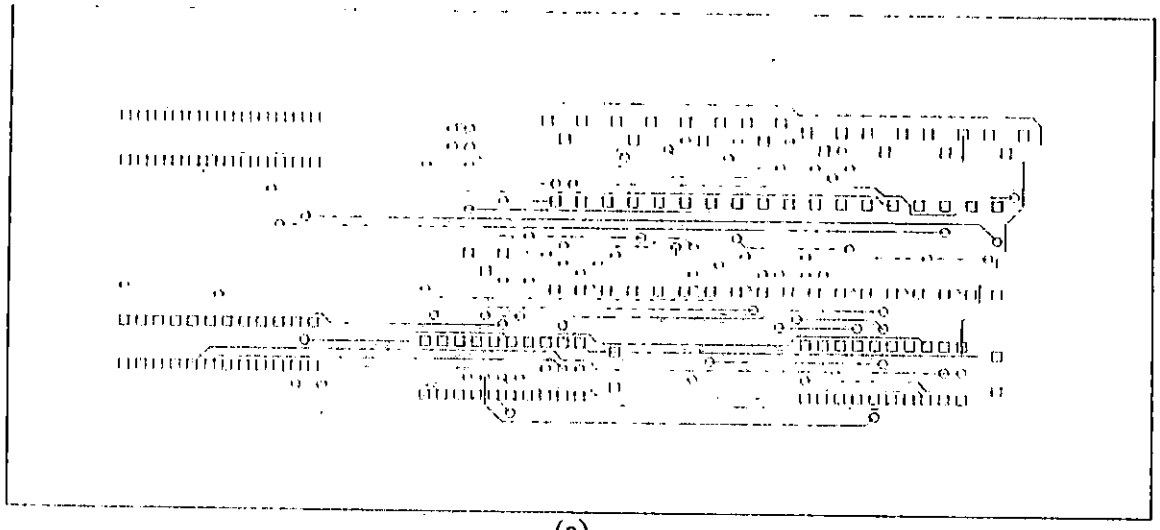


(a)

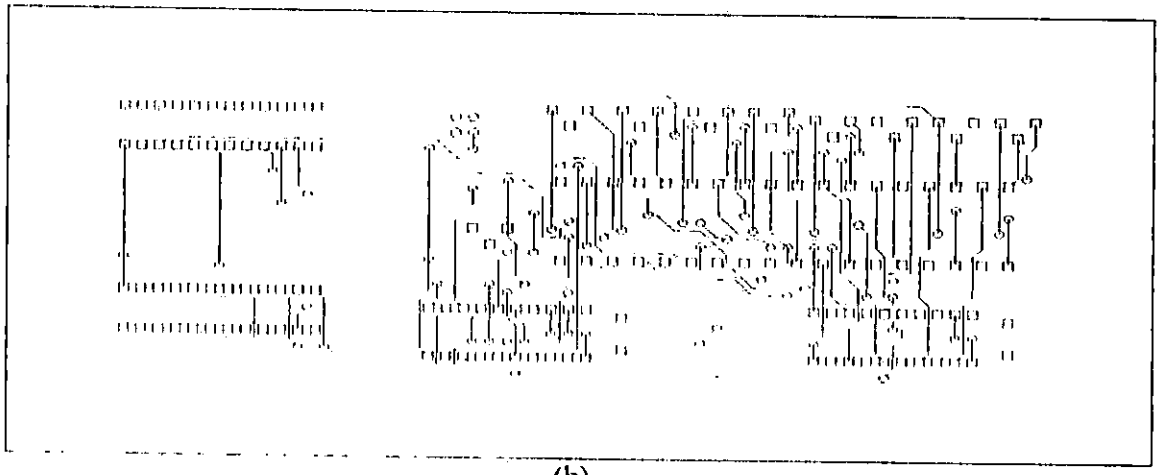


(b)

Fig. 2.13 Esquema de la tarjeta impresa, (a) Lado de los componentes (b) Lado de la soldadura

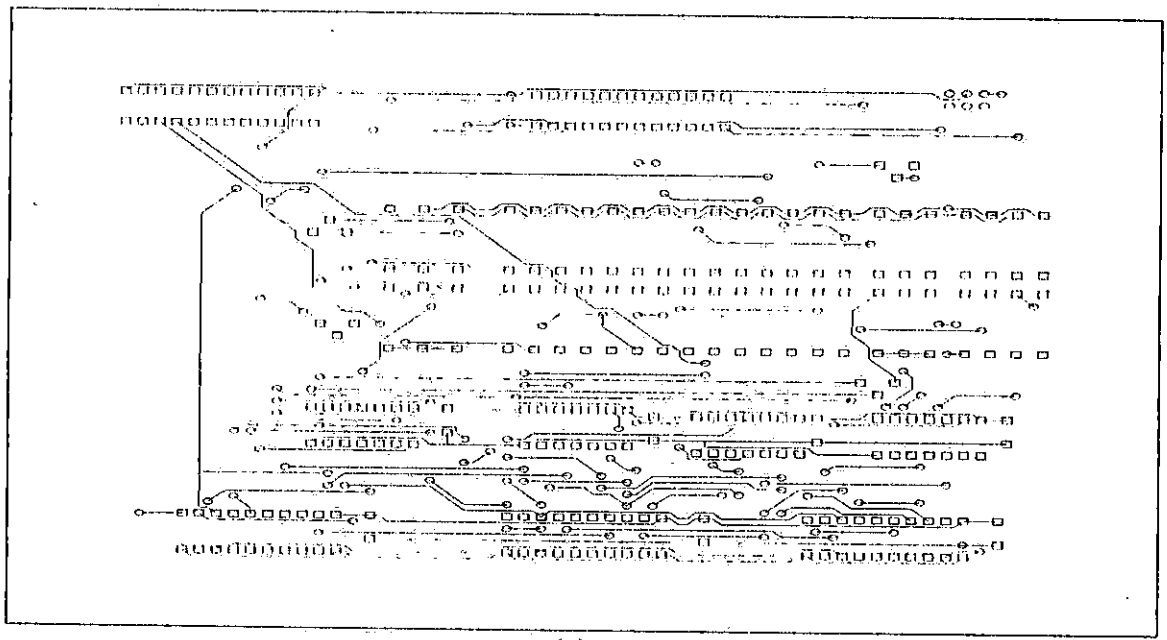


(a)

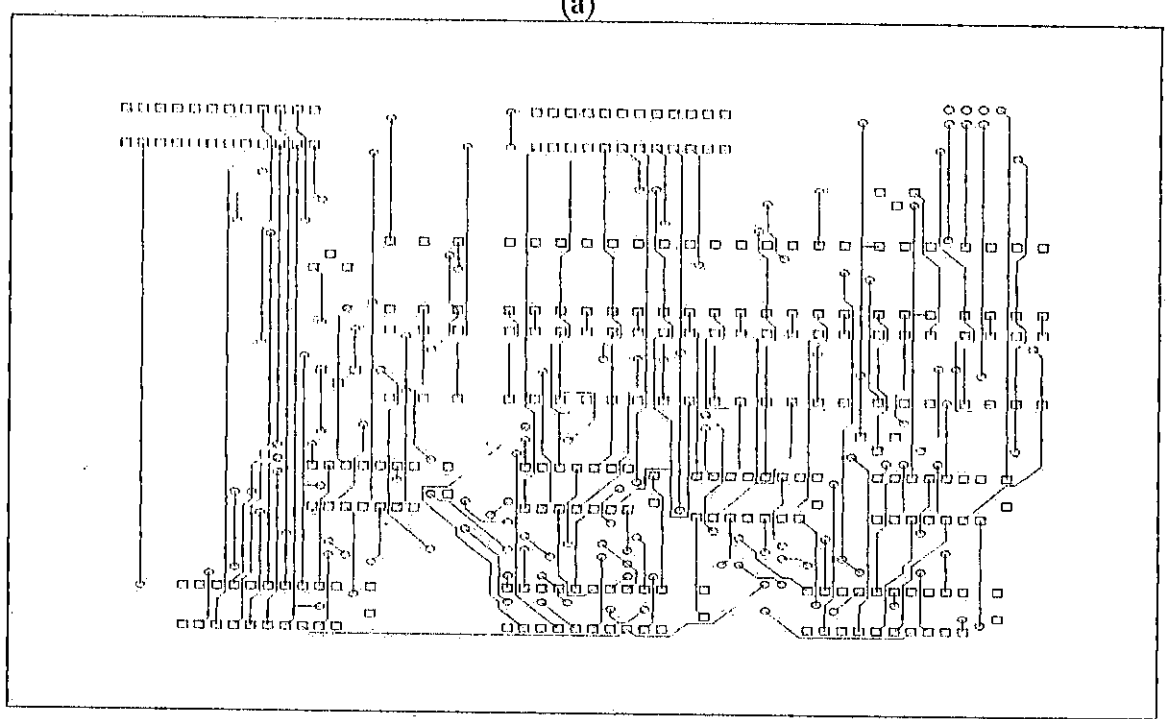


(b)

Fig. 2.14 Esquema de la tarjeta impresa, (a) Lado de los componentes (b) Lado de la soldadura



(a)



(b)

Fig. 2.15 Esquema de la tarjeta impresa, (a) Lado de los componentes (b) Lado de la soldadura

Todos los circuitos impresos realizados hasta el momento los realice con el auxilio de la computadora, y . el correspondiente Software apropiado, tal como EAGLE. Y como resultado final se obtiene los diseños de las tarjetas de calidad en los impresos.
El procedimiento de diseño de los impresos se presenta a continuación.

Pasos:

1. Diseñar los circuitos primero
2. introducir al EAGLE los elementos de cada circuito de tal manera de lograr cada tarjeta.
3. Unir con el EAGLE cada punto de los circuitos con señales.
4. Mandar a en rutar todas las señales de la tarjeta diseñada con el comando AUTO del EAGLE para que el paquete utilice su algoritmo para en rutar todas las señales, sin excepción alguna.
5. Imprimir los archivos en plotear de tinta en papel vegetal.
6. Sacar negativos o positivos según el que se necesite, esto depende del método a ser utilizado para hacer el impreso en tableta. Para el caso en particular el negativo fue el utilizado, porque se hizo por el método de serigrafía.
7. Realizar la pantallas necesarias por caras para los impresos
8. Realizar las pintas de las caras de los circuitos de tal manera que coincida. Utilizando perforaciones con broca de 1/64 para no dañar el impreso.
9. Cuando esta estampado completamente el dibujo en la tableta de cobre de dos caras se procede realizar la quema de la tableta con **Percloruro** de hierro.
10. Cuando la tarjeta esta terminada se realizan las perforaciones con broca de 1/64 como guía para luego utilizar broca 1/32 y agrandar los orificios donde eran alojados los elementos de las tarjetas.
11. Preparado de la tableta con un liquido llamado KESTER antes de realizar las soldaduras en ella con estaño evita que el estaño pierda sus propiedades y se produzcan soldaduras frías.
12. Colocación de todos los elementos en la tableta
13. Realizar pruebas de la tarjeta y depurar errores hasta que funcione correctamente, y corregir los cortocircuitos que se encuentren produciendo como resultado del proceso de fabricación.

CONCLUSIONES

1. El diseño de las tarjetas impresas de doble cara se logran con gran calidad, garantía, y confiabilidad como producto final, si se realiza con el Software EAGLE y el auxilio de la computadora.

2. El método sugerido y seguido en este apartado para la construcción de circuitos impresos, previa obtención de los negativos del circuito es el de Serigrafía.

3. Para la Conmutación de voltajes necesarios en los diferentes pines de la base ZIF, estos se obtienen con los circuitos compuestos de Latch, inversores ("open collector"), resistencias, transistores, diodos, garantizando la conmutación de los niveles con el menor consumo de potencia y menor espacio utilizado en las tabletas impresas.

REFERENCIAS BIBLIOGRAFICAS

-Tocci, Ronald J. Sistemas Digitales, Principios y Aplicaciones. México Prentice Hall Hispanoamericana S.A, 5ª. Edición 1993.

-Texas Instruments Enginneering staff. TTL Data Book For Design Engeneers. Dallas, Texas, Texas Instruments Incorporated. 2da. De. 1976

-Texas Instruments General Information Funcional Index Field Programmable Lógic USA, Texas Instruments Incorporated 1985

-Texas Instruments Programmable Lógic Data Book, Dallas Texas, Texas Instruments Incorporated. 1990

- Trabajo de Graduación Programador de Arreglos Lógicos Programables (PALs) El Salvador, UCA 1990

-Bolestar, R y Nashelsky, L. Electronica, Teoría de circuitos. México, Edit. Prentice Hall 4ª. Edit. 1993

-Trabajo de Graduación Diseño y Construcción de un Programador Inteligente de Circuitos de Arreglo Lógico Programable (PAL) Basado en el Microprocesador 8085. El Salvador (Universidad Politecnica) 1996

-Advanced Micro Devices Programmable Array Logic Handbook, 1984

-Trabajo de graduación Diseño e Implementación de un programador de PAL operado por un computador, El Salvador (Unversidad Politecnica) 1987

-Advanced Micro Devices Programmable Array Logic Handbook, 1997

CAPITULO III

DESARROLLO DEL SOFTWARE QUE CONFORMARAN LOS DRIVER DE BAJO NIVEL QUE MANEJARAN EL HARDWARE.

Introducción

En este capitulo se diseña y desarrolla el software para el sistema de este trabajo, este debe ser capaz de manejar una tarjeta CPAL controladora. La que gobernara sobre las tarjétas de las fuentes conmutables

El software diseñado e implementado deberá de generar las palabras necesarias a enviar en el bus para colocar un determinado nivel de voltaje en algún pin en particular.

El software desarrollado para este sistema esta escrito en lenguaje Visual C++ 5.0 de la Microsoft, este lenguaje ofrece un marco de la aplicación basado en Windows llamado *la biblioteca de clases* "Microsoft Foundation Class Library" versión 5.0

Pero la aplicación desarrollada hasta este momento es basada prácticamente solamente C++. Por que, se diseñan los Driver que manejaran y controlaran el Hardware diseñado y Construido, para el programador de PAL's.

Objetivos

1. -Desarrollo del software (drivers) necesarios para controlar el Hardware diseño y construido.
2. -Desarrollo de una interfaz visual que permita utilizar los drivers

3.0 Análisis del Hardware para sacar las palabras a enviar en el bus.

Para el caso que se desarrollara un software apropiado al hardware construido primero se presenta la necesidad de realizar un análisis de las palabras necesarias a enviar sobre el bus para conseguir el voltaje deseado en un pin particular.
Para ello es necesario realizar un encapsulamiento de las palabras a enviar sobre el bus con las siguientes variables.

3.1Codificación de toda la Información

Es necesario realizarlo en palabras de 16 bits en una tabla NoElmPalabras son 105 palabras
Para los 8 bits MSB se encapsula la siguiente información

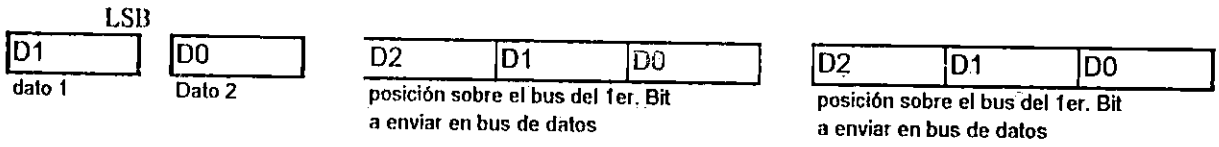
D4	D3	D2	D1	D0

Control de número de pin sobre la base ZIF

D2	D1	D0

control de Enable

Para los 8 bits LSB se encapsula la siguiente información
MSB

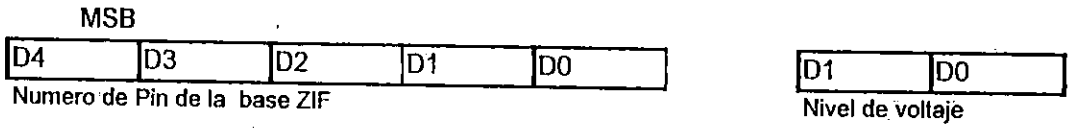


La palabra formada con 16 bits garantiza describir completamente las posiciones sobre el bus y el valor lógico que es necesario enviarle sobre el mismo esto se realiza en los 8 bits menos significativos. Con los 8 bits más significativos sé encansula la siguiente información el número de enable, y el número de pin. Al cual se le estará enviando la palabra encapsulada en los bits menos significativos.

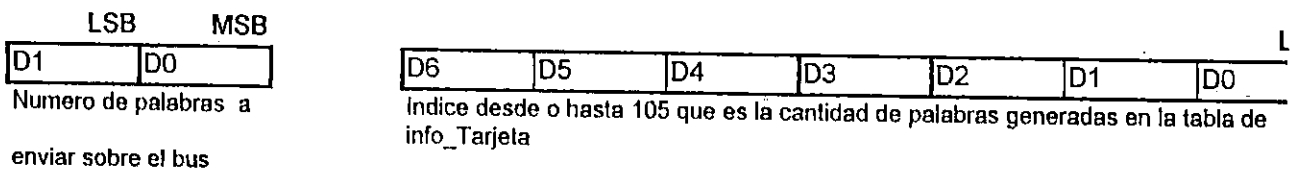
La palabra formada hasta este momento me describe completamente el hardware de las fuentes conmutables.

Para ello es necesario auxiliarnos de otra tabla NoElmDescripcion para con esta ultima ubicarse en cualquier posición de la tabla NoElmPalabras

Tabla de NoElmDescripcion la cual esta formada por 89 palabras
Para los 8 bits MSB se encapsula la siguiente información



Para el primer bit MSB y los 8 bits LSB se encapsula la siguiente información



Similarmente se realiza el encapsulamiento de las tablas proporcionadas por el fabricante Texas Instruments y de Advanced Micro Device, solo que se realiza con las siguientes particularidades.

Cuando sé esta trabajando con las tablas de proporcionadas por los fabricantes **Input line select y la Product line select,** se realiza también un encapsulamiento de la información en palabras de 16 bits con los siguientes formatos:

Tabla Input Line Select, donde se encuentran la información proporcionada por los fabricantes para las líneas de entradas, donde se encuentra descrito el nivel de voltaje que se necesita colocar en el pin en particular. Por conveniencia de niveles de voltajes se agruparon los pines que presentan niveles de voltajes iguales en una tabla común para mejor interpretación de la información por el software.

Codificación de Tabla Input Line
Palabra de 16 bits

MSB																	LSB
INFO	X		X	X	X	X	L/	PI9	PI8	PI7	PI6	PI5	PI4	PI3	PI2	PI1	PI0

Según la información de los fabricantes los pines PI0 al PI9 se les debe de colocar los niveles de voltajes de 10.5v, 5v, y 0v, según la tabla 1.1 en él capitulo I

La codificación realizada sigue la siguiente lógica:
1. -Cuando en el pin se necesita colocar 10.5v en la palabra se encontrará 1 lógico, y el bit MSB INFO de la palabra no interesa el valor que él tenga.
2. - Cuando sé necesita un nivel de 5.v ó de 0v el que se necesita colocar en cualquier pin Pix, entonces se analiza de la siguiente manera.
Si es 5v entonces el valor lógico para ese pin es 1 lógico en el BIT INFO. (MSB de toda la palabra)
Si es 0v entonces el valor lógico para ese pin es de 0 lógico en el BIT INFO(MSB de toda la palabra)

Codificación de Tabla Product Line Select
Palabra de 16 bits

MSB																	LSB
X	PO0	PO1	PO2	PO3	PA2	PA1	PA0	PI9	PI8	X	PO0	PO1	PO2	PO3	PA2	PA1	PA0

Para esta tabla se realice que por cada fila de este arreglo es igual a dos filas de la tabla de l fabricante y se hacen las siguientes consideraciones. Según tabla 1.2 en él capitulo I.

Si es 10.5v entonces el valor lógico para ese pin es 1 lógico en el BIT donde se encuentra ubicado el pin.
Si es Zv entonces el valor lógico para ese pin es de 0 lógico en el BIT donde se encuentra ubicado el pin.

3.2 Implemetación de las funciones de los Drivers.

El concepto general del software que se desarrollara estará basado en las tablas codificadas con anterioridad, para tener toda la información necesaria para quemar un fusible almacenada en estructura. Y si se necesita solo se tomará.

Para ello se parte de que ya se tienen todas las tablas generaras descritas anteriormente residentes en memoria, se procederá a realizar la conversión de la información de Hexadecimal a Código ASII, que es un archivo mucho más pequeño. Que

quedara residente en memoria. Y esto se realiza con una función implementada se ha llamado TABLAAPP.C.

Función TABLAAPP: C

```
#include <stdio.h>
#include <string.h>
void main(int narg, char *arg[]) {
        if (narg>1) {
                FILE *file_t = NULL;
                file_t = fopen(arg[1], "r");
                FILE *file_b = NULL;
                unsigned short value = 0;
                char *p = strchr(arg[1], '.');
                if (p != NULL) {
                        *(++p) = 'd'; *(++p) = 'r'; *(++p) = 'v';
                        file_b = fopen(arg[1], "w+b");
                }
                int i=0;
                if (file_t != NULL && file_b != NULL) {
                        while(!feof(file_t)) {
                                if(fscanf(file_t, "%x",&value)<=0) break;
                                i++;fwrite(&value, sizeof(unsigned short), 1, file_b);
                        }
                }
                if (file_t != NULL) fclose(file_t);
                if (file_b != NULL) fclose(file_b);
        } else {
                printf("Este prog. lee un archivo texto de # en exadecimal (0x0000)\ny
genera un arhivo binario\n\nUse : tablaapp <namefile.txt>\n\tOut: namefile.drv");
        }
        return;
}
```

En esta función recibe como argumento un archivo de Texto y retorna un archivo de codigo ASCII. Que es el que continúen almacenando toda la información necesaria de las tablas antes codificadas.

Cuando se tiene toda la información almacenada en un archivo ASCII generado por el Programa TABLAAPP:C se debe de realizar el procesamiento de esa información y realizar la inicialización del hardware y almacenar toda la información de las tablas en estructuras diferentes .

```
int InicializeTarjeta(void) {
int i;
        char filename[255];

        strcpy(filename, DIRNOMBRETABLAS_BIN);
```

```
        strcat(filename, "out.txt");

        g_File = fopen (filename, "w");
        for (i=0; i<10; i++) last_latch[i] = 0;

        _outp((unsigned short)ID_BC, (int)BYTECTRL_LEER);
         outp((unsigned short)ID_PB, (int)BYTECTRL_INICIALIZA);
        _outp((unsigned short)ID_BC, (int)BYTECTRL_ESCRIBIR);
        strcpy(filename, DIRNOMBRETABLAS_BIN);
        strcat(filename, NOMBRETABLAS_BIN);

        if ((fileP_b = fopen(filename, "r+b")) != NULL) {
                if ((g_elmFila = (unsigned short *)malloc((NOELMFILA *
sizeof(unsigned short)) + 1)) == NULL)
                return EROR_MEMORIANODISPONIBLE;
                else if ((n_g_elmFila = fread(g_elmFila, sizeof(unsigned short),
NOELMFILA, fileP_b))<NOELMFILA)
                        return ERROR_LEERDATOSBIN;

                if ((g_elmColumna = (unsigned short *)malloc((NOELMCOLUMNA *
sizeof(unsigned short)) + 1)) == NULL)
                return EROR_MEMORIANODISPONIBLE;
                else if ((n_g_elmColumna = fread(g_elmColumna, sizeof(unsigned short),
NOELMCOLUMNA, fileP_b))<NOELMCOLUMNA)
                        return ERROR_LEERDATOSBIN;

                if ((g_elmPalabras = (unsigned short *)malloc((NOELMPALABRAS *
sizeof(unsigned short)) + 1)) == NULL)
                return EROR_MEMORIANODISPONIBLE;
                else if ((n_g_elmPalabra = fread(g_elmPalabras, sizeof(unsigned short),
NOELMPALABRAS, fileP_b))<NOELMPALABRAS)
                        return ERROR_LEERDATOSBIN;

                if ((g_elmDescripcion = (unsigned short
*)malloc((NOELMDESCRIPCION * sizeof(unsigned short)) + 1)) == NULL)
                return EROR_MEMORIANODISPONIBLE;
                else if ((n_g_elmDescripcion = fread(g_elmDescripcion, sizeof(unsigned
short), NOELMDESCRIPCION, fileP_b))<NOELMDESCRIPCION)
                        return ERROR_LEERDATOSBIN;
//              for(i=0;i<10;i++) mem_datos_enables[i]=0;

        }

        return VERDADERO;
}
```

Continuando con la lógica de programación se continua a realizar la codificación de los Pines, Tipos de PAL´s y cantidad de Pines, con la siguiente función.

```
int CodificarPines(InfoPal *XXPalXXXX) {
        int i, j, k, l, np, npa, nw, id, gnd;
        int Error = FALSO;

        switch(XXPalXXXX->d_fabricante) {
                case TEXAS:
                        switch(XXPalXXXX->d_numPines) {
                                case TXNPINPAL16:
                                        if ((XXPalXXXX->d_pines = (PinesZif
*)malloc((TXNPINPAL16 * sizeof(PinesZif)) + 1)) == NULL) return VERDADERO;
                                        for(npa = 0, i=0, np = 0; i<TXNPINPAL16 -1 &&
!Error; i++, np = 0) {
                                                if (i>8) gnd = 1; else gnd = 0;

                                                for (j=0; j<NOELMDESCRIPCION;j++) if
((((g_elmDescripcion[j] & 0xf800)>>0x000b) & 0x001f) == (i + 3 + gnd)) { npa = j + 1;
np++; }

                                                if (np>0) {
                                                        XXPalXXXX-
>d_pines[i].d_nPosiciones = np;
                                                        XXPalXXXX->d_pines[i].d_lEntrada
= NULL;
                                                        if ((XXPalXXXX-
>d_pines[i].d_lEntrada = (LDeEntrada        *)malloc((np * sizeof(LDeEntrada)) +
1))==NULL) { Error = VERDADERO; break; }
                                                        for (l = 0,j=npa-np; j<npa;j++, l++) {
                                                                nw = (((g_elmDescripcion[j]
& 0x0180)>>0x0007) & 0x0003);
                                                                id = (((g_elmDescripcion[j] &
0x007f)>>0x0000) & 0x007f);
                                                                XXPalXXXX-
>d_pines[i].d_lEntrada[l].d_nPalabras = nw;
                                                                XXPalXXXX-
>d_pines[i].d_lEntrada[l].d_palabras = NULL;
                                                                if ((XXPalXXXX-
>d_pines[i].d_lEntrada[l].d_palabras  = (InfoTarjeta *)malloc((nw * sizeof(InfoTarjeta))
+ 1)) == NULL) { Error = VERDADERO; break; }
                                                                switch((((g_elmDescripcion[j]
& 0x0600)>>0x0009) & 0x0003)) {
                                                                        case N_VOLTAJE_0:
                                                                                XXPalXXXX-
>d_pines[i].d_lEntrada[l].d_voltaje  = N_VVOLTAJE_0;
```

```
                                                    break;
                                                case N_VOLTAJE_5:
                                                    XXPalXXXX-
>d_pines[i].d_lEntrada[l].d_voltaje = N_VVOLTAJE_5;
                                                    break;
                                                case N_VOLTAJE_12:
                                                    XXPalXXXX-
>d_pines[i].d_lEntrada[l].d_voltaje = N_VVOLTAJE_12;
                                                    break;
                                                case N_VOLTAJE_21:
                                                    if (i<=9 ||
i==18) XXPalXXXX->d_pines[i].d_lEntrada[l].d_voltaje = N_VVOLTAJE_21;
                                                    else
XXPalXXXX->d_pines[i].d_lEntrada[l].d_voltaje = N_VVOLTAJE_Z;
                                                    break;
                                                }
                                                for (k=0; k<nw;k++) {
                                                    XXPalXXXX-
>d_pines[i].d_lEntrada[l].d_palabras[k].d_palabra = g_elmPalabras[id + k];
                                                }
                                            }
                                        }
                                    }
                                    break;

                                case TXNPINPAL20:
                                    break;
                            }
                            break;
                        case AMD:
                            break;
                        default:
                            break;
                    }
                    return Error;
}
```

Función que colocará el nivel de voltaje en un pin en particular se inplementa a continuación con el objeto que se realice la colocación del nivel de voltaje discretamente. Los niveles de voltajes pero a la velocidad de micro y del software.

Esta es la función que presenta el corazón del drivers desarrollado hasta el momento. Como todo se ha desarrollado de una manera de estructuras la implementación

de los Drivres se pueden continuar desarrollando funciones utilizando los drivers que serán los que manejarán el hardware construido y diseñado hasta el momento.

```
int ObtenerPin(int npin,  int voltaje, Palp *Pines, InfoPal *XXPalXXXX) {
        int i, j, k, idx, npos,npos1;

        // si npin<10 | npin>14 => ad = D_PC para enables;
        // si npin == 13 | npin == 14 => ad = D_PB para los enables
        // todas las palabras de datos son con ad = D_PA

        unsigned short ad_datos = D_PA;
        unsigned short ad_enable;
        unsigned short enable_d, nenb;
        unsigned short dato_d, dato_t, dato_d1;
        unsigned short stop,aux,aux1,aux2;

        Pines->nwda = 0;
        switch (XXPalXXXX->d_numPines) {
                case TXNPINPAL16:

                        if (npin<10) idx = npin - 1;
                        else idx = npin - 2;

                        for (i=0; i < XXPalXXXX->d_pines[idx].d_nPosiciones; i++) {
                                stop=0;

                                if (voltaje == XXPalXXXX-
>d_pines[idx].d_lEntrada[i].d_voltaje) {

        //if((voltaje==N_VVOLTAJE_Z)&&((npin<=11)||(npin==20))) continue;

                                        Pines->wda = (Wdad *)malloc(sizeof(Wdad) *
(3*XXPalXXXX->d_pines[idx].d_lEntrada[i].d_nPalabras)+1);
                                        Pines->nwda = 3*XXPalXXXX-
>d_pines[idx].d_lEntrada[i].d_nPalabras;
                                        aux=0;
                                        aux1=0;
                                        aux2=0;
                                        for (j = 0,k=0; j<XXPalXXXX-
>d_pines[idx].d_lEntrada[i].d_nPalabras;j++,k+=3) {
                                                // para enable

                                                if((npin<10) && (voltaje == XXPalXXXX-
>d_pines[idx].d_lEntrada[i].d_voltaje)){
                                                        nenb = npos = ((XXPalXXXX-
>d_pines[idx].d_lEntrada[i].d_palabras[j].d_palabra & 0x0700)>>0x0008);
```

```
                                                enable_d =0;enable_d=1; enable_d
<<= npos;
                                        }
                                        else if (((npin>12) && (npin<21)) &&
(voltaje == XXPalXXXX->d_pines[idx].d_lEntrada[i].d_voltaje)){
                                                nenb = npos =
(((XXPalXXXX->d_pines[idx].d_lEntrada[i].d_palabras[j].d_palabra) &
0x0700)>>0x0008);
                                                if(nenb<4){enable_d = 1;
enable_d <<= (npos + 4);}
                                                else {enable_d = 1; enable_d
<<= (npos - 4);}
                                                nenb += 4;
                                        }
                                        else if (((npin==12) || (npin==11)) &&
(voltaje == XXPalXXXX->d_pines[idx].d_lEntrada[i].d_voltaje)){
                                                nenb = npos =
(((XXPalXXXX->d_pines[idx].d_lEntrada[i].d_palabras[j].d_palabra) &
0x0700)>>0x0008);
                                                enable_d = 1; enable_d <<=
(npos - 4);
                                                nenb += 4;
                                        }


                                        // para datos dio..
                                        npos   = (XXPalXXXX-
>d_pines[idx].d_lEntrada[i].d_palabras[j].d_palabra & 0x0007);
                                        dato_d = ((XXPalXXXX-
>d_pines[idx].d_lEntrada[i].d_palabras[j].d_palabra & 0x0040)>>0x0006);
                                        if (npos!=0) dato_d <<= npos;

                                        npos1   = ((XXPalXXXX-
>d_pines[idx].d_lEntrada[i].d_palabras[j].d_palabra & 0x0038)>>0x0003);
                                        dato_d1 = ((XXPalXXXX-
>d_pines[idx].d_lEntrada[i].d_palabras[j].d_palabra & 0x0080)>>0x0007);
                                        if (npos1!=0) dato_d1 <<= npos1;
                                        if (npin<10 || npin>12) ad_enable = D_PC;
                                        else if (npin == 11 || npin == 12) ad_enable =
D_PB;
                                        if((npin==13)&&(npos1!=7))ad_enable =
D_PB;


                                        int f_sum1=VERDADERO;
```

```
FALSO;

FALSO;

FALSO;

FALSO;

FALSO;

FALSO;

FALSO;

FALSO;

FALSO;

FALSO;

FALSO;

FALSO;

FALSO;

1]|=dato_t;
```

```
int f_sum = VERDADERO;
dato_t=0;

if ((npos==7) && (npin==1))  f_sum =

if ((npos1==7) && (npin==1))  f_sum1 =

if ((npos1==5) && (npin==11)) f_sum1 =

if ((npos==5) && (npin==11)) f_sum =

if ((npos==6) &&(npin==13)) f_sum =

if ((npos1==3) &&(npin==13)) f_sum1 =

if ((npos==4)&&(npin==15)) f_sum =

if ((npos1==3)&&(npin==15)) f_sum1 =

if ((npos1==3) && (npin==17)) f_sum1 =

if ((npos==6) && (npin==17)) f_sum =

if ((npos1==7) && (npin==19)) f_sum =

if ((npos==6) && (npin==19)) f_sum =

if ((npos1==1) && (npin==19)) f_sum1 =

if ((npos1==7) && (npin==20)) f_sum1 =


if (f_sum1) dato_t |= dato_d1;
if (f_sum) dato_t |= dato_d;
if(j==0) aux=enable_d;

if((enable_d==aux)&&(j!=0))dato_t|=stop;
if(j==1)&&(aux!=enable_d))aux=enable_d;
//if((enable_d==aux)&&(j!=0))dato_t|=stop;
//depurar el pin 1*****
if((voltaje==0)&&(npin==1)) vol_cero[npin-

if((npos==4)&&(voltaje==5)&&(npin==1)){
        dato_t<<=1;
        dato_t^=vol_cero[npin-1];
}
```

```
        if((npos1==5)&&(voltaje==12)&&(npin==1)){
                                                dato_t>>=1;
                                                dato_t^=vol_cero[npin-1];

                                        }
                                        if((npos==6)&&(voltaje==21)&&(npin==1))
{
                                                dato_t=dato_d1;
                                                dato_t>>=1;
                                                dato_t^= vol_cero[npin-1];
                                        }

                                        //depurar el pin 11*******

                                        //depurar el pin 12****
                                        if((voltaje==0)&&(npin==12))
vol_cero[npin-1]=dato_t;

        if((npos1==5)&&(voltaje==12)&&(npin==12)){
                                                dato_t>>=2;
                                                dato_t^=vol_cero[npin-1];

                                        }

        if((npos==3)&&(voltaje==99)&&(npin==12)){
                                                dato_t<<=2;
                                                dato_t^=vol_cero[npin-1];
                                        }
                                        //depurar pin 12
                                        //depurar pin 13
                                        if((voltaje==0)&&(npin==13)&&(j==0))
vol_cero[npin-1]=dato_t;
                                        if((voltaje==0)&&(npin==13)&&(j!=0))
vol_cero[npin-1]=dato_t;

        if((npos1==6)&&(voltaje==12)&&(npin==13))dato_t&=vol_cero[npin-1];
                                        if((voltaje==99)&&(npin==13)){
                                                dato_t>>=1;
                                                dato_t^=vol_cero[npin-1];
                                        }
                                        //pin 13

                                        //depurar el pin 14
                                        if((voltaje==0)&&(npin==14))
vol_cero[npin-1]=dato_t;
```

```
if((npos1==4)&&(voltaje==12)&&(npin==14)){
                                        dato_t>>=1;
                                        dato_t^=vol_cero[npin-1];
                                    }

if((npos==3)&&(voltaje==99)&&(npin==14)){
                                        dato_t<<=1;
                                        dato_t^=vol_cero[npin-1];
                                    }
                                    //pin 14

                                    //depurar el pin 15
                                    if((voltaje==0)&&(npin==15)&&(j==0)){
                                        vol_cero[npin-1]=dato_t;
                                        aux2=aux;
                                    }

if((voltaje==0)&&(npin==15)&&(aux2!=aux)) vol_cero[npin-1]=dato_t;

if((npos1==1)&&(voltaje==99)&&(npin==15)){
                                        dato_t>>=1;

                                        dato_t^=vol_cero[npin-1];
                                    }

                                    //pin1 15
                                    //depurar el pin 16
                                    if((voltaje==0)&&(npin==16))vol_cero[npin-
1]=dato_t;
                                    if((voltaje==12)&&(npin==16)){
                                        dato_t>>=1;
                                        dato_t^=vol_cero[npin-1];
                                    }
                                    if((voltaje==99)&&(npin==16)){
                                        dato_t<<=1;
                                        dato_t^=vol_cero[npin-1];
                                    }
                                    //pin 16
                                    //depurar 17
                                    if((voltaje==0)&&(npin==17)&&(j==0)){
                                        vol_cero[npin-1]=dato_t;
                                        aux2=aux;
                                    }

if((voltaje==0)&&(npin==17)&&(aux2!=aux)&&(j!=0))vol_cero[npin-1]=dato_t;
```

```
        if((npos==0)&&(npos1==1)&&(voltaje==12)&&(npin==17)){
                                        dato_t<<=1;
                                        dato_t^=vol_cero[npin-1];
                                    }

        if((voltaje==99)&&(npin==17)&&(npos==2)&&(npos1==1)){
                                        dato_t<<=1;
                                        dato_t^=vol_cero[npin-1];
                                    }

                                    //pin17
                                    //depurar pin18
                                    if((voltaje==0)&&(npin==18))vol_cero[npin-
1]|=dato_t;
                                    if((voltaje==99)&&(npin==18)){
                                        dato_t<<=3;
                                        dato_t^=vol_cero[npin-1];

                                    }
                                    if((voltaje==12)&&(npin==18)){
                                        dato_t>>=3;
                                        dato_t^=vol_cero[npin-1];
                                    }
                                    //pin18
                                    //depurar pin 19
                                    if((voltaje==0)&&(npin==19)&&(j==0)){
                                        vol_cero[npin-1]|=dato_t;
                                        aux2=aux;
                                    }

      if((voltaje==0)&&(npin==19)&&(aux2!=aux))vol_cero[npin-1]=dato_t;
                                    if((voltaje==99)&&(npin==19)){
                                        dato_d1>>=1;
                                        dato_t=dato_d1;
                                        dato_t^=vol_cero[npin-1];
                                    }
                                    //pin 19
                                    //depurar pin 20
                                    if((voltaje==0)&&(npin==20))vol_cero[npin-
1]|=dato_t;
                                    if((voltaje==5)&&(npin==20)){
                                        dato_d1>>=1;
                                        dato_t=dato_d1;
                                        dato_t^=vol_cero[npin-1];
```

```
                                        }
                                        if((voltaje==12)&&(npin==20)){
                                                dato_t>>=1;
                                                dato_t^=vol_cero[npin-1];

                                        }
                                        if((voltaje==21)&&(npin==20)){
                                                dato_t<<=2;
                                                dato_t^=vol_cero[npin-1];
                                        }



                                        //pin 20

                                        Pines->wda[k].wd = dato_t;
                                        Pines->wda[k].ad = ad_datos;
                                        stop=Pines->wda[k].wd;
                                        Pines->wda[k+1].wd = enable_d;
                                        Pines->wda[k+1].ad = ad_enable;
                                        Pines->wda[k].ne = nenb;
                                        Pines->wda[k+2].wd = 0x0000;
                                        Pines->wda[k+2].ad = ad_enable;


                                }
                        }
                }
                break;
        case TXNPINPAL20:
                break;
        }

return VERDADERO;

}
```

CONCLUSIONES

1.- El software desarrollado hasta este momento se desarrollo utilizando estructuras por que son los drivers, este software nunca de devera modificar para que funcione el proyecto, lo que permitirá que posteriormente se pueda desarrollar una interfaz visual utilizando Clases en Visual C++ versión x.x. tomando como base el hardware y los drivers desarrollados. Lo cual no limitara en este momento al usuario para utilizar el programador, pero se le exigira un nivel de conocimiento mayor por que debera conocer la estructura interna de los PALs.

RECOMENDACIONES

1.- Es recomendable que se desarrolle en algún trabajo de ingeniería una interfaz de alto nivel para que utilice los drivers desarrollados hasta este momento es este trabajo de graduación. Y asi se pueda realizar la utilización de este trabajo de graduación en aplicaciones de Electronica Digital con utilización de PAL's.

BIBLIOGRAFIA

1.-Aprendiendo Visual C++2 En 21 días
Tercera Edición
PRETINCE HALL HISPANOAMERICANA S.A

2.-Progrese con Visual C++
David J. Kruglinski
McGraw-Hill

3.-Como Programar en C
H.M DEITEL/ P.J:DEITEL
Segunda Edición
PRETINCE HALL HISPANOAMERICANA S.A

4.- La Ayuda incorparada del lenguaje Visual C++5.0