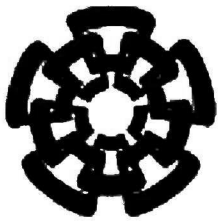
xx (101578.1)



CINVESTAV

Centro de Investigación y de Estudios Avanzados del IPN
Unidad Guadalajara

ARQUITECTURA Y DISEÑO DE UNA PLATAFORMA PARA EL DESARROLLO DE APLICACIONES PARA LA EVALUACIÓN DE CIRCUITOS DE ALTA INTEGRACIÓN

Tesis que presenta:
ARTURO JOSE GARCIA GARCIA

Para obtener el grado de:
Maestro en Ciencias

En la especialidad de:
Ingeniería Eléctrica

Guadalajara, Jal., Noviembre 2001

ARQUITECTURA Y DISEÑO DE UNA PLATAFORMA PARA EL DESARROLLO DE APLICACIONES PARA LA EVALUACIÓN DE CIRCUITOS DE ALTA INTEGRACIÓN

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:
ARTURO JOSE GARCIA GARCIA
Ingeniero en Computación
UDG, 1994-1998

Becario del CONACYT, Expediente # 129244

Director de Tesis:
Dr. Manuel Edgardo Guzmán Rentería

CINVESTAV del IPN Unidad Guadalajara, Noviembre del 2001

AGRADECIMIENTOS

Agradezco mi asesor de tesis el Dr. Manuel Guzmán Rentería por el apoyo brindado en el desarrollo de esta tesis.

Agradezco a Jesús Palomino por haberme dado la oportunidad de implementar mi tesis en TDCOM S.A. de C.V.

Agradezco a mi esposa por su amor y apoyo.

Agradezco a Ramón Parra Michel por su amistad, confianza y todo el apoyo moral que me ha brindado durante toda mi carrera profesional.

Agradezco al CONACYT por haberme otorgado una beca para continuar con mis estudios de maestría.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

INTRODUCCIÓN

1 Introducción

Los sistemas de transmisión de datos son una parte esencial en la vida industrial y estos sistemas están tan desarrollados que cualquier empresa por pequeña que sea, puede instalar, de manera sencilla, una red de computadoras donde puede comunicar a velocidades aceptables: datos, voz e incluso imágenes. Lo anterior ha permitido que las empresas que han integrado esta tecnología tengan una gran capacidad de procesamiento de información llegando al punto de poder ampliar su mercado a niveles internacionales. El creciente flujo de información trae como consecuencia la demanda de sistemas de comunicación que sean más veloces en su procesamiento de información, robustos y que su integración ocupe el menor espacio posible.

Atendiendo a esos requerimientos las empresas que se dedican al desarrollo de circuitos de alta integración de comunicaciones, cuentan con varios grupos de desarrollo que tienen como tarea mejorar las tecnologías existentes y desarrollar otras nuevas, siendo una de sus metas presentar un nuevo dispositivo de comunicaciones por lo menos cada año. Los nuevos sistemas antes de salir al mercado tienen que pasar por una etapa de pruebas de funcionalidad y verificación del diseño en laboratorio, y posteriormente, una evaluación exhaustiva hecha por el fabricante de sistemas de comunicación.

Para implementar estas etapas es conveniente desarrollar un sistema de evaluación, el cual está integrado por una tarjeta, donde reside el circuito integrado a evaluar, la tarjeta se conecta a una computadora ya sea por un bus o un puerto. En la computadora reside un software que se usa para configurar y monitorear los registros del circuito integrado. Hay que notar que algunos de los circuitos de alta integración pueden tener alrededor de quinientos registros o más. Esta es una de las razones para tener software con cierto grado de inteligencia que ayude a la configuración y monitoreo de registros. Lo anterior es especialmente importante cuando se depura y aprende el uso del circuito integrado. En etapas posteriores se puede utilizar el software para crear manejadores de dispositivo y en

etapas anteriores, cuando se simula como parte del diseño del circuito integrado, se puede utilizar para configurarlo.

El sistema de evaluación mencionado puede ser una de las partes claves para que un circuito integrado de comunicaciones que proponga una nueva tecnología, o esté mejorando una ya existente, tenga éxito en el mercado ya que un buen diseño de este sistema de evaluación puede permitir a los clientes potenciales configurar y conocer todas las funcionalidades de un circuito integrado de comunicaciones de manera sencilla sin invertir mucho tiempo en comprender su estructura interna.

El propósito de esta tesis, es proponer e implementar una arquitectura de software, portable y extensible, para desarrollar sistemas para la evaluación de circuitos integrados de comunicaciones.

El sistema desarrollado basado en a la arquitectura propuesta tiene como alcance apoyar a los ingenieros en la etapa de prueba en el laboratorio y a los clientes al momento de la evaluación del circuito integrado.

Esta tesis está organizada como sigue: en el capítulo dos se presenta la descripción general del sistema para la evaluación de circuitos integrados, en el capítulo tres se presentan los requerimientos del sistema para la evaluación de circuitos integrados, en el capítulo cuatro se presenta la arquitectura propuesta, en el capítulo cinco se presenta el diseño de un módulo de un sistema de evaluación de circuitos integrados siguiendo la arquitectura propuesta y en el capítulo seis se presentan las conclusiones referentes a esta tesis.

CAPÍTULO 2

DESCRIPCIÓN GENERAL DEL SISTEMA

2 Descripción general del sistema

En la figura 1 se muestra un diagrama a bloques del sistema de evaluación. Aquí aparece la tarjeta de evaluación que es donde reside el circuito integrado sujeto a evaluación, también aparece una computadora personal que es donde reside el software de configuración y monitoreo del circuito integrado. No profundizaremos en el diseño de la tarjeta de evaluación, sólo mencionaremos que cuenta con una interfaz serie o un bus a través del cual se conecta a la computadora personal. En la tarjeta de evaluación reside un microcontrolador o un dispositivo programable que es capaz de recibir y enviar comandos a la computadora personal. Para la comunicación entre la computadora personal y la tarjeta de evaluación se usan protocolos de comunicación que tampoco tocamos aquí. También se conectan a la tarjeta dispositivos de análisis de comunicaciones. No profundizaremos en la tarea que desempeñan estos dispositivos, sólo mencionaremos algunas de sus funciones a lo largo de esta tesis cuando así se requiera. Algunas de las tareas comunes de estos dispositivos de análisis son por ejemplo, recibir y verificar las tramas de datos que envía el circuito sujeto a evaluación, de esta forma podemos verificar que el circuito transmite las tramas tal y como se configuraron en el software de evaluación, esta trama posteriormente puede ser transmitida de vuelta al circuito integrado, en esta retransmisión el analizador puede insertar errores en las tramas, de esta forma se podrá monitorear desde el software de evaluación la recepción de esas tramas y ver como responde el circuito sujeto a evaluación a tramas con errores; ésta es una prueba muy común en la evaluación de sistemas de comunicación. Es importante mencionar que existen librerías de software que permiten conectar, configurar y controlar los dispositivos de análisis; éstas son comúnmente distribuidas por los fabricantes de los dispositivos de análisis.

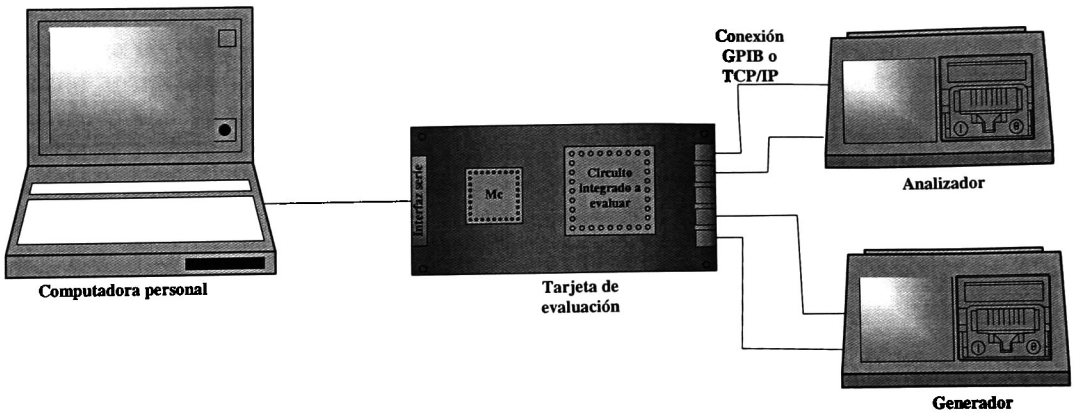


Figura 1. Diagrama a bloques del sistema de evaluación.

CAPÍTULO 3

REQUERIMIENTOS DEL SISTEMA

3 Requerimientos del sistema

3.1 Módulos del software

El software, desde la perspectiva del usuario, está dividido en cuatro módulos que son:

- Módulo de control y monitoreo
- Módulo de interpretación de código script
- Módulo de manejador de dispositivo
- Módulo de protocolo de comunicación con el hardware

En la figura 2 se muestran los módulos en los que debe estar dividido el software, más adelante se explica con más detalle la relación entre los módulos.

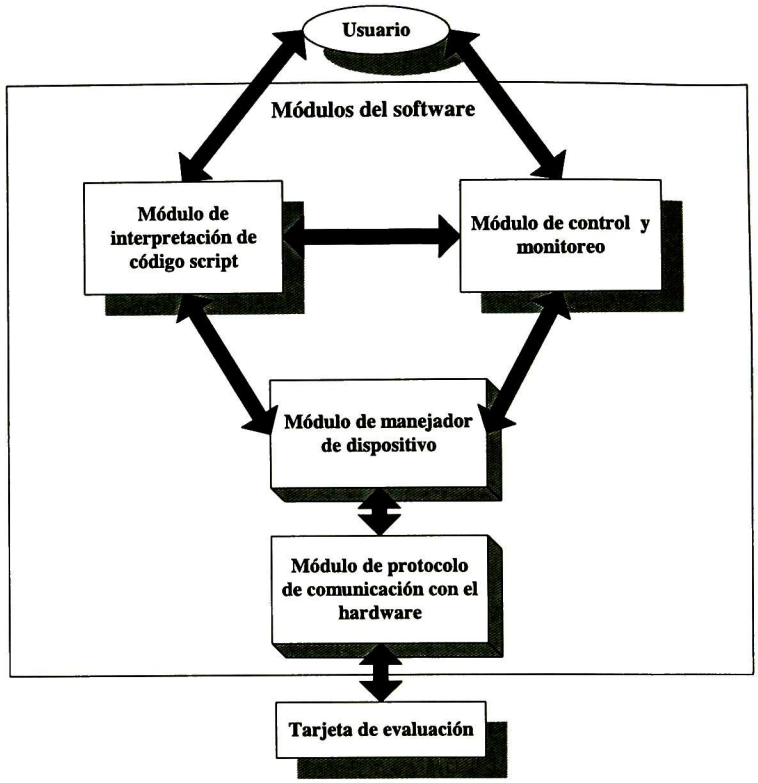


Figura 2. Módulos del software.

3.1.1 Requerimientos del módulo de control y monitoreo

El módulo de control y monitoreo es un ingrediente muy importante del sistema para la evaluación de circuitos integrados, desde este módulo el usuario interactuará con el hardware que se va a evaluar. Con la ayuda de este módulo el usuario podrá configurar, monitorear y simular algunas funcionalidades del hardware, entre otras cosas. En esta sección se describirán todas las funcionalidades que debe tener este módulo.

En la figura 3 se muestra un diagrama de caso de uso[1] que representa el módulo de control y monitoreo.

[1] Los casos de uso son una herramienta importante del análisis de requerimientos. Éstos permiten delimitar el dominio de un módulo de software a partir de un proceso.

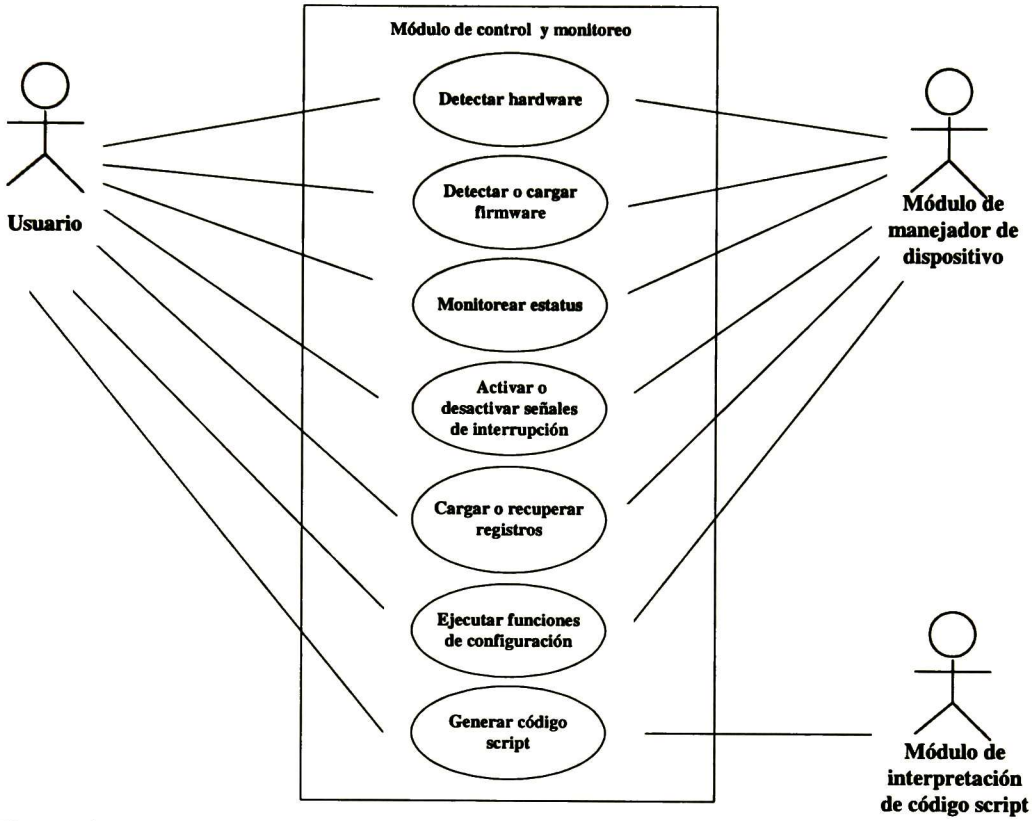


Figura 3. Diagrama de caso de uso que representa el módulo de control y monitoreo.

En el módulo de control y monitoreo se debe implementar lo siguiente:

a) Interacción con el usuario
b) Interacción con el módulo de interpretación de código script
c) Interacción con el módulo de manejador de dispositivo

A continuación explicamos con detalle las funciones mencionadas.

3.1.1.1 Interacción con el usuario

En la figura 2, presentada anteriormente, podemos ver que existe una interacción entre el usuario y el módulo de control y monitoreo, para que exista esta comunicación, el módulo de control y monitoreo debe proporcionar al usuario el conjunto de funciones para interactuar con la tarjeta de evaluación. Estas funciones son las siguientes:

- **Detectar e identificar el hardware** – El módulo de control y monitoreo debe proporcionar al usuario una opción para detectar la presencia o ausencia del hardware con el que va a trabajar, así como la identificación del mismo, consulte el apéndice A para una descripción más detallada de este proceso.

- **Detectar y cargar el firmware del microcontrolador que reside en la tarjeta de evaluación** – El módulo de control y monitoreo debe proporcionar al usuario una opción para cargar firmware en el microcontrolador que reside en la tarjeta de evaluación. Antes de cargar un firmware se revisa si previamente se ha cargado firmware, en caso de que se haya hecho una carga se indica cuál es la versión de lo cargado. Esta funcionalidad le permite al usuario saber si el firmware ya ha sido cargado, o bien, remplazar la versión existente del firmware en el microcontrolador, consulte el apéndice A para más detalles.

- **Presentar al usuario el mapa completo de los registros del hardware** – El módulo de control y monitoreo debe proporcionar al usuario, el mapa completo de todos los registros del hardware que va a configurar y monitorear, para propósito de claridad y entendimiento del circuito bajo evaluación, sus registros deben ser clasificados en:

  a) Registros de configuración
  b) Registros de estatus
  c) Registros de interrupción
  d) Registros con información de desempeño

- **Monitorear periódicamente los registros de estatus** – El módulo de control y monitoreo debe proporcionar procedimientos para la obtención periódica de los valores que contienen los registros de estatus como por ejemplo, registros de alarmas, sincronización de señales, etc, además debe proporcionar al usuario la capacidad de guardar cada cambio en los registros de estatus en una base de datos para después poder generar estadísticas.

- **Activar o desactivar las señales de interrupción** – El módulo de control y monitoreo debe proporcionar al usuario la opción de activar y desactivar las señales de interrupción que vienen de la tarjeta de evaluación.

- **Cargar o recuperar la información de los registros con información de desempeño** – Los registros con información de desempeño usualmente guardan datos estadísticos o de desempeño que el circuito recibe o calcula. La información leída de estos registros se guarda en una base de datos. También se debe tener la capacidad de poder cargar a estos registros información leída de un archivo; la información que contienen estos registros tiene un formato definido por el diseñador del hardware.

- **Cargar o recuperar la información de los registros de configuración** – El módulo de control y monitoreo debe proporcionar al usuario la capacidad y las validaciones adecuadas para poder leer y modificar la información de los registros de configuración.

- **Proporcionar un conjunto de funciones de configuración para el hardware que se está evaluando** – Por medio de estas funciones el usuario configura algunos registros del hardware que se está evaluando. Por cada módulo de hardware, en el circuito bajo evaluación, existe un grupo de funciones en el módulo de control y monitoreo que se usan para modificar los registro que lo configuran.

En la figura 4 se muestra, a manera de ejemplo, un diagrama a bloques que ilustra esta idea.

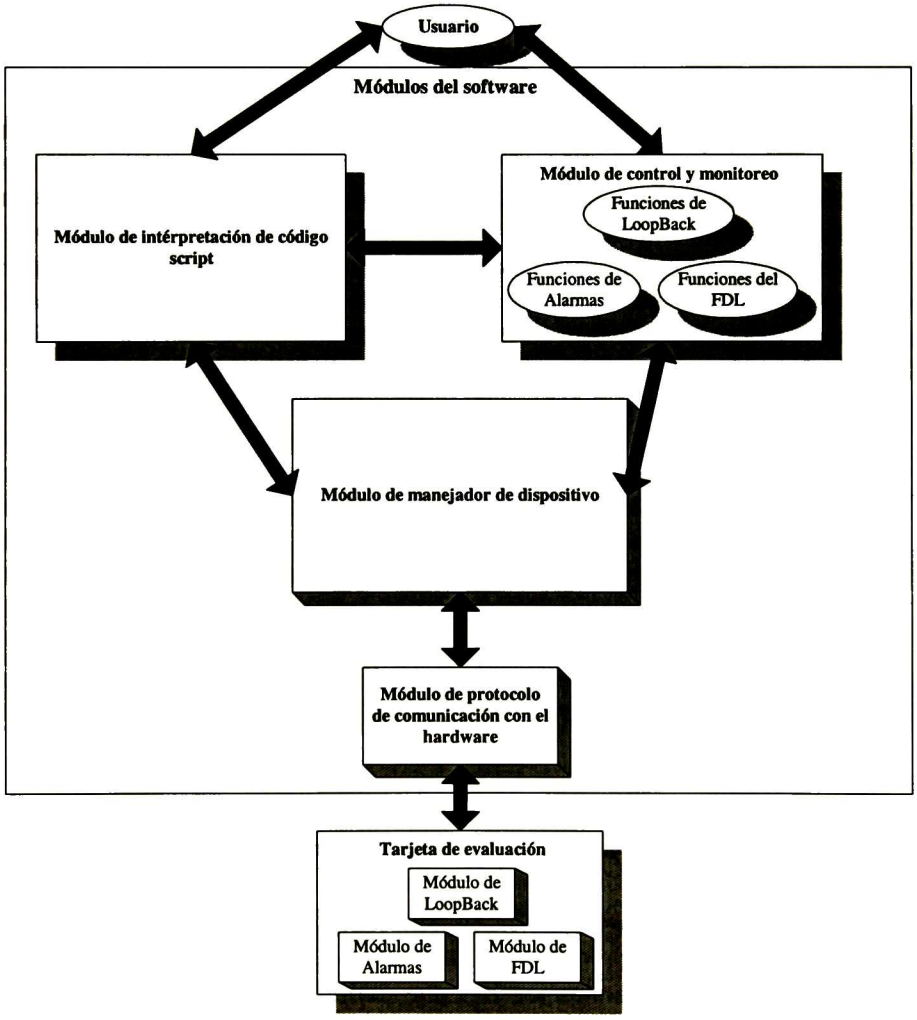


Figura 4. El módulo de control y monitoreo debe presentar al usuario una imagen espejo del hardware.

- **Guardar en archivos las configuraciones particulares del hardware** – El módulo de control y monitoreo debe proporcionarle al usuario la opción de poder traducir la configuración de un módulo o grupo de módulos del hardware que está evaluando a una secuencia de llamadas a procedimientos en el lenguaje C, para más detalles consulte el apéndice B. Llamamos a esta secuencia de procedimientos un *script de configuración*. El script de configuración se debe guardar de manera persistente en un archivo. Un archivo puede contener varios scripts de configuración. Es importante notar que el orden de los scripts en un archivo de configuración es significativo pues los circuitos de alta integración normalmente no aceptan secuencias de configuración arbitrarias.

3.1.1.1.1 Modos de operación para el usuario

El módulo de control y monitoreo debe proporcionar al usuario dos modos de operación:

- **Modo en línea** – Este modo de operación ocurre cuando la aplicación detecta e identifica adecuadamente al hardware. En este modo de operación se pueden ejecutar las funciones de detección y carga de firmware, monitoreo de estatus, manejo de señales de interrupción, carga y recuperación de registros, ejecución de funciones de configuración y generación de scripts.

- **Modo fuera de línea** – La aplicación entra en modo fuera de línea cuando el hardware no está conectado al CPU (Central Processig Unit – Unidad Central de Procesamiento), o cuando el usuario deliberadamente desea no conectar el hardware con la aplicación para entrar en este modo de operación. En este modo de operación la aplicación le permite al usuario crear gráficamente configuraciones y guardarlas en script de configuración. Estos scripts pueden ser utilizados por el sistema aquí descrito o por algún otro software.

En el apéndice A se muestra un diagrama de flujo de los modos de operación del sistema para la evaluación de circuitos integrados.

3.1.1.2 Interacción con el módulo de interpretación de código script

Para que exista la interacción, que se mostró en la figura 2, entre el módulo de interpretación de código script y el módulo de control y monitoreo, este último debe conectarse con la interfaz de comunicación que define el módulo de interpretación de código script. Por medio de esta interfaz el módulo de interpretación de código script, le

comunica al módulo de control y monitoreo las actualizaciones de la información hechas a los registros del hardware, después de haber ejecutado un script de configuración, con

el propósito de que el módulo de control y monitoreo presente al usuario de manera gráfica los datos actualizados.

3.1.1.3 Interacción con el módulo de manejador de dispositivo

Para que exista la interacción, que se mostró en la figura 2, entre el módulo de manejador de dispositivo y el módulo de control y monitoreo, el módulo de control y monitoreo debe conectarse con la interfaz de comunicación que define el módulo de manejador de dispositivo. El módulo de control y monitoreo envía y recibe todas las peticiones al hardware por medio de esta interfaz.

3.1.2 Requerimientos del módulo de interpretación de código script

Al módulo de interpretación de código script se le indica un archivo con scripts de configuración que decodifica y ejecuta. Los scripts son una manera muy conveniente de configurar un circuito de alta integración, pues permite un gran ahorro de esfuerzo en la configuración como ahora explicaremos. Normalmente las diferentes configuraciones del circuito tienen mucho en común y el uso de los scripts evita que cada vez se tenga que configurar en detalle. Una vez que se hizo la configuración de una parte del circuito, el código que realiza esa configuración se puede reusar. El fabricante de circuitos integrados le proporciona al cliente una serie de scripts los cuales le permiten configurar su circuito de una manera más amigable.

En la figura 5 se muestra un diagrama de caso de uso, que representa el módulo de interpretación de código script.

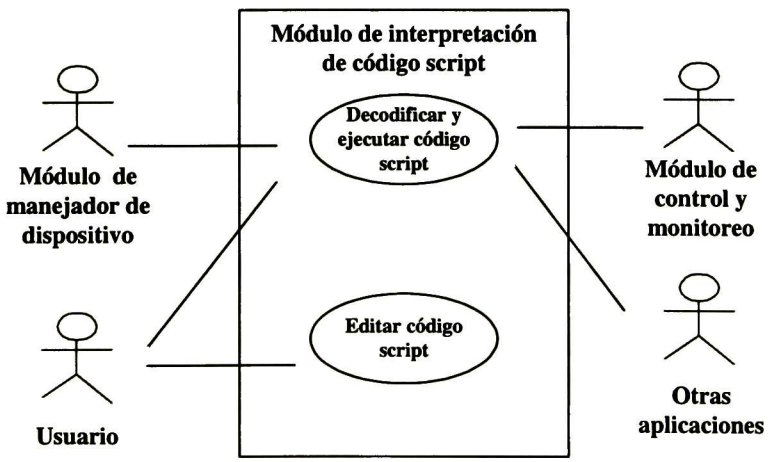


Figura 5. Diagrama de caso de uso, que representa el módulo de interpretación de código script.

En el módulo de interpretación de código script se debe implementar lo siguiente:

a) Interacción con el usuario
b) Interacción con el módulo de control y monitoreo
c) Interacción con el módulo de manejador de dispositivo

A continuación explicamos con detalle las funciones mencionadas.

3.1.2.1 Interacción con el usuario

Para que exista la interacción, que se mostró en la figura 2, entre el usuario y el módulo de interpretación de código script, este último debe proporcionar al usuario, en un ambiente gráfico, las siguientes funciones:

- **Decodificar y ejecutar scripts** - El intérprete decodifica y ejecuta los scripts en el orden en que aparecen. Como resultado de la ejecución de un script se logra la configuración de uno o varios grupos de módulos del hardware que se están evaluado, esta configuración también se ve reflejada en el módulo de control y monitoreo y en las aplicaciones que estén conectadas con el módulo de interpretación de código script, esta conexión con otras aplicaciones será explicada más adelante.

- **Editar scripts** – El módulo de manejador de interpretación de código script debe proporcionarle al usuario la capacidad de poder editar y guardar los scripts.

La estructura del lenguaje script desarrollado en este trabajo es bastante sencilla, está compuesto de comandos de alto nivel, que son ordenados por lotes. Este lenguaje no contiene instrucciones de control de flujo ni condicionales, tampoco soporta el manejo de variables ni creación de procedimientos. En el apéndice B se presentan los requerimientos y el análisis para el desarrollo de un lenguaje script más robusto, el cual está basado en la estructura del lenguaje de programación C.

3.1.2.2 Interacción con el módulo de control y monitoreo

Para que exista la interacción, que se mostró en la figura 2, entre el módulo de control y monitoreo y el módulo de interpretación de código script, este último debe proporcionar una interfaz de comunicación por medio del cual el módulo de control y monitoreo pueda conectarse. A través de esta interfaz se puede conectar cualquier aplicación de usuario.

Cuando el módulo de interpretación de código script ejecuta un script, la configuración hecha al hardware que se está evaluando debe verse reflejada en las aplicaciones que estén conectadas al intérprete de código script, ésto se logra con la interfaz de comunicación.

3.1.2.3 Interacción con el módulo de manejador de dispositivo

Para que exista la interacción, que se mostró en la figura 2, entre el módulo de manejador de dispositivo y el módulo de interpretación de código script, este último debe conectarse con la interfaz de comunicación que define el módulo de manejador de dispositivo. El módulo de interpretación de código script envía y recibe todas las peticiones del hardware por medio de esta interfaz de comunicación.

3.1.3 Requerimientos del módulo de manejador de dispositivo

El módulo de manejador de dispositivo es el módulo donde se implementan los comandos de comunicación y configuración del hardware. Este módulo manejador de dispositivo debe estar diseñado para poder portar su implementación a otros sistemas operativos, consultar referencias [8] y [18], facilitando así la implementación de manejadores del circuito integrado que se está evaluando.

El módulo de manejador de dispositivo ejecuta las peticiones enviadas de los módulos de control y monitoreo e interpretación de código script, estas peticiones son recibidas por el manejador de dispositivo a través de una interfaz de comunicación definida por éste. También por esta misma interfaz, el módulo manejador de dispositivo envía los acuses de recibo de las peticiones y la información devuelta por una petición. Por esta interfaz se pueden conectar otras aplicaciones de usuario.

El módulo de manejador de dispositivo manda las configuraciones y peticiones al hardware a través del módulo de protocolo de comunicación con el hardware, esta comunicación se da por medio de una interfaz de comunicación definida por el módulo de protocolo de comunicación con el hardware.

En la figura 6 se muestra un diagrama de un caso de uso, que representa el manejador de dispositivo.

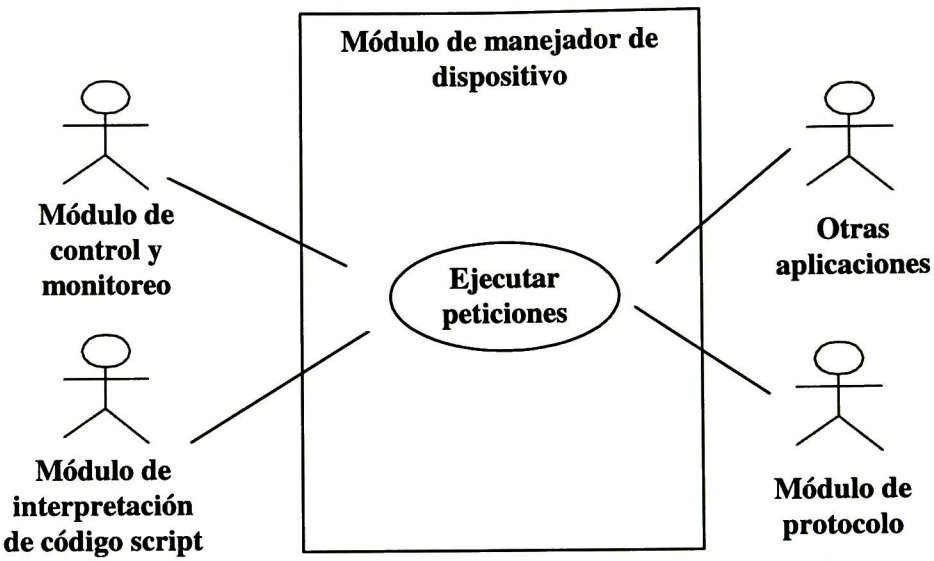


Figura 6. Diagrama de caso de uso, que representa el manejador de dispositivo.

En el módulo manejador de dispositivo debe se debe implementar lo siguiente:

a) Interacción con el módulo de control y monitoreo
b) Interacción con el módulo de protocolo de comunicación con el hardware

A continuación explicamos con detalle las funciones mencionadas.

3.1.3.1 Interacción con el módulo de control y monitoreo

Para que exista la interacción, que se mostró en la figura 2, entre el módulo de manejador de dispositivo y el módulo de control y monitoreo, este último debe conectarse con la interfaz de comunicación que define el módulo de manejador de dispositivo.

Además de la interfaz de comunicación, este módulo debe contener el mapa completo de las direcciones del hardware y debe implementar todos los comandos para la configuración y comunicación con la tarjeta. Estos comandos deben estar encapsulados basándose en los módulos que tenga el hardware, esto quiere decir, que el código de objetos del manejador de dispositivo debe hacer una imagen espejo del hardware. En la figura 7 se muestra, a manera de ejemplo, un diagrama a bloques que ilustra esta idea.

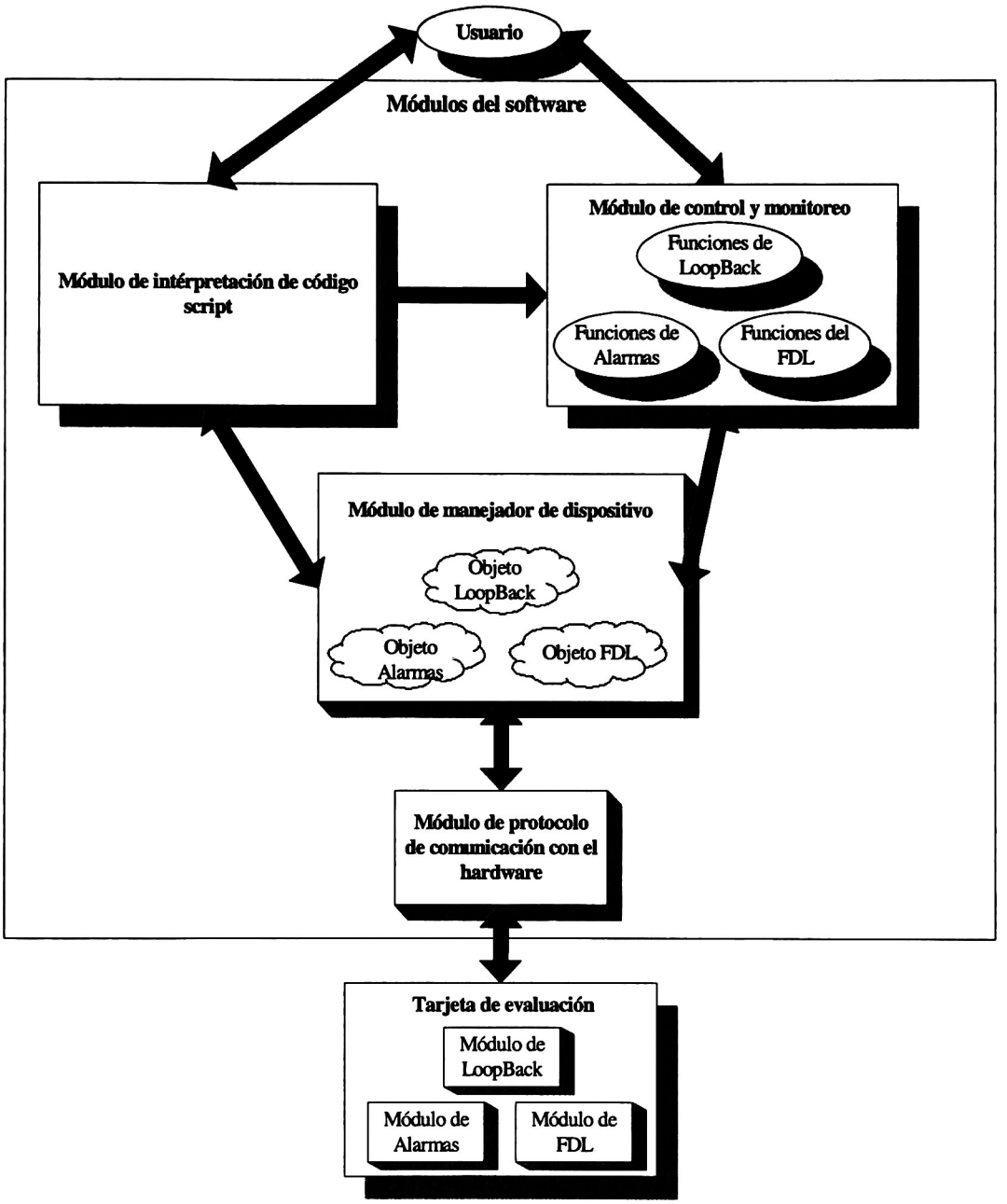


Figura 7. El código del manejador de dispositivo debe hacer una imagen espejo del hardware.

3.1.3.2 Interacción con el módulo de protocolo de comunicación con el hardware

Para que exista la interacción, que se mostró en la figura 2, entre el módulo de protocolo de comunicación con el hardware y el módulo de manejador de dispositivo, este último debe conectarse con la interfaz de comunicación que define el módulo de protocolo de comunicación con el hardware.

3.1.4 Requerimientos del módulo de protocolo de comunicación con el hardware

El módulo de protocolo de comunicación con el hardware debe ser compacto, eficiente y asegurar la entrega confiable de la información en ambas direcciones. En este trabajo no se profundizará en el diseño y desarrollo de este protocolo, la referencia [19] puede servir de modelo para el diseño y desarrollo de este protocolo, pero se definirá la interfaz de comunicación para que el módulo de manejador de dispositivo pueda conectarse al módulo de protocolo de comunicación con el hardware. Utilizando esta interfaz el módulo de manejador de dispositivo enviará y recibirá mensajes del hardware. El módulo de protocolo de comunicación con el hardware por su parte se asegurará que estos mensajes lleguen al hardware, también el módulo de protocolo de comunicación con el hardware recibirá los mensajes enviados del hardware y los entregará al módulo de manejador de dispositivo.

En la figura 8 se presenta un escenario que ilustra la ruta que sigue el procesamiento de un mensaje en el sistema para la evaluación de circuitos integrados, desde el momento que un usuario le pide al módulo de control y monitoreo que ejecute el mensaje, hasta que éste es recibido y procesado por el hardware.

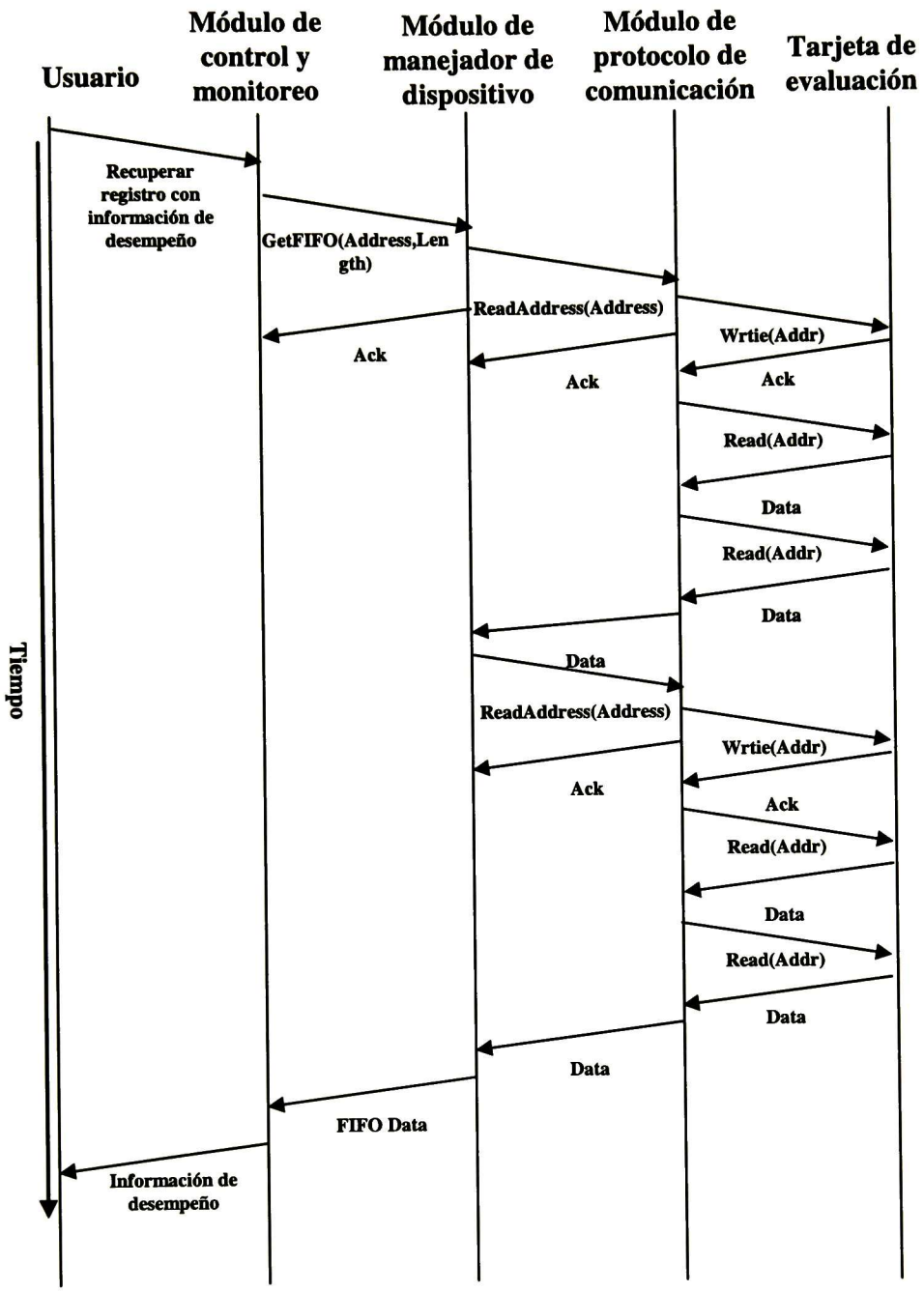


Figura 8. Escenario del procesamiento de un mensaje en el sistema de evaluación de circuitos integrados.

CAPÍTULO 4

ARQUITECTURA

4 Arquitectura

Teniendo en cuenta los requerimientos expuestos en el capítulo anterior, la arquitectura fue conceptualizada a partir de una arquitectura multicapas orientada a objetos que adopta los principios básicos de un modelo clásico de una arquitectura de tres capas [11], en la figura 9 se muestra un diagrama a bloques que describe la vista clásica de una arquitectura de tres capas.

Las funciones de cada uno de estas capas son:

- **Presentación** - Ventanas, reportes, etc.
- **Lógica de aplicaciones** - Tareas y reglas que rigen el proceso.
- **Almacenamiento** - Mecanismos de almacenamiento persistente.

La tarea más importante de la arquitectura de tres capas consiste en aislar la lógica de la aplicación y en convertirla en una capa intermedia bien definida y lógica del software. En la capa de presentación se realiza relativamente poco procesamiento de la aplicación; las ventanas envían a la capa intermedia las peticiones de trabajo.

En un diseño orientado a objetos, la capa de la lógica de aplicaciones se puede descomponer en otras capas menos densas y así prescindir de la designación de arquitectura de tres capas y hablar en cambio de arquitecturas multicapas, en las cuales está implícita la capa intermedia de la lógica de aplicaciones.

Entonces la capa lógica puede estar constituida por las siguientes capas:

- **Dominio** Clases que representan los conceptos del dominio; por ejemplo un módulo del circuito integrado a evaluar.

- **Servicios** - Clases que representan los conceptos de servicio de las funciones del sistema tales como: las comunicaciones, seguridad, manejo de archivos y bases de datos.

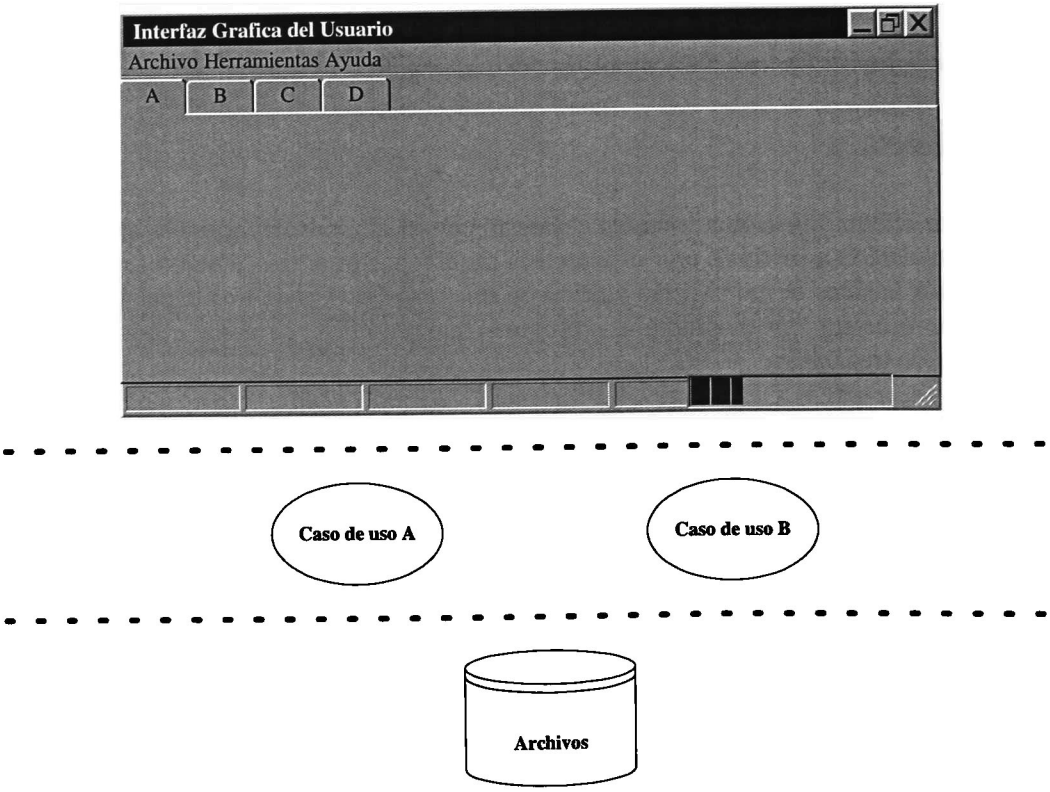


Figura 9. Vista clásica de una arquitectura de tres capas.

Es posible agregar más capas y descomponer las ya existentes; por ejemplo la capa de *servicios* puede dividirse en servicios de alto y bajo nivel. Presentar la arquitectura basándonos en un modelo multicapas como el que acabamos de explicar, presenta las siguientes ventajas:

- Aislamiento de la lógica de aplicaciones en componentes independientes susceptibles de reutilizarse después en otros sistemas
- Distribución de las capas en varios procesos
- Asignación de los diseñadores para que construyan determinadas capas

4.1 Arquitectura del sistema para la evaluación de circuitos integrados

Con los principios expuestos anteriormente, desarrollamos las unidades arquitectónicas del sistema en términos de los paquetes[1] del lenguaje UML[2]. De esta forma por medio de un paquete podemos describir los grupos de elementos o subsistemas de una capa de la arquitectura. En la figura 10 presentamos la arquitectura[3] detallada del sistema para la evaluación de circuitos integrados en términos de los paquetes UML.

[1] Un paquete es un conjunto de cualquier tipo de elementos de un modelo: clases, casos de uso u otros paquetes anidados.

[2] UML son las siglas de Unified Modeling Languaje (Lenguaje Unificado de construcción de Modelos), notación con que se construyen sistemas por medio de conceptos orientados a objetos [10] y [17].

[3] Es importante mencionar que en esta arquitectura los estratos no están acoplados en un sentido restringido como es el caso del modelo OSI (Open System Interconnection – Interconexión se Sistemas Abiertos) de 7 capas, donde los elementos de la capa N no pueden acceder a los servicios de la capa o estrato inferior N-1. Aunque tal limitación es posible, la arquitectura tiene la flexibilidad de ser de capas transparentes, donde los elementos de una capa o estrato se comunican con cualquier otra capa.

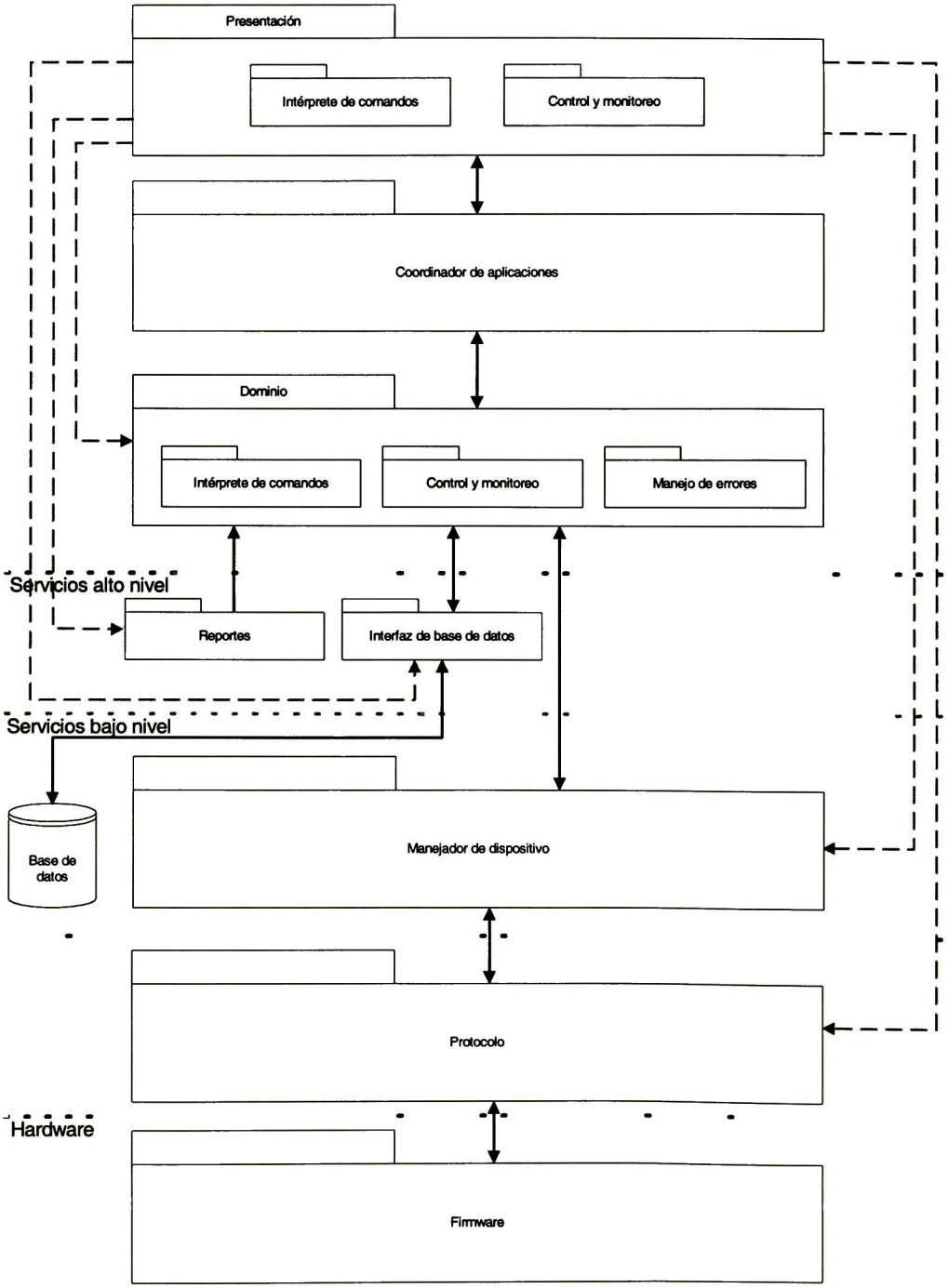


Figura 10. Arquitectura detallada del sistema para la evaluación de circuitos integrados.

La arquitectura del sistema para la evaluación de circuitos integrados está compuesta por los siguientes estratos[4]:

- Presentación
- Coordinador de aplicaciones
- Dominio
- Servicios
- Protocolo
- Hardware (tarjeta de evaluación)

La capa de servicio la podemos conceptualizar como integrada de dos subestratos donde el primero lo llamaremos servicios de alto nivel, que a su vez está compuesto por dos particiones llamadas: *reportes* e *interfaz de base de datos;* y al segundo lo llamaremos servicios de bajo nivel que está integrado también por dos particiones llamadas: *manejador de dispositivos y base de datos.*

En la figura 10, presentada anterior mente, las flechas negras representan la visibilidad entre los paquetes, las flechas grises punteadas también significan visibilidad entre paquetes, pero este tipo de visibilidad será explicado más adelante.

A continuación explicaremos con detalle cada una de las unidades arquitectónicas de esta arquitectura.

4.1.1 Capa de presentación

En la capa de presentación se implementa la interfaz gráfica de usuario, que desde la perspectiva de nuestra arquitectura, esta capa debe contener las clases que definen las ventanas, a las que llamaremos *vistas* de aquí en adelante, que se encargan de la entrada y salida sin que conserven los datos ni ofrezcan directamente la funcionalidad del sistema.

Como vimos en la figura 10, esta capa tiene dos particiones:

- **Control y monitoreo** – Esta partición contiene las vistas para configurar, monitorear y simular algunas funciones del hardware, entre otras cosas. Estas vistas deben satisfacer los requerimientos expuestos en el capítulo 3, consultar sección 3.1.1.1.

[4] Es importante mencionar que una arquitectura multicapas está compuesta de estratos y particiones. Los estratos de una arquitectura representan las capas verticales, mientras que las particiones representan la división horizontal de subsistemas relativamente paralelos a un estrato.

- **Intérprete de código script** – Esta partición contiene las vistas para editar, depurar e interpretar un código script. Estas vistas deben satisfacer los requerimientos expuestos en el capítulo 3, consultar sección 3.1.2.1.

Hay que recordar que uno los objetivos de esta tesis es presentar una arquitectura que sea portable y extensible, para lograr estos objetivos diseñamos la capa de presentación aplicando patrones de programación orientada a objetos, consulte las referencias [7], [9], [15] y [20] para más detalles sobre este tema, que tratan la portabilidad y extensibilidad de un sistema.

El primer problema que encontramos al diseñar una capa de presentación es el de definir la visibilidad de otras capas respecto a la capa de presentación. No es conveniente que haya una visibilidad directa de otras capas de una arquitectura de software con los objetos vista porque éstos se encuentran relacionados con una aplicación en particular, así de esta forma podemos reutilizar las capas que están abajo de la capa de presentación en una nueva interfaz gráfica.

El patrón que se aplica para este caso es el patrón *Separación Modelo-Vista*. Donde el modelo se refiere a la capa dominio y la vista a la capa de presentación. Este patrón establece que se deben definir las clases del modelo para que no tengan acoplamiento[5] ni visibilidad directa con respecto a las clases vista, también define que los datos de la aplicación y de la funcionalidad se conserven en las clases del modelo, no en las de la vista.

Las ventajas que ofrece este patrón de diseño son las siguientes:

- Permite desarrollar independientemente el modelo y la vista
- Reduce al mínimo el trabajo producido por los cambios de requerimientos
- Permite conectar fácilmente otras vistas a la capa de modelo, sin que ésto la afecte
- Permite vistas simultáneas y múltiples
- Permite ejecutar la capa de modelo independientemente de la vista; un ejemplo de ésto es un sistema de procesamiento de mensajes o por lotes
- Permite transportar fácilmente la capa de modelo a otro esquema de vista

Con esto explicamos el significado de las flechas grises en la figura 10, presentada anteriormente, esto quiere decir que en teoría la capa de presentación puede tener visibilidad de casi todas las capas inferiores, pero con excepción de la capa de coordinador de aplicaciones, las capas interiores no deben tener conocimiento de la capa de presentación, pero en nuestra aplicación, la visibilidad expresada con flechas grises está restringida, la capa de presentación sólo se comunica con una capa inferior.

[5] El acoplamiento es una medida de la fuerza con la que una clase está conectada a otras clases, le permite conocerlas y recurrir a ellas. Una clase con bajo acoplamiento no depende de muchas otras.

4.1.2 Capa de coordinador de aplicaciones

La capa de coordinador de aplicaciones contiene clases que tienen la responsabilidad[6] de mediar entre la capa de presentación y la de dominio. Esta capa desempeña las siguientes tareas:

- Mapear la información entre los objetos de la capa de dominio y la de presentación
- Responder a los eventos procedentes de la capa de presentación
- Abrir ventanas que muestran la información proveniente de los objetos de la capa de dominio
- Soportar la capacidad de que varias vistas muestren simultáneamente la información procedente de los objetos de la capa de dominio
- Notificarles a las vistas cuando cambie la información para que éstas actualicen la información presentada al usuario

4.1.3 Capa de dominio

La capa del dominio contiene las tareas y reglas que rigen las operaciones del sistema, también contiene las interfaces para atender las peticiones de la capa de coordinador de aplicaciones y para comunicarse con la capa de servicio, esta última contiene módulos de gran importancia como son el manejador de dispositivo, el manejador de bases de datos y la creación de reportes.

Esta capa está compuesta de tres particiones, representadas gráficamente en la figura 10, las cuales son:

- Intérprete de código script
- Control y monitoreo
- Manejo de errores

A continuación presentaremos la estructura interna de cada una de estas particiones.

4.1.3.1 Intérprete de código script

Esta partición contiene los objetos que se encargan de la edición, depuración, análisis e interpretación de código script. En la figura 11 se muestra la estructura interna de esta partición.

[6] Las responsabilidades se relacionan con las obligaciones de un objeto respecto a su comportamiento.

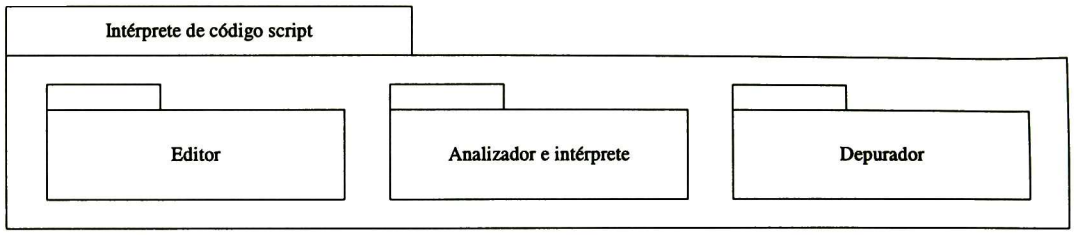


Figura 11. Estructura interna de la partición intérprete de código script.

En el apéndice B se expone el análisis para desarrollar cada una de las subparticiones de esta partición.

4.1.3.2 Control y monitoreo

Esta partición contiene los objetos que se encargan de procesar y validar las peticiones hechas por la capa de presentación a través de la capa de coordinador de aplicaciones. Las principales operaciones que se encarga de procesar y validar son:

- Detectar hardware
- Detectar y cargar firmware
- Monitorear estatus
- Activar señales de interrupción
- Ejecutar funciones de configuración
- Generar código script

También se encarga de enviar a la capa de presentación la información actualizada procedente de la capa de hardware enviada a través de la capa de manejador de dispositivo. Otra de las tareas que realiza es la comunicación con la capa de manejador de dispositivo, esta comunicación tiene entre sus funciones principales la atención de eventos y el envío de peticiones a ésta.

La estructura interna de esta partición está basada en las unidades arquitectónicas del chip en evaluación, consultar sección 3.1.1.

4.1.3.3 Manejo de errores

Esta partición se encarga del manejo de errores procedentes de las capas inferiores y los generados en la validación y procesamiento de las peticiones hechas por las capas superiores.

Las tareas que involucra el manejo de errores son las siguientes:

- Detectar el error
- Enviar un evento a través de la capa de dominio para que la capa de presentación notifique el error al usuario
- Registrar el error en una base de datos

4.1.4 Capa de servicio

La capa de servicio contiene los objetos de servicio del sistema como son, reportes, comunicaciones, manejo de base de datos, seguridad etc. En la arquitectura presentada, la capa de servicio la hemos dividido en dos subcapas: de alto y bajo nivel.

A continuación presentaremos la estructura interna de cada una de estas subcapas.

4.1.4.1 Servicios de alto nivel

La estructura interna de esta subcapa incluye los objetos que se encargan del manejo de reportes y de los objetos que manejan las bases de datos creadas por el sistema.

4.1.4.2 Servicios de bajo nivel

Esta subcapa contiene las funciones que se encargan de interactuar con el hardware, estas funciones están encapsuladas en un paquete llamado manejador de dispositivo. El manejador de dispositivo cumple con las responsabilidad del patrón agente dispositivo, el cual tiene la responsabilidad de interactuar con la capa de hardware y atender las peticiones hechas por las capas superiores.

La estructura interna del manejador de dispositivo está basada en las unidades arquitectónicas del chip en evaluación, consultar sección 3.1.3.

Es importante notar que el manejador de dispositivo no se comunica directamente con el hardware, ya que por reusabilidad es conveniente delegar las responsabilidad de definir el medio físico por el cual se comunicará con el hardware y los mecanismos para asegurar la comunicación confiable de los datos a la capa de protocolo. En esta subcapa también se incluyen las bases de datos que utiliza el sistema.

4.1.5 Capa de protocolo de comunicación

La capa de protocolo de comunicación tiene la responsabilidad de implementar la comunicación confiable con el hardware. El análisis, diseño e implementación de un protocolo de este tipo está expuesto en el apéndice C, consultar también sección 3.1.4.

4.1.6 Capa de hardware

En la capa de hardware reside el circuito integrado sujeto a evaluación y un microcontrolador que aloja un firmware que recibe y envía comandos a la computadora a través del puerto de comunicaciones.

Con ésto concluimos uno de los objetivos principales de esta tesis que es la presentación de una arquitectura de software portable y extensible, para desarrollar sistemas de evaluación de circuitos integrados de comunicaciones de alta velocidad.

En el siguiente capítulo presentaremos el diseño de un módulo del sistema de evaluación.

CAPÍTULO 5

DISEÑO DE UN MÓDULO DEL SISTEMA

5 Diseño de un módulo del sistema

El objetivo de este capítulo es presentar de manera gráfica el diseño de un módulo del sistema de evaluación. Para este propósito tomaremos el submódulo de envío y recepción de BOPs (Bit Oriented Protocol – Protocolo Orientado a Bit) del módulo FDL(Facility Data Link – Sección de Enlace de Datos) de un circuito de comunicaciones para T1[1]

5.1 Módulo FDL

Este módulo detecta y genera los mensajes trasportados en el canal de enlace de datos para tramas T1. Hay dos tipos de mensajes[2] transportados en el canal de enlace de datos, los cuales son:

- **BOP (Bit Oriented Protocol – Protocolo Orientado a Bit)** – Compuesto por repeticiones de patrones de bits llamados codewords.

- **MOP (Message Oriented Protocol – Protocolo Orientado a Mensage)** – Compuesto por mensajes usando el protocolo LAPD (Link Access Procedure on the D channel - Procedimiento de Acceso de Enlace en el canal D).

[1] La referencia [14] presenta los conceptos básicos sobre este tema.
[2] El formato de estos mensajes está definido en las referencias [1], [2] y [13].

5.2 Requerimientos

El submódulo de BOPs debe proporcionar al usuario las siguientes funciones:

- Habilitar o deshabilitar la recepción de BOPs
- Habilitar o deshabilitar la detección de RAI (Remote Alarm Indication – Indicación de Alarma Remota)
- Configuración del número de BOPs que deben ser recibidos para declarar un BOP válido, consultar referencia [1]
- Inicializar a cero el contador de BOPs
- Enviar y recibir BOPs

5.3 Presentación del diseño

En la figura 12 se presenta el diseño de la partición control y monitoreo del submódulo de BOPs del módulo FDL del sistema de evaluación de circuitos integrados, este diseño sigue la arquitectura propuesta en el capítulo 4. Siguiendo la figura 12, vemos que cuando el usuario oprime un botón de la ventana se ejecuta el siguiente flujo de control: primero se envía un mensaje a la instancia PControlMonitoreo de la capa de presentación, esta instancia construye el mensaje que será enviado en un evento a la instancia CoordinadorControlMonitoreo de la capa coordinador de aplicaciones, ésta se encargará de enviar el mensaje a la instancia DControlMonitoreo de la capa de dominio, la cual validará el mensaje y obtendrá una instancia del manejador de dispositivo de la capa de servicio, a través de la cual ejecutará los mensajes del módulo FDL por medio de la instancia PFDL, esta última usa la capa de protocolo para comunicarse con el hardware.

Como vimos la capa de coordinador de aplicaciones sólo se limita a ser un mediador entre la capa de presentación y la capa de dominio, respondiendo a los eventos provenientes de la capa de presentación y mapeando información proveniente de los objetos de la capa de dominio. La capa de Dominio es la que se encarga de validar los mensajes y obtener los objetos de la capa de servicio para comunicarse con el hardware.

En la figura 13 se expone el diseño de la partición intérprete de comandos[3] del submódulo de BOPs, del módulo FDL del sistema de evaluación de circuitos integrados. Este módulo sigue un flujo de control similar al expuesto para la partición control y monitoreo.

[3] Este debe de ser visto como el diseño parcial del intérprete de comandos del sistema de evaluación de circuitos integrados, porque esta partición debe soportar todos los modelos del circuito integrado a evaluar.

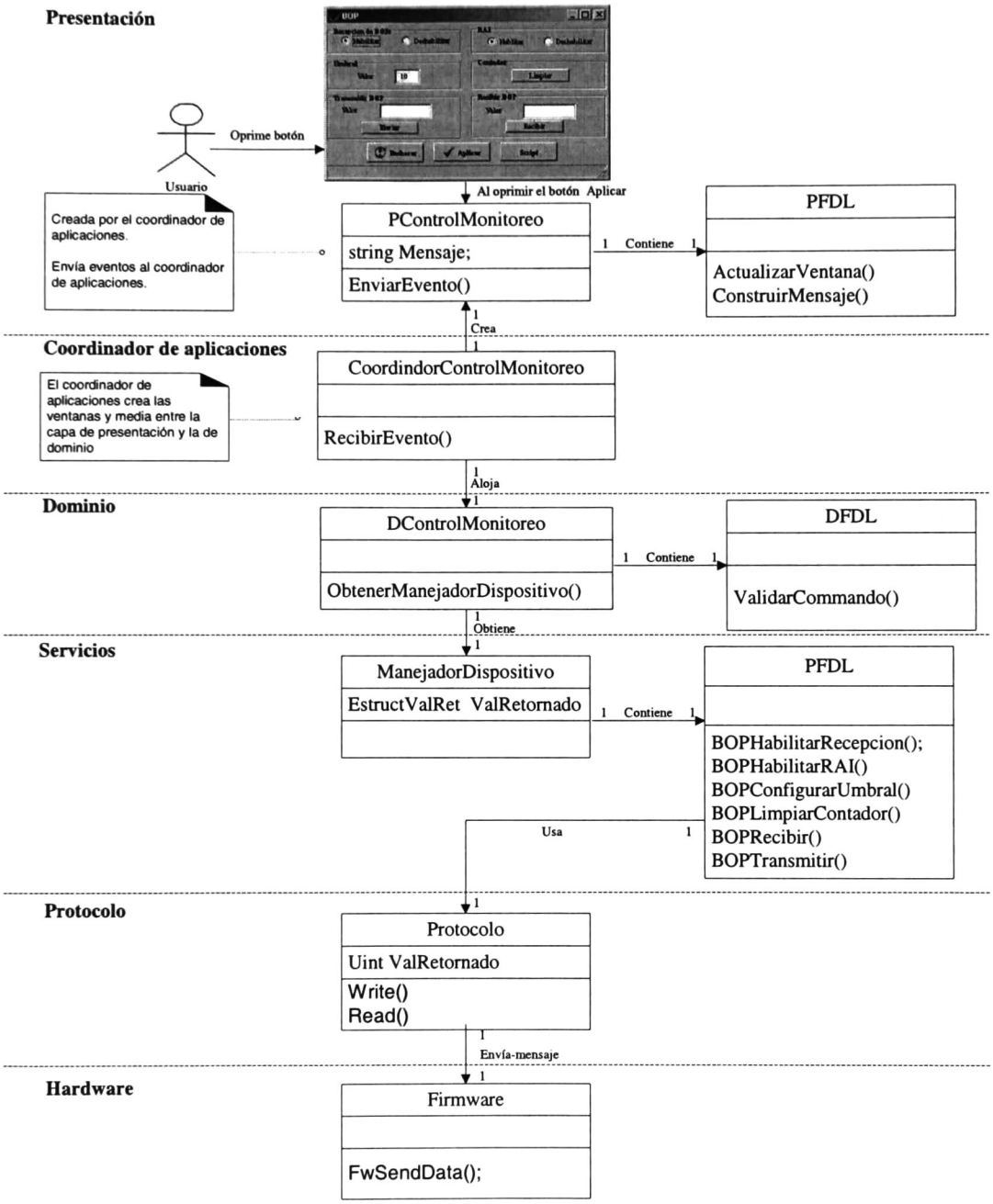


Figura 12. Diseño de la partición control y monitoreo del submódulo de BOPs del módulo FDL del sistema para la evaluación de circuitos integrados.

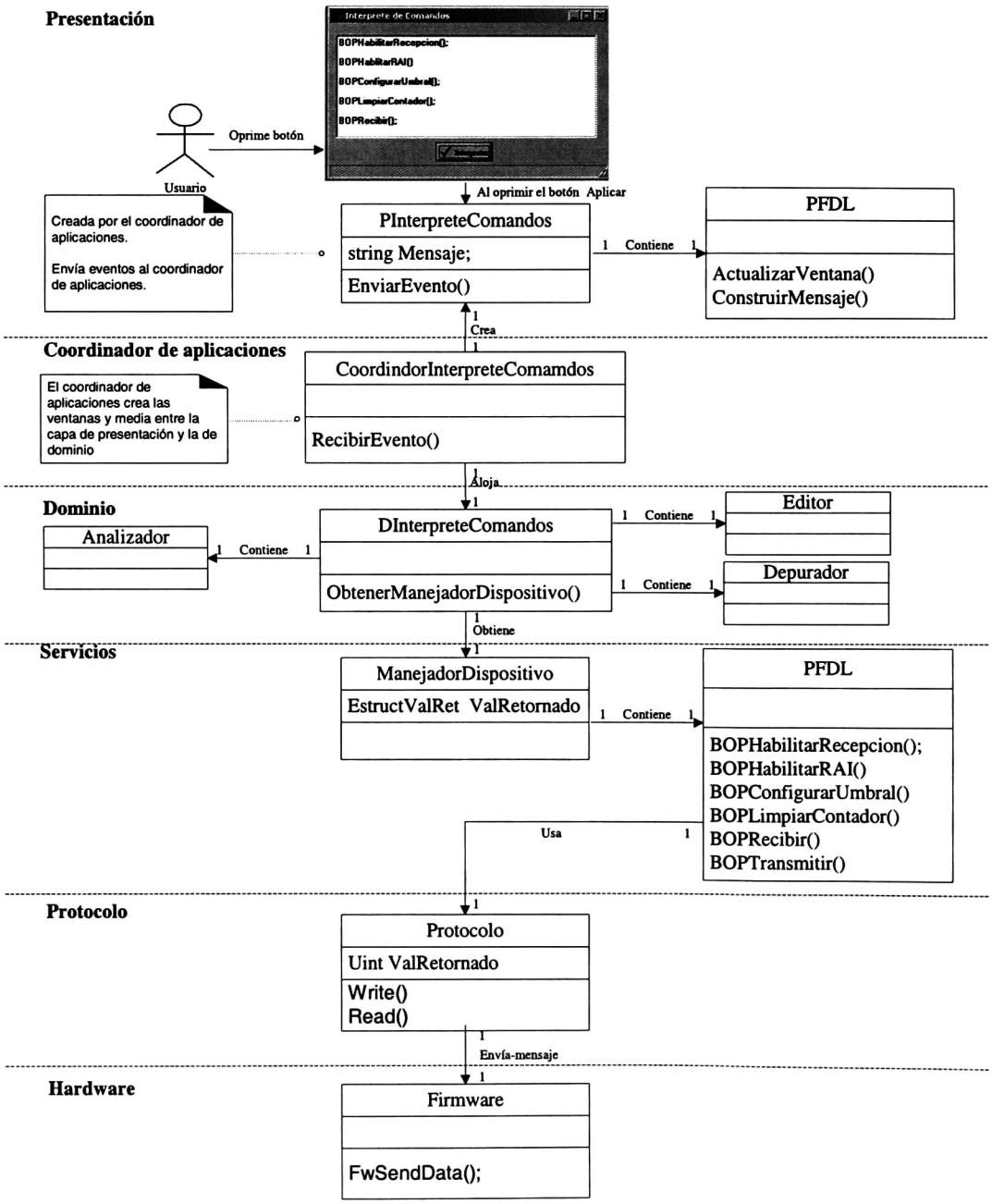


Figura 13. Diseño de la partición intérprete de comandos del submódulo de BOPs del módulo FDL del sistema para la evaluación de circuitos integrados.

En los diseños anteriores vimos que la aplicación de la arquitectura propuesta en el capitulo 4 permite que cada capa pueda ser desarrollada de manera independiente, además, exceptuando por las capas de presentación y coordinador de aplicaciones, permite que el código sea fácilmente portable entre diferentes sistemas de desarrollo de aplicaciones gráficas de usuario, también, permite la reutilización de código en sistemas de evaluación de circuitos integrados con funcionalidad similar.

CAPÍTULO 6

CONCLUSIONES

Esta tesis presentó una arquitectura para el desarrollo de aplicaciones de evaluación de circuitos de alta integración, la cual fue diseñada tomando como base una arquitectura multicapas, donde cada capa tiene un bajo acoplamiento y una alta cohesión[1] con respecto a las otras capas, ésto permitió cumplir con los requerimientos de extensibilidad y portabilidad ofreciendo, además las siguientes ventajas:

- Permite desarrollar independientemente cada una de las capas
- Reduce al mínimo el trabajo producido por los cambios de requerimientos
- Simplifica el mantenimiento
- Soporta una gran capacidad de reutilización y portabilidad de algunas capas ya que cada capa está destinada a un propósito específico

Actualmente el trabajo presentado en esta tesis se explota comercialmente por una compañía, TDCOM S.A. de C.V., que se dedica al diseño de circuitos de alta integración. Como resultado de usar la arquitectura propuesta y del reuso logrado gracias a ella, se pudieron reducir considerablemente los tiempos de desarrollo de un sistema de evaluación.

[1] Una alta cohesión caracteriza a las clases que tienen responsabilidades moderadas en un área funcional y colabora con otras para llevar a cabo las tareas. Una clase con baja cohesión hace muchas funciones no afines o tiene responsabilidades excesivas.

APÉNDICE A

Descripción del proceso para activar los modos de operación del sistema para la evaluación de circuitos integrados

El propósito de este escrito es presentar por medio de un diagrama de flujo el proceso que se sigue para activar los modos de operación que proporciona el sistema para la evaluación de circuitos integrados, éste cuenta con dos modos de operación:

- **Modo en Línea** – Este modo de operación ocurre cuando la aplicación detecta e identifica adecuadamente la tarjeta de evaluación, además se asegura que el firmware esté cargado en el microcontrolador que reside en la tarjeta de evaluación. En este modo de operación se pueden ejecutar las funciones de detección y carga de firmware, monitoreo de estatus, manejo de señales de interrupción, carga y recuperación de registros, ejecución de funciones de configuración y generación de scripts.

- **Modo fuera de línea** – La aplicación entra en modo fuera de línea cuando el hardware no está conectado adecuadamente al CPU (Central Processig Unit – Unidad Central de Procesamiento), o cuando el usuario deliberadamente desea no conectar el hardware con la aplicación, para trabajar en modo fuera de línea. En este modo de operación la aplicación le permite al usuario crear gráficamente configuraciones y guardarlas en scripts de configuración. Estos scripts pueden ser utilizados por el sistema aquí descrito o por algún otro software.

Vemos entonces que el proceso para activar los modos de operación, está directamente relacionado con el estatus que devuelven los procesos para la detección del hardware y carga de firmware.

En la figura A.1 se muestra un diagrama de flujo que ilustra el proceso de detección del hardware, carga del firmware al hardware y la activación de los modos de operación del sistema para la evaluación de circuitos integrados.

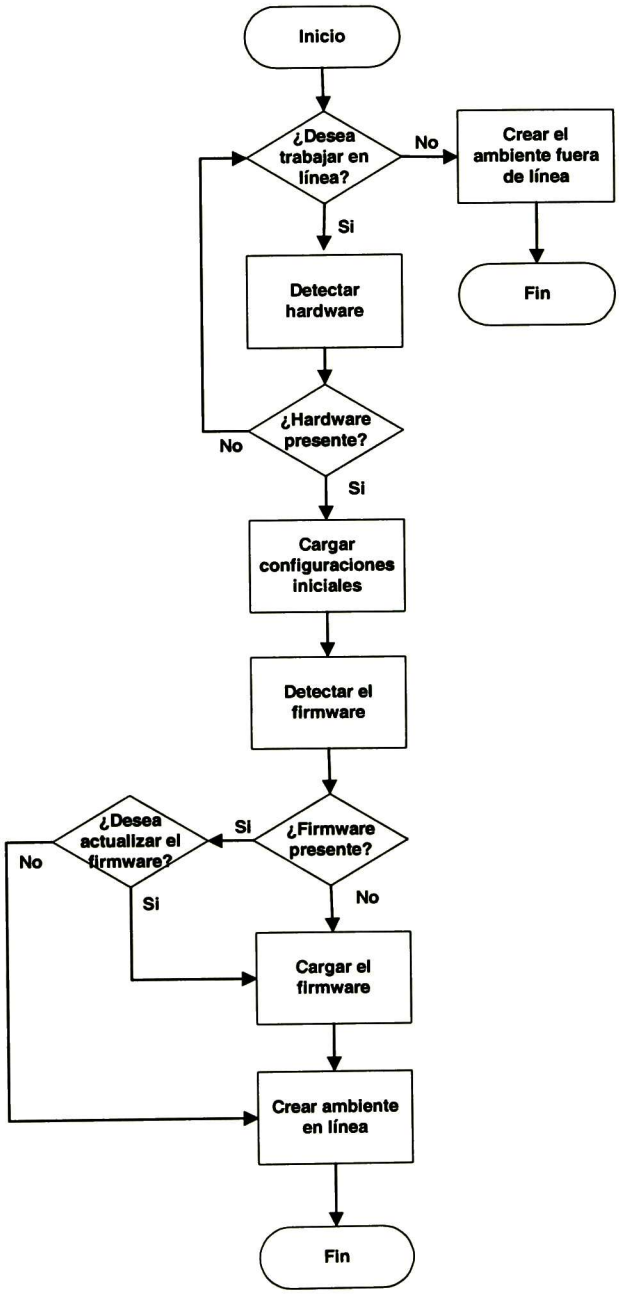


Figura A.1. Diagrama de flujo que presenta el proceso de detección del hardware, carga del firmware al hardware y la activación de los modos de operación del sistema para la evaluación de circuitos integrados.

APÉNDICE B

Requerimientos y análisis para el desarrollo de un software intérprete de código script basado en el lenguaje de programación C

El propósito de este escrito es proporcionar una propuesta para el desarrollo de un intérprete de código script, para ser usado en el sistema para la evaluación de circuitos integrados, este lenguaje script está basado en el lenguaje de programación C.

1 Introducción

Los scripts son una manera muy conveniente de implementar las pruebas de evaluación de un circuito de comunicaciones, pues permite un gran ahorro de esfuerzo como ahora explicaremos. Normalmente los diferentes casos de prueba[1] para la evaluación de un circuito o dispositivo de análisis tienen mucho en común y el uso de los scripts evita que cada vez se tenga que codificar en detalle todo el caso de prueba. Una vez que se hizo la codificación de un caso de prueba de una parte del sistema a evaluar, el código script que realiza ésto se puede reusar en otras pruebas de evaluación que incluyan esta configuración. El fabricante de circuitos integrados le proporciona al cliente una serie de scripts los cuales le permiten al cliente configurar su circuito de una manera más amigable.

En la figura B.1 se muestra un diagrama a bloques de los módulos principales de un software para la evaluación de circuitos integrados que incluye dentro de su estructura un módulo para la interpretación de circuitos integrados y cómo este módulo interactúa con el sistema.

[1] Un caso de prueba es una configuración particular del circuito de comunicaciones.

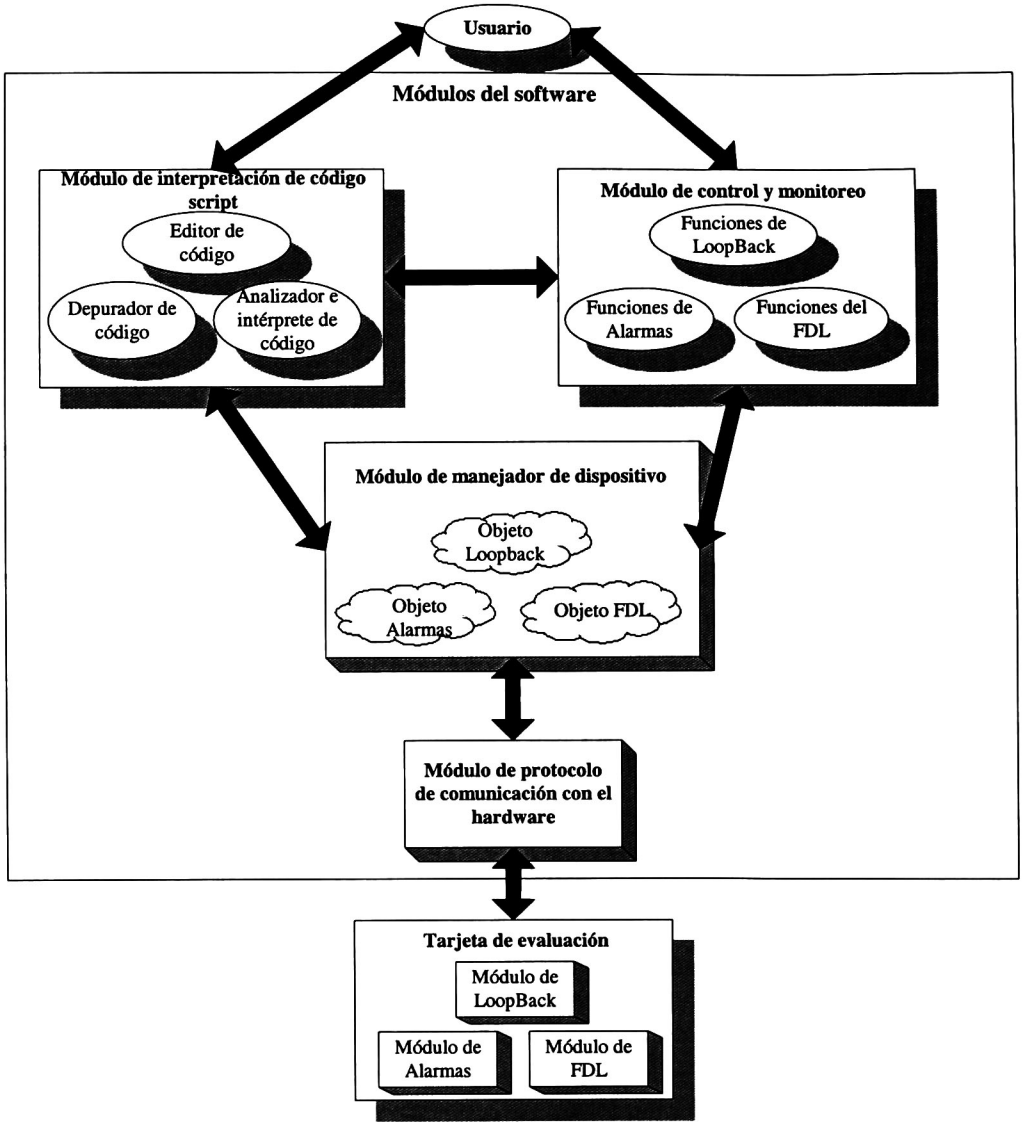


Figura B.1. Módulos de software de un sistema para la evaluación de circuitos integrados.

Como vimos en la figura B.1 el módulo intérprete de código script está compuesto de tres módulos:

- Módulo de análisis e interpretación de código script
- Módulo de depuración de código script
- Módulo de edición de código script

A continuación se expondrán los requerimientos y análisis de cada uno de estos módulos.

2 Requerimientos y análisis para el módulo de análisis e interpretación de código script

Empezaremos presentando la estructura de este lenguaje script, el cual llamaremos lenguaje C script, y después presentaremos los análisis utilizados para su validación y posterior interpretación.

2.1 Estructura del lenguaje C script

La estructura del lenguaje C script está dividida en:

- Palabras reservadas
- Operadores
- Tipos de datos

Este lenguaje no tiene manejo de procedimientos, la declaración de variables sólo puede ser hecha en una sección definida del cuerpo del programa, además estas variables son globales a todo el código y no distingue el uso de mayúsculas y minúsculas, estos requerimientos se explicarán más adelante.

2.1.1 Palabras reservadas

En la tabla B.1. se muestra una lista de las palabras reservadas del lenguaje C script.

Palabra reservada	Tipo
If	Secuencia de control
Else	Secuencia de control
While	Secuencia de control
Loop	Secuencia de control
Break	Secuencia de control
Exit	Secuencia de control
Wait	Secuencia de control
Link	Palabra reservada que liga (incluye) un archivo script
Include	Directiva
Define	Directiva
VarList	Encabezado de definición de variables

Tabla B.1. Lista de palabras reservadas del lenguaje C script.

La palabra reservada **Link** es usada para ligar un archivo script, esta liga funciona como una macro. La secuencia de control **Loop** itera una sección de código script un número definido de veces. La secuencia de control **Wait** detiene la ejecución de código en un tiempo definido en milisegundos.

2.1.2 Operadores

Hay diferentes tipos de operadores, los cuales los hemos agrupado de la siguiente forma:

- Operadores del lenguaje
- Operadores relaciónales
- Operadores lógicos
- Operadores de manejo de bits

2.1.2.1 Operadores del lenguaje

En la tabla B.2. se muestra una lista de los operadores del lenguaje C script.

Operador	Descripción
{	Operador usado en las secuencias de control y en la sección de declaración de variables
}	Operador usado en las secuencias de control y en la sección de declaración de variables
(	Operador usado en las secuencias de control. Expresiones y métodos del manejador de dispositivo
)	Operador usado en las secuencias de control. Expresiones y métodos de un manejador de dispositivo
.	Operador usado en los métodos de un manejador de dispositivo
,	Operador usado para separar parámetros y en la declaración de variables
;	Operador usado para separar proposiciones sintácticas
"	Operador usado en la declaración de directivas y en la definición de una liga de archivo
#	Operador usado en la declaración de directivas
//	Operador usado en la declaración de cometarios
\	Operador usado en la declaración de rutas de archivos

Tabla B.2. Lista de operadores del lenguaje C script.

2.1.2.2 Operadores relacionales

En la tabla B.3. se muestra una lista de los operadores relacionales del lenguaje C script.

Operador	Descripción
>	Mayor que
<	Menor que
>=	Mayor igual
<=	Menor igual
==	Igual
!=	Diferente de

Tabla B.3. Lista de operadores relacionales del lenguaje C script.

2.1.2.2 Operadores lógicos

En la tabla B.4. se muestra una lista de los operadores lógicos del lenguaje C script.

Operador	Descripción
&&	Conjunción
||	Disyunción
!	Negación

Tabla B.4. Lista de operadores lógicos del lenguaje C script.

2.1.2.4 Operadores de manejo de bits

En la tabla B.5. se muestra una lista de los operadores de manejo de bits del lenguaje C script.

Operador	Descripción
&	Operación AND
|	Operación OR
^	Operación XOR
<<	Corrimiento a la derecha
>>	Corrimiento a la izquierda
~	Complemento

Tabla B.5. Lista de operadores de manejo de bits del lenguaje C script.

2.1.3 Tipos de datos

En la tabla B.6. se muestra una lista de los tipos de datos del lenguaje C script.

Tipo	Descripción	Rango
Uint8	8 Bits	0 a 255
Uint16	16 Bits	0 a 65,535
Uint32	32 Bits	0 a 4,294,967,295

Tabla B.6. Lista de tipos de datos del lenguaje C script.

Con esto terminamos de presentar la estructura del lenguaje.

2.2 Proceso para la interpretación del código script

El proceso para la interpretación de un código script involucra los siguientes pasos:

- Análisis léxico
- Análisis sintáctico
- Análisis semántica
- Interpretación del código

Cuando se codifica un script y se procede a interpretarlo, los posibles errores de tipo léxico, sintáctico y semántico que contenga el código serán detectados en el proceso de análisis y se listarán al usuario los errores encontrados.

A continuación se presenta el análisis para la implementación de los analizadores léxico, sintáctico y semántico.

2.2.1 Análisis léxico

El análisis léxico agrupa los caracteres para identificar símbolos y operadores que son unidades léxicas, tales como palabras reservadas, identificadores y operadores. Este proceso usa una máquina de estados finitos, el desarrollo de esta máquina de estados finitos lo presentaremos más adelante. El análisis léxico ignorara los espacios en blanco, tabuladores y caracteres de fin de línea, así como todos los caracteres después del operador // hasta que encuentre un fin de línea.

A continiación presentamos los diagramas de transición para el análisis léxico del lenguaje C script.

2.2.1.1 Diagramas de transición para el análisis léxico del lenguaje C script

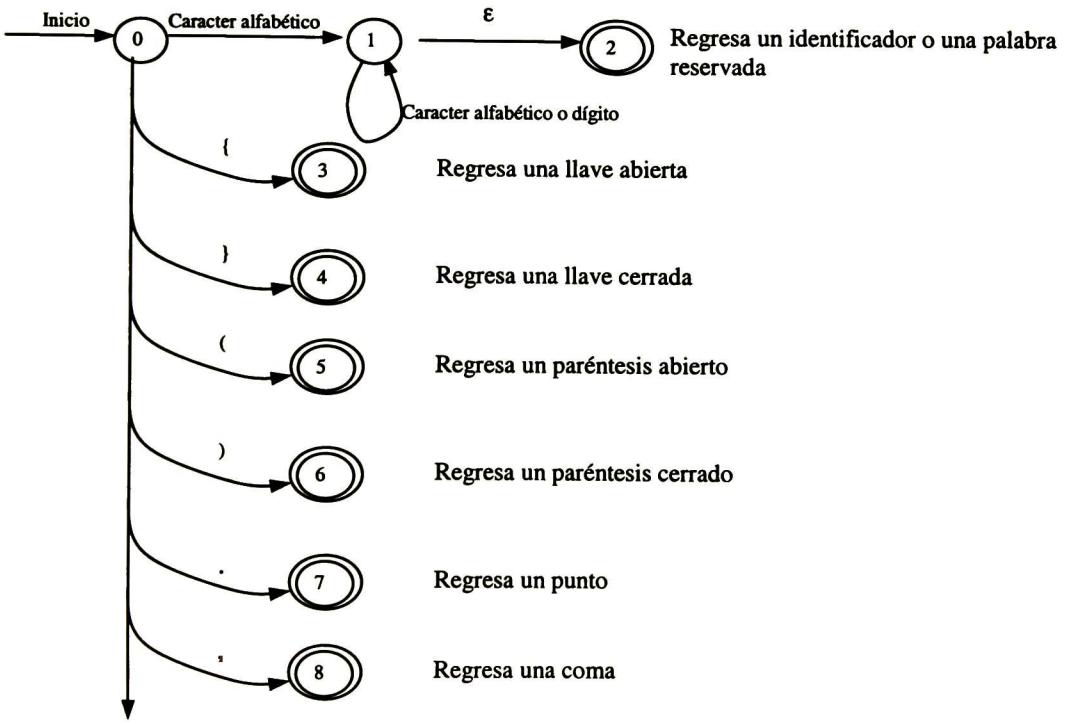
Los diagramas de transición son una máquina de estados finitos que acepta unidades léxicas como son:

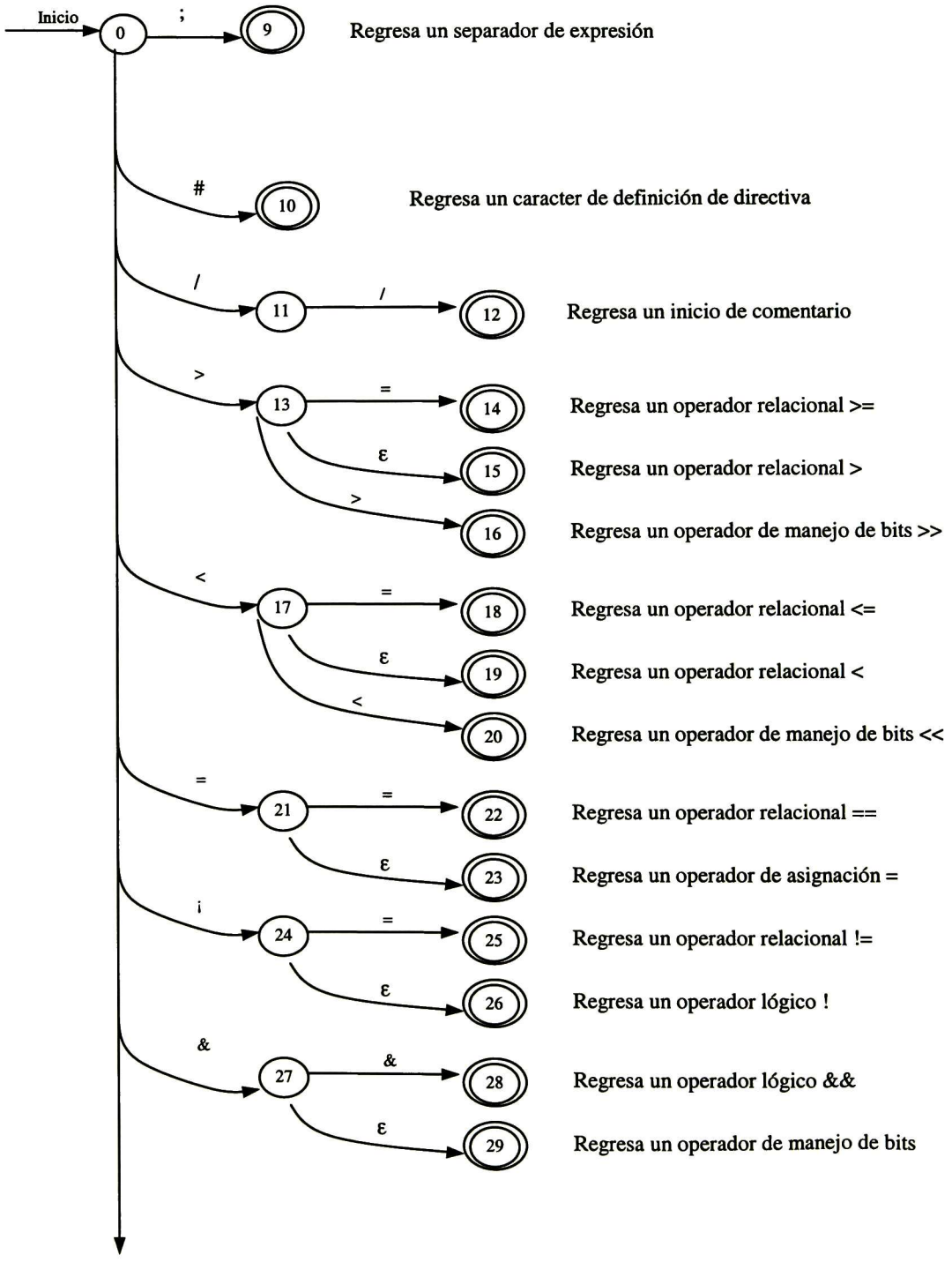
- Palabras reservadas
- Operadores
- Identificadores
- Constantes
- Cadenas de caracteres
- Signos de puntuación

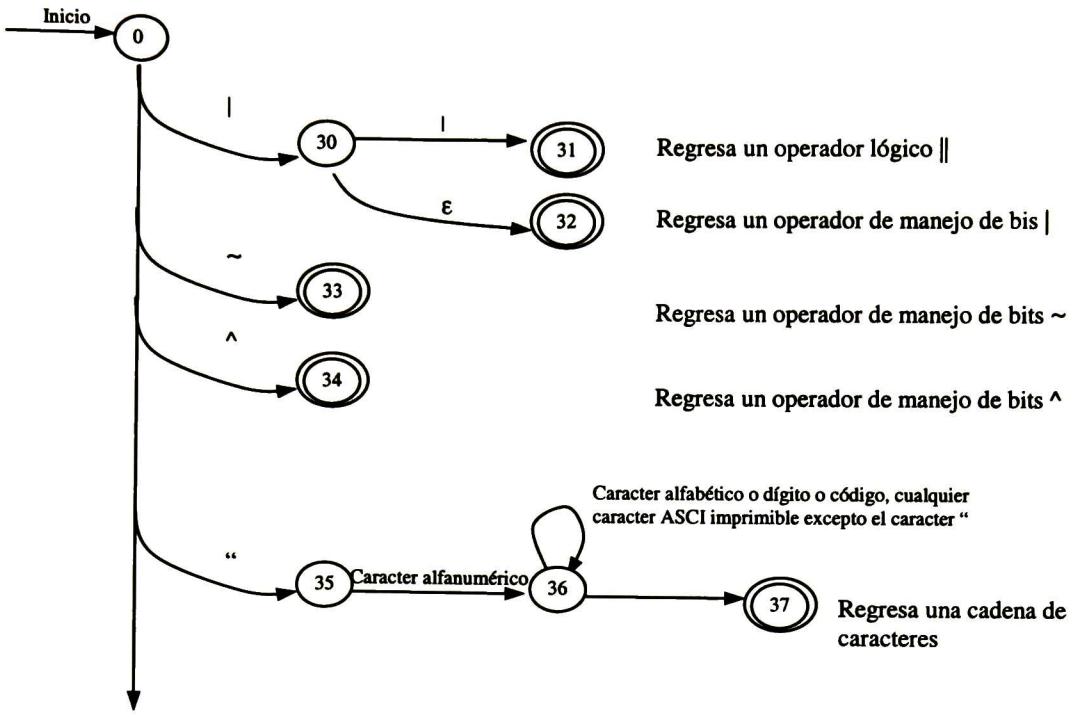
El diagrama de transición propuesto para ser usado en el análisis léxico es el siguiente:

Dígito ⟶ 0 | 1| 2| 3| 4| 5| 6| 7| 8| 9

Caracter alfabético ⟶ [(a .. z) | (A ..Z)]*




Inicio
0
;
9
Regresa un separador de expresión
#
10
Regresa un caracter de definición de directiva
/
11
/
12
Regresa un inicio de comentario
>
13
=
14
Regresa un operador relacional >=
ε
15
Regresa un operador relacional >
>
16
Regresa un operador de manejo de bits >>
<
17
=
18
Regresa un operador relacional <=
ε
19
Regresa un operador relacional <
<
20
Regresa un operador de manejo de bits <<
=
21
=
22
Regresa un operador relacional ==
ε
23
Regresa un operador de asignación =
¡
24
=
25
Regresa un operador relacional !=
ε
26
Regresa un operador lógico !
&
27
&
28
Regresa un operador lógico &&
ε
29
Regresa un operador de manejo de bits



Con esto terminamos el análisis de la máquina de estados finitos, para más sobre este tema consulte la referencia [12].

2.2.2 Análisis sintáctico

El análisis sintáctico[2] analiza el flujo de símbolos y se asegura que la secuencia de esos símbolos esté correcta, por ejemplo:

if (expresión) proposición // Sintácticamente correcto
if (expresión proposición // Falta paréntesis, sintacticámente incorrecto

La producción de reglas generadas con ayuda de diagramas de sintaxis, son usados para generar un árbol de análisis sintáctico implementado como autómata.

El árbol de análisis sintáctico describe la estructura sintáctica de la entrada. En la figura B.2. se muestra un árbol de análisis sintáctico para la expresión: *posición = inicial + velocidad * 60*

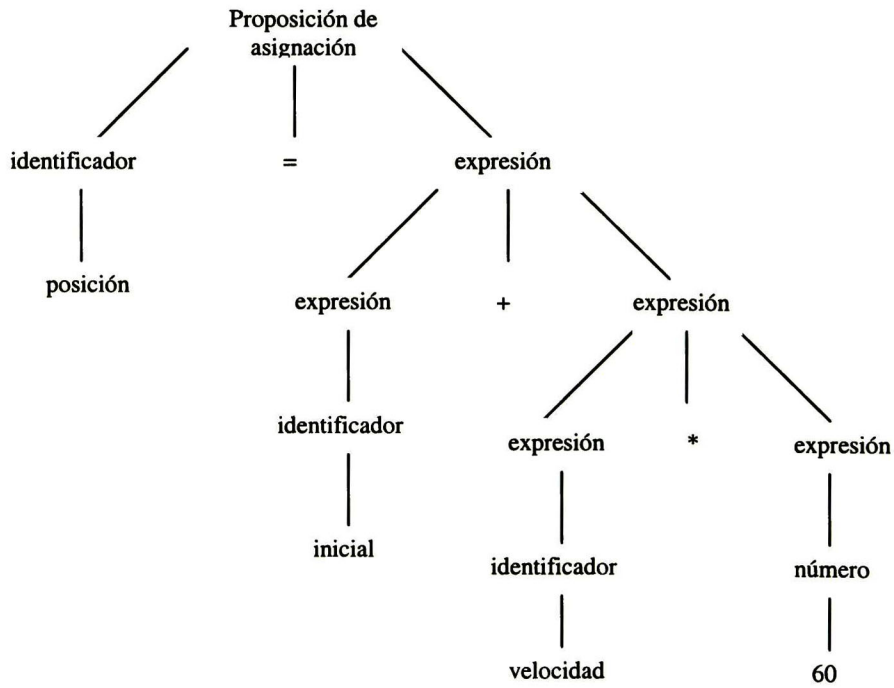


Figura B.2. Árbol de análisis sintáctico para *posición = inicial + velocidad * 60.*

[2] La división entre análisis léxico y sintáctico es algo arbitraria. Generalmente se elige una división que simplifique la tarea completa del análisis. Un factor para determinar esta división es si una construcción del lenguaje fuente es recursiva o no. Las construcciones léxicas no requieren recursión, mientras que las sintácticas suelen requerirlas.

Una representación más común de esta estructura sintáctica es la que da el árbol sintáctico, el cual es una representación compacta del árbol de análisis sintáctico como se muestra en la figura B.3.

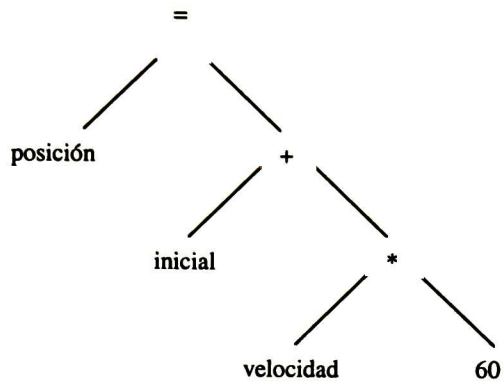
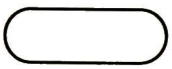


Figura B.3. Árbol sintáctico para *posición = inicial + velocidad * 60.*

A continuación presentamos los diagramas de sintaxis para el análisis sintáctico y las reglas de producción del lenguaje C script.

2.2.2.1 Diagramas de sintaxis para el análisis sintáctico del lenguaje C script


Representa una palabra reservada

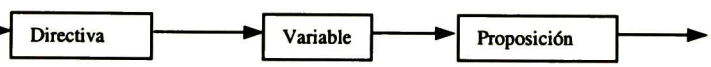

Representa un operador

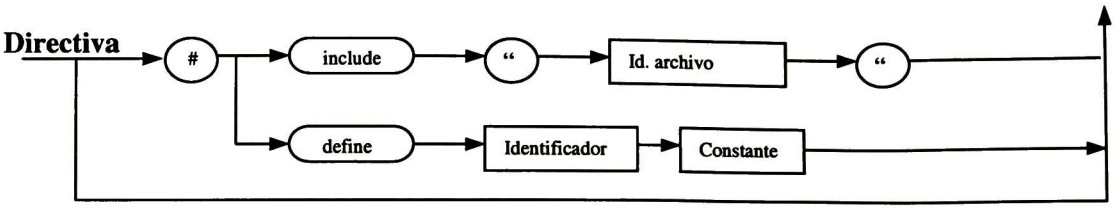

Representa una unidad sintáctica

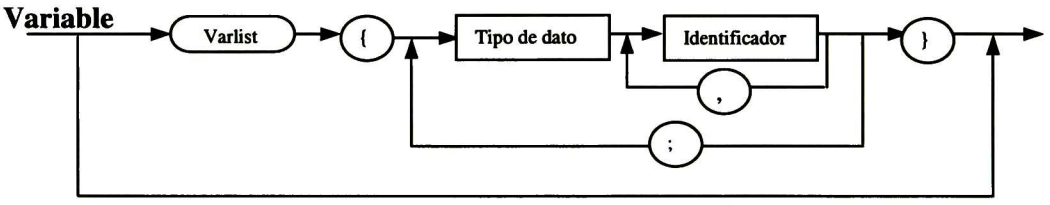
Dígito ⟶ 0 | 1| 2| 3| 4| 5| 6| 7| 8| 9

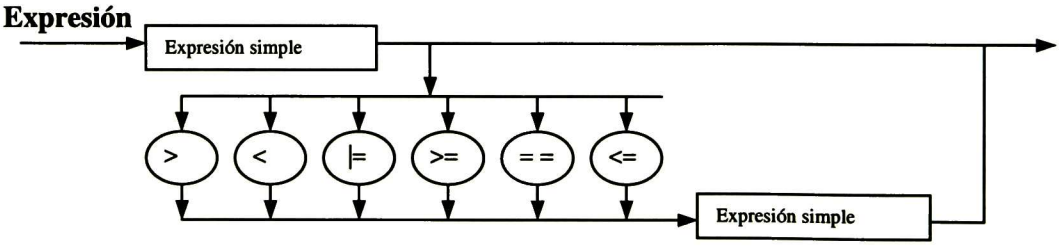
Caracter alfabético ⟶ [(a ... z) | (A … .Z)]*

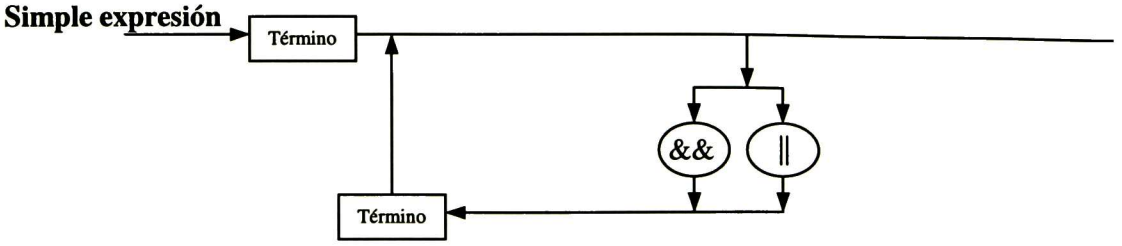
Script

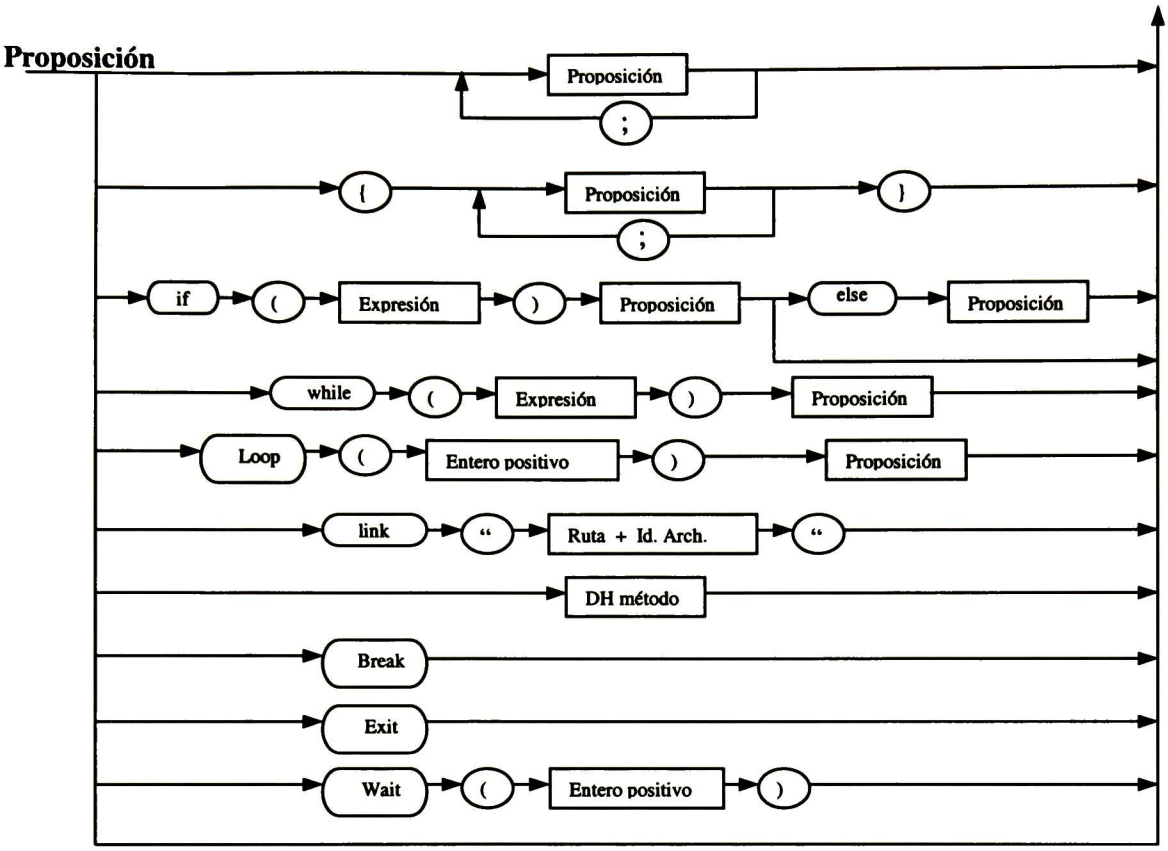


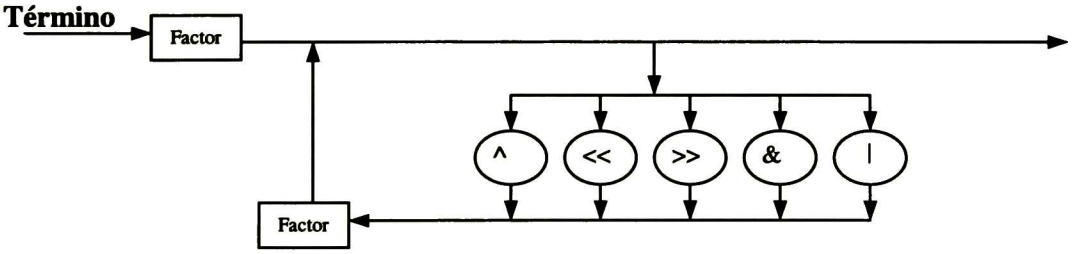

Directiva
#
include
“
Id. archivo
“
define
Identificador
Constante

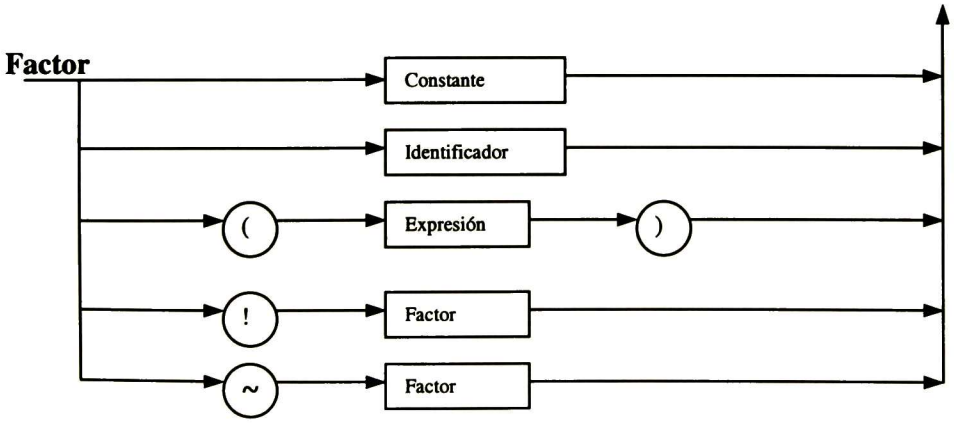

Variable
Varlist
{
Tipo de dato
Identificador
}
,
;


Expresión
Expresión simple
>
<
|=
>=
= =
<=
Expresión simple

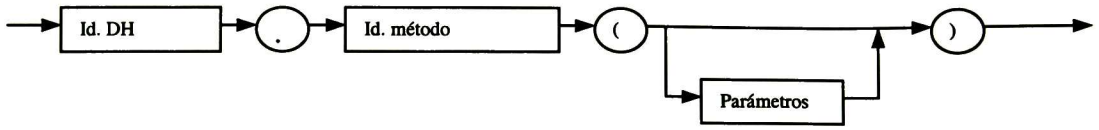

Simple expresión
Término
&&
||
Término


Proposición
Proposición
;
{
Proposición
}
;
if
(
Expresión
)
Proposición
else
Proposición
while
(
Expresión
)
Proposición
Loop
(
Entero positivo
)
Proposición
link
“
Ruta + Id. Arch.
“
DH método
Break
Exit
Wait
(
Entero positivo
)

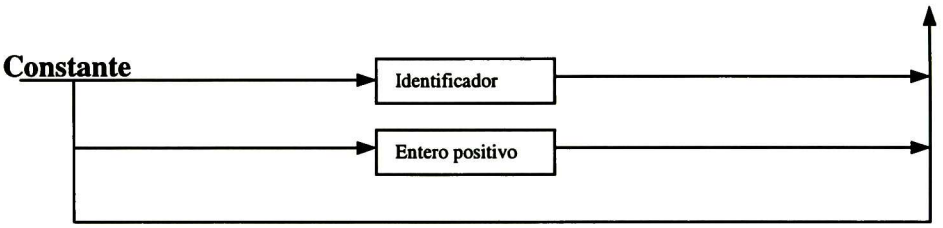

Término
Factor
^
<<
>>
&
|
Factor


Factor
Constante
Identificador
(
Expresión
)
!
Factor
~
Factor

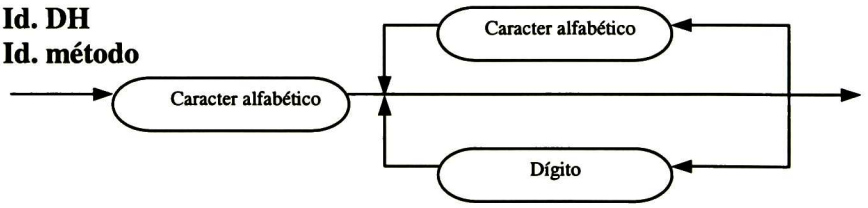
DH método



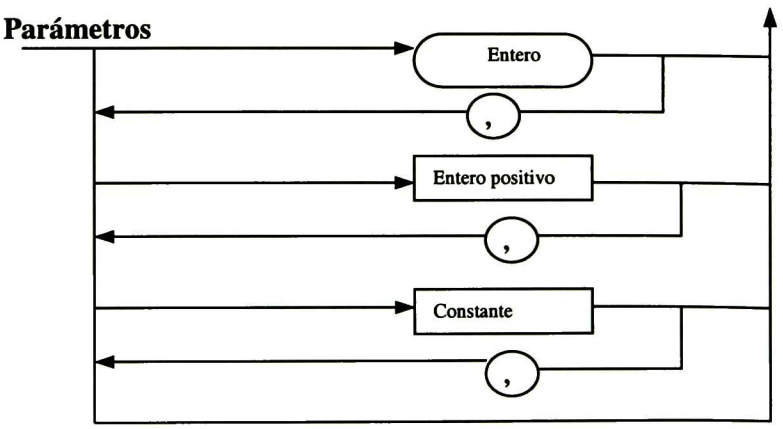
Constante



Id. DH
Id. método



Parámetros



2.2.2.2 Reglas de producción para el lenguaje C script

Las reglas de producción definen una gramática libre de contexto, donde sus componentes léxicos son:

- Símbolos terminales (palabras reservadas y operadores)
- Símbolos no terminales (unidades sintácticas)
- Un conjunto de reglas de producción:
  [No terminal] ⟶ [Terminal o no terminal] [Terminal o no terminal]
- Un símbolo inicial

A continuación damos unos ejemplos para obtener las reglas de producción a partir de los diagramas de sintaxis:

Script ⟶ Directiva | Variable | Proposición

Directiva ⟶ # *include* "Id. archivo"
| # *define* identificador Contstante

Proposición ⟶ Proposición ;
| { Proposición ; }
| if (Expresión) Proposición *else* Proposición
| if (Expression) Proposition
| while (Expresion) Proposición
| Loop (Entero positivo) Proposición
| Link "Ruta + Id. archivo";
| DH método
| Break
| Exit;
| Wait (Entero positivo)
| ϕ

2.2.3 Análisis semántico

La fase de análisis semántico revisa el programa fuente para tratar de encontrar errores semánticos y reúne la información sobre los tipos para la fase posterior de generación de código. En ella se utiliza la estructura jerárquica determinada por la fase de análisis sintáctico para identificar los operadores y operandos de expresiones y proposiciones.

Un componente importante del análisis semántico es la verificación de tipos[3], donde el compilador verifica si cada operador tiene operandos permitidos por la epecificación del lenguaje fuente; por ejemplo:

```
int Total;
char Caracter;
Caracter = 'A';
Total = 5;          // Semánticamente correcto
Total = Caracter;  // Tipo incompatible, semánticamente incorrecto
```

2.2.4 Intérprete de comandos

Después de que un código script pasó por los análisis ya mencionados y está libre de errores, se procede a interpretar el código, esta interpretación lo que hace es mandar una configuración al hardware, en el orden y secuencia programado en el script.

3 Requerimientos del módulo para la depuración de código script

Este módulo de software permitirá la depuración del código script, sus principales funciones son:

- **Ejecutar** – Esta función activa la interpretación del código script.
- **Parar** – Esta función detiene la interpretación del código script.
- **Salir** – Esta función permite salir del módulo de depuración de código script.
- **Insertar/remover marcas de corte** – Esta función permite marcar/desmarcar una línea determinada en el código, ésto es con el fin de que la interpretación del código script se detenga en una línea marcada.

[3] La verificación o comprobación de tipos informa de un error si se aplica un operador o un operando incompatible.

- **Paso simple** – Esta función interpreta una línea de código script y se detiene en la siguiente.

- **Ventana de visualización de variables** – Esta función despliega una ventana de diálogo donde se pueden agregar las variables definidas en un código script, para que muestren al usuario el valor que contienen, este valor puede ser formateado a hexadecimal, binario, octal y decimal.

4 Requerimientos del módulo de edición de código script

Este módulo debe tener todas las funciones básicas de un editor de texto común, como mínimo requerimiento enumeraremos el siguiente menú de funciones que el editor debe presentar al cliente:

- **Funciones de manejo de archivo**
  a) Crear archivo
  b) Guardar archivo
  c) Guardar archivo con un nombre diferente
  d) Abrir archivo
  e) Cerrar archivo
  f) Salir del sistema

- **Funciones de edición de código**
  a) Seleccionar texto
  b) Copiar texto
  c) Pegar texto
  d) Cortar texto
  e) Borrar texto
  f) Buscar texto
  g) Ir a línea

Desde este editor se debe poder llamar a las funciones del módulo de depuración de código script, entonces estas funciones deben mostrarse en el menú de opciones del editor. Por ejemplo:

- **Funciones de depuración de código**
  a) Ejecutar
  b) Parar
  c) Salir
  d) Insertar/remover marcas de corte
  e) Paso simple
  f) Ventana de visualización de variables

No daremos más detalles sobre el análisis y diseño de este editor ya que en el mercado se pueden encontrar editores bastante poderosos que permiten la integración de compiladores e intérpretes.

APÉNDICE C

Descripción general de la arquitectura para el desarrollo de manejadores de dispositivos de red en Windows 2000/NT

El propósito de este escrito es dar una descripción general de la arquitectura para el desarrollo de manejadores de dispositivo de red en Windows 2000/NT (Windows, de aquí en adelante).

Nota:

Los conceptos referentes a la arquitectura del sistema operativo Windows no son tratados aquí, consulte el apéndice D para tener una visión general de la arquitectura del sistema operativo Windows y revise la referencia [3].

En la figura C.1 se muestra un diagrama a bloques de la arquitectura para el desarrollo de manejadores de dispositivo de red en Windows.

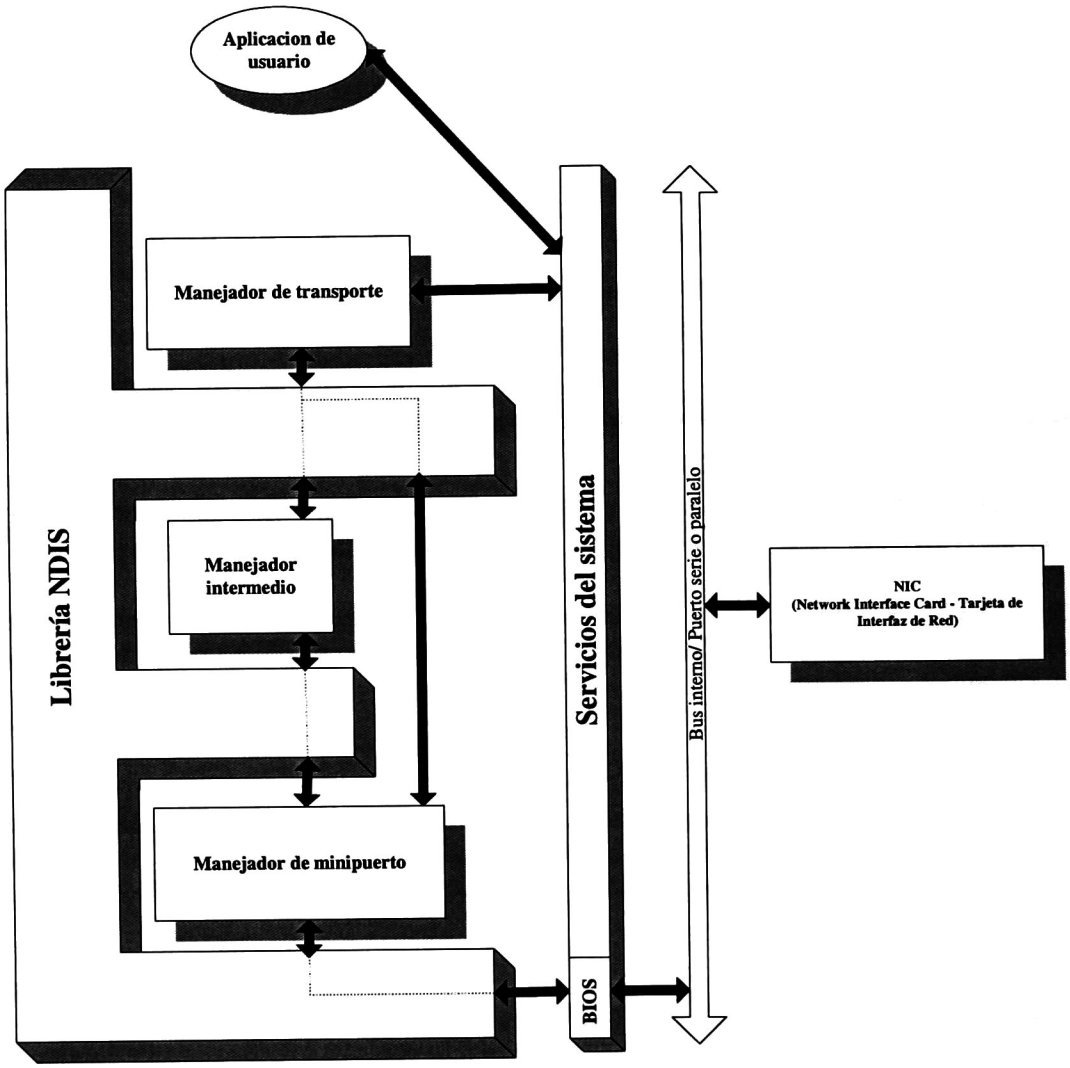


Figura C.1. Arquitectura para el desarrollo de manejadores de dispositivo de red en Windows.

A continuación se dará una descripción general de la librería NDIS y de los manejadores que soporta. Los manejadores de transporte, intermedio y minipuerto se les llama genéricamente *manejadores de red.*

1 Librería NDIS

La librería NDIS usa una abstracción del hardware de red hecha por los manejadores de red. El NDIS también especifica una interfaz estándar entre las capas de los manejadores de red, abstrayendo los manejadores de hardware de bajo nivel hacia capas superiores. NDIS también mantiene el estado de la información y los parámetros de los manejadores de red, incluyendo punteros a funciones y otros valores del sistema.

La librería NDIS soporta los siguientes tipos de manejadores:

- Manejadores de minipuerto
- Manejadores intermedios
- Manejadores de transporte

A continuación daremos una breve explicación de estos tipos de manejadores.

1.1 Manejadores de minipuerto

Un manejador de minipuerto tiene dos funciones principales:

- Manejar el envío y recepción de datos a través del NIC (Network Interface Card – Tarjeta de Interfaz de Red)
- Conectarse con manejadores de una capa superior, tales como manejadores intermedios y manejadores de transporte

Un manejador de minipuerto comunica al NIC con los niveles superiores a través de la librería NDIS. El NDIS exporta un gran conjunto de funciones (función Ndisxxx), éstas encapsulan todas las funciones del sistema operativo que un minipuerto necesita llamar. El manejador de minipuerto por su parte tiene que exportar un conjunto de funciones de entrada (función minipuertoOpen, minipuertoRx, minipuertoXxx) que el NDIS llama para accesar al manejador de minipuerto.

1.2 Manejadores intermedios

Los manejadores intermedios son usados de las siguientes formas:

- Para trasladar los paquetes de información entre diferentes medios de red. Por ejemplo, la función entre un manejador intermedio Ethernet y un minipuerto ATM, es fragmentar y defragmentar los paquetes Ethernet a paquetes ATM y viceversa
- Para filtrar paquetes

1.3 Manejadores de transporte

Un manejador de transporte, es el manejador de alta jerarquía en el NDIS. Un manejador de transporte asigna paquetes, copia los datos que la aplicación le envió dentro de paquetes y manda esos paquetes al manejador de bajo nivel por medio de llamadas a funciones del NDIS. Un manejador de transporte también proporciona una interfaz para recibir los paquetes entrantes de un manejador de nivel bajo y transfiere los datos recibidos a la aplicación.

Hemos concluido la descripción general de la arquitectura para el desarrollo de manejadores de dispositivo de red en Windows. Para conocer acerca de la arquitectura para el desarrollo de manejadores de dispositivos en otro sistema operativo consulte las referencias [3] y [16].

APÉNDICE D

Descripción general de la arquitectura Windows 2000/NT

El propósito de este escrito es dar una breve descripción de la arquitectura de Windows 2000/NT. Para una descripción más amplia consulte la referencia [6.]

1 Diseño de Windows 2000/NT

Windows 2000/NT (Windows, de aquí en adelante) es el resultado de una compleja compilación de metas idealizadas y requerimientos reales de mercado, los diseñadores de Windows centraron su atención en los siguientes requerimientos:

- **Compatibilidad** – El sistema operativo debe soportar un gran número de software y hardware existente.

- **Robustez y fiabilidad** – El sistema operativo tiene que resistir el ataque de usuarios maliciosos, las aplicaciones deben estar aisladas una de otra como sea posible.

- **Portabilidad** – El sistema operativo debe ser capaz de correr en las plataformas existentes y en las futuras.

- **Extensibilidad** – Debe ser posible adicionar nuevos módulos y soportar nuevos manejadores de Entrada/Salida (E/S, de aquí en adelante) sin modificar el código base existente.

- **Desempeño** – El sistema operativo debe ser capaz de funcionar en plataformas con multiprocesadores.

Todos estos requerimientos fueron complejamente balanceados, cuidando que el tiempo de desarrollo del sistema operativo fuera razonable, obteniendo así una arquitectura del sistema operativo. En la figura D.1 se muestra un diagrama a bloques de la arquitectura del sistema operativo Windows que los diseñadores dieron para satisfacer los requerimientos antes mencionados.

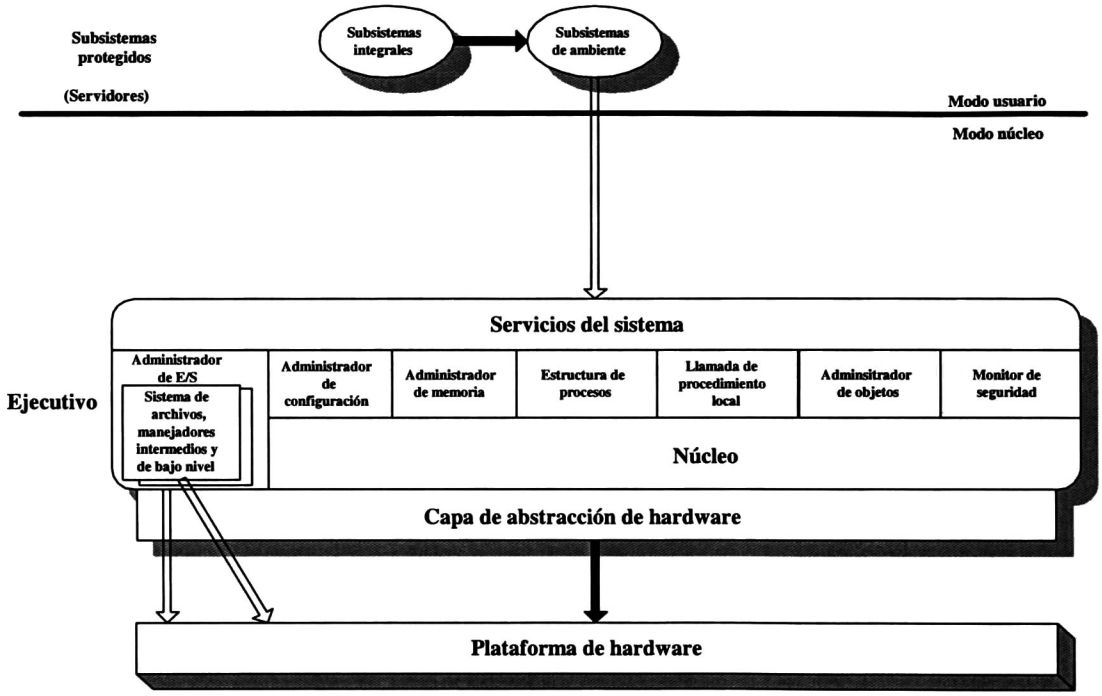


Figura D.1. Arquitectura de Windows.

Consulte las referencias [3], [8] y [18] para conocer un poco más sobre el diseño de sistemas operativos y entender mejor las secciones siguientes.

Antes de explicar las funciones de los componentes de la arquitectura se explicacrán los modos de operación.

2 Privilegios de hardware en Windows

Hay cosas que no se deben permitir hacer a los programas de aplicación en un ambiente multitarea, por ejemplo, la manipulación del manejo de memoria de hardware o parar el procesador, son dos ejemplos de acciones que pueden causar serios problemas. Dependiendo de los tipos de aplicaciones, Windows hace uso de mecanismos de revisión de privilegios de hardware que garantizan la integridad del sistema.

Para evitar las dependencias de hardware, Windows usa un modelo simplificado para describir los privilegios de hardware. Este modelo, entonces, instancia cualquier mecanismo de revisión de privilegios que esté disponible en un CPU dado. Un CPU debe ser capaz de operar en estos dos modos:

- **Modo núcleo** – En este modo, una tarea puede ejecutar una instrucción privilegiada y ésta tiene un acceso completo a algún manejador de E/S. Este modo corresponde al Ring 0 en un procesador Intel 80x86.

- **Modo usuario** – En este modo, el hardware previene la ejecución de instrucciones privilegiadas y revisa los accesos a referencias de memoria y espacio de E/S. En un procesador Intel 80x86 este modo corresponde al Ring 3.

A continuación se dará una explicación general de los componentes que residen en el *modo núcleo* del sistema operativo.

2.1 Componentes base del sistema operativo

Los componentes base de Windows implementan la plataforma general del sistema operativo dentro de la cual se construye todo un ambiente complejo. En la figura D.2 se muestra un diagrama a bloques de los componentes base que son: capa de abstracción de hardware, núcleo y ejecutivo; éstos están dentro del código del modo núcleo.

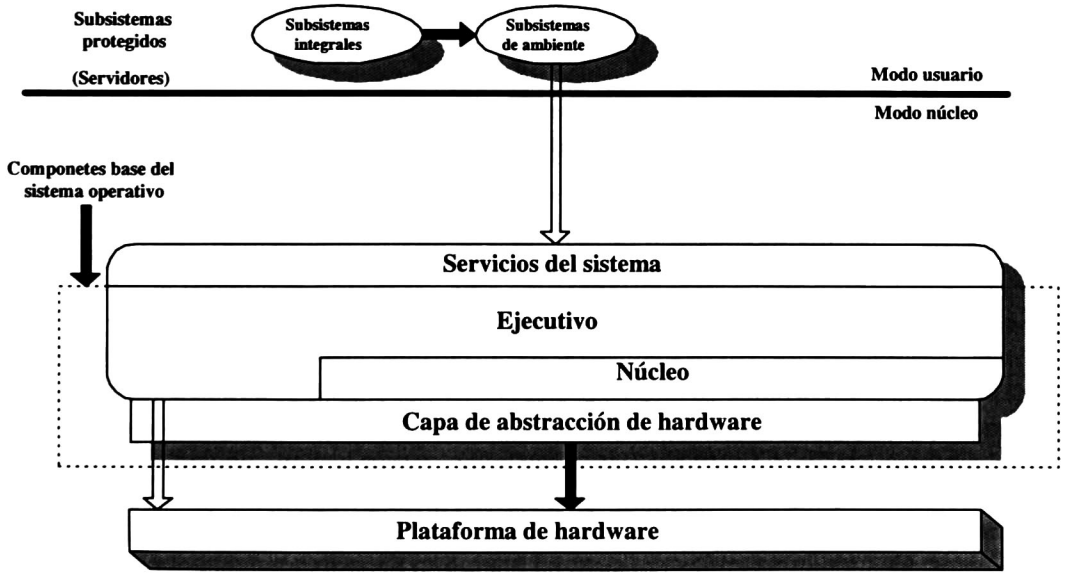


Figura D.2. Arquitectura de los componentes base del modo núcleo.

A continuación se dará una descripción de cada uno de estos bloques.

2.1.1 Capa de abstracción de hardware

La capa de abstracción de hardware (HAL, de aquí en adelante, por sus siglas en inglés, Hardware Abstraction Layer) abstrae el hardware del sistema que no incluye al CPU. La HAL exporta un conjunto de funciones bien definidas tales como:

- Relojes
- Buses de E/S
- Manejadores de registros
- Controladores de interrupciones
- Controladores DMA (Dierect Memory Access – Acceso Directo a Memoria)

Varios componentes del sistema usan las funciones del HAL para interactuar con el hardware, que no incluye al CPU. Su propósito es ocultar los detalles específicos de la plataforma y quitar la necesidad de tener diferentes versiones de sistemas operativos para diferentes plataformas de sistema. El uso de las funciones del HAL hace al núcleo y a los manejadores de dispositivos binarios, compatibles a través de plataformas con la misma arquitectura de CPU.

2.1.2 Núcleo

El núcleo presenta una vista del CPU. Entre otras cosas, el núcleo proporciona mecanismos para:

- Despacho de interrupciones y excepciones
- Sincronización y planificación de hilos
- Sincronización de multiprocesadores
- Mantenimiento de reloj

Cuando las capas superiores del sistema operativo usan los servicios del núcleo, éstas ignoran la arquitectura del CPU. Esto hace posible que el código fuente de los manejadores y componentes de alto nivel del sistema operativo sean portables a través de diferentes arquitecturas de CPU, esto quiere decir que, para portar Windows a una nueva arquitectura de CPU, sólo se tiene que reescribir el núcleo. Para ésto, los diseñadores de Windows diseñaron un micronúcleo que fuera lo más pequeño posible. Una característica importante del núcleo es que ofrece una interfaz basada en objetos para sus clientes. Cuando otras partes del sistema operativo necesitan ayuda del núcleo, hacen uso de los servicios del núcleo por medio de llamadas a funciones que crean y manipulan varios tipos de objetos. Estos objetos se clasifican en dos categorías:

- **Objetos despachadores** – Éstos son usados principalmente para el manejo y sincronización de hilos.

- **Objetos de control** – Estos objetos afectan el comportamiento del sistema operativo.

2.1.3 Ejecutivo

El ejecutivo es por mucho el componente más grande y complejo en Windows. Su trabajo es implementar las funciones básicas normalmente asociadas con un sistema operativo. Tenemos que el ejecutivo y el núcleo usan el HAL para interactuar con algún hardware que no pertenece al CPU, pero el ejecutivo tiene una ventaja sobre el núcleo, que es la de ser portable a través de cualquier arquitectura de CPU.

En la figura D.3 se muestra un diagrama a bloques de la arquitectura de los componentes del ejecutivo. Los componentes del ejecutivo son completamente independientes, éstos se comunican entre ellos a través de interfaces bien definidas. Esta modularidad permite remplazar cualquier componente del ejecutivo sin afectar a otras partes del sistema operativo.

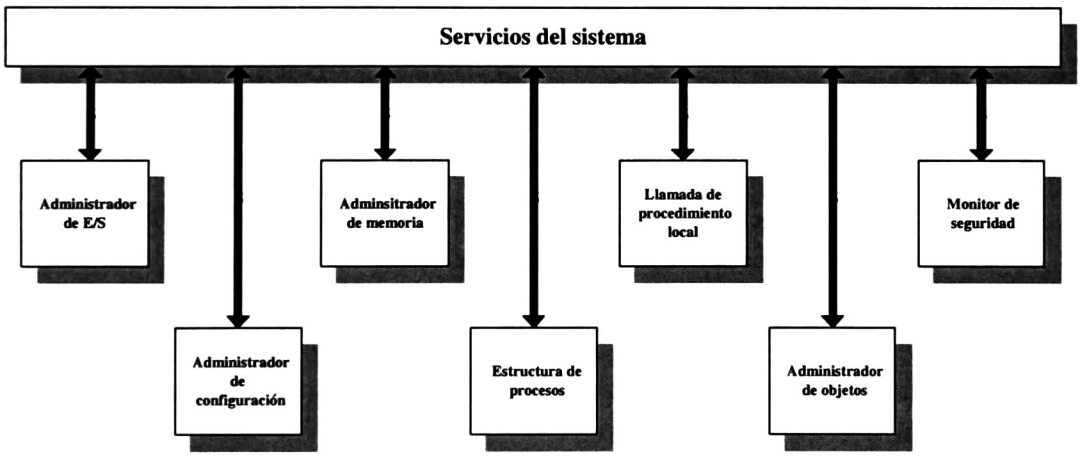


Figura D.3. Arquitectura de los componentes del modo ejecutivo.

A continuación se dará una descripción general de los módulos del ejecutivo. Véase figura D.1. y D.3.

2.1.3.1 Servicios del sistema

Todos los sistemas operativos deben limitar a los procesos que corren en modo usuario la posibilidad de ejecutar código en modo núcleo. En Windows, el despachador de servicios del sistema usa una técnica basada en excepciones del hardware del CPU para permitir el acceso de los servicios del ejecutivo al código que corre en modo usuario.

2.1.3.2 Administrador de objetos

El ejecutivo ofrece este servicio a los procesos del modo usuario a través de una interfaz basada en objetos. Esos objetos ejecutivos representan archivos, procesos, hilos y segmentos de memoria. El uso de estos objetos proporciona un mecanismo para el seguimiento de recursos y para reforzar la seguridad.

2.1.3.3 Administrador de configuración

Desde el punto de vista de la escritura de un manejador de dispositivo, el principal trabajo de administración de configuración es el de mantener un modelo de todo el hardware y software instalado en la máquina. Esto se hace usando una base de datos llamada Registro (Registry, de aquí en adelante). Los manejadores están ligados al Registry a través de conexiones.

Entre otras cosas, los manejadores usan el Registry para:

- Identificarse como verdaderos componentes del sistema
- Encontrar y localizar periféricos de hardware
- Habilitar medidas de desempeño del manejador

2.13.4. Administrador de procesos

El administrador de procesos es el componente ejecutivo que maneja la creación, manejo y destrucción de procesos e hilos. Éste también proporciona un conjunto de servicios estándares para la sincronización de las actividades de los hilos.

2.1.3.5 Monitor de seguridad

Este componente ejecutivo ofrece al sistema políticas de seguridad. El monitor de seguridad no define la política de seguridad, este trabajo corresponde al subsistema de seguridad local. El monitor de seguridad simplemente proporciona un conjunto de primitivas que pueden llamar los componentes del modo
núcleo y modo usuario para validar accesos a los objetos y hace la revisión de privilegios de usuario.

Los manejadores de dispositivo normalmente no trabajan con el monitor de seguridad, los manejadores de dispositivo no se preocupan por sus problemas de seguridad, el administrador de E/S se encarga de manejar esos detalles.

2.1.3.6 Manejador de memoria

Dentro de Windows cada proceso tiene un máximo de cuatro gigabytes de espacio de direcciones virtuales. La primera mitad de este espacio de direcciones contiene código de procesos privados y datos. Éste también mantiene archivos de mapeo de objetos y DLLs (Dinamic Link Library – Librería de Enlace Dinámico) que los procesos están usando. La mitad superior de cada espacio de direcciones de los procesos sólo contienen código del modo núcleo. Uno de los trabajos del administrador de memoria del ejecutivo es mantener la ilusión de un gran espacio de direcciones usando técnicas de manejo de memoria virtual como el de paginación por demanda.

2.1.3.7 Llamada de procedimiento local

La llamada de Procedimiento Local (LPC, de aquí en adelante por sus siglas en inglés, Local Procedure Call) es un mecanismo de paso de mensajes usado para la comunicación entre procesos de la misma máquina. Los LPCs son usados principalmente para proteger los subsistemas, éstos serán descritos más adelante, y a sus clientes. Los manejadores de dispositivo no tienen acceso a los LPCs.

2.1.3.8 Administrador de E/S

Este componente del ejecutivo transforma en rutinas de manejadores las peticiones de E/S del modo usuario y el modo núcleo. El manejador de E/S puede comunicarse con todos los manejadores de la misma forma. Esto hace que el administrador de E/S no tenga que conocer los detalles del hardware.

Con esto hemos terminado la explicación de los componentes del sistema operativo que operan en el *modo núcleo*. A continuación daremos una explicación general de los subsistemas que operan en el modo usuario del sistema operativo.

2.2 Extensiones de la base del sistema operativo

Los componentes del ejecutivo no implementan una interfaz de usuario o definen alguna política externa de seguridad. Tampoco ofrecen una interfaz de programación. La base de los componentes del modo núcleo sólo proporciona una plataforma genérica de sistema operativo.

En la arquitectura de Windows, los subsistemas protegidos fueron implementados como un grupo de procesos privilegiados del modo usuario, en la figura D.4. se muestra un diagrama a bloques de la interacción entre los subsistemas que operan en el modo usuario con el resto de la arquitectura del sistema operativo Windows.

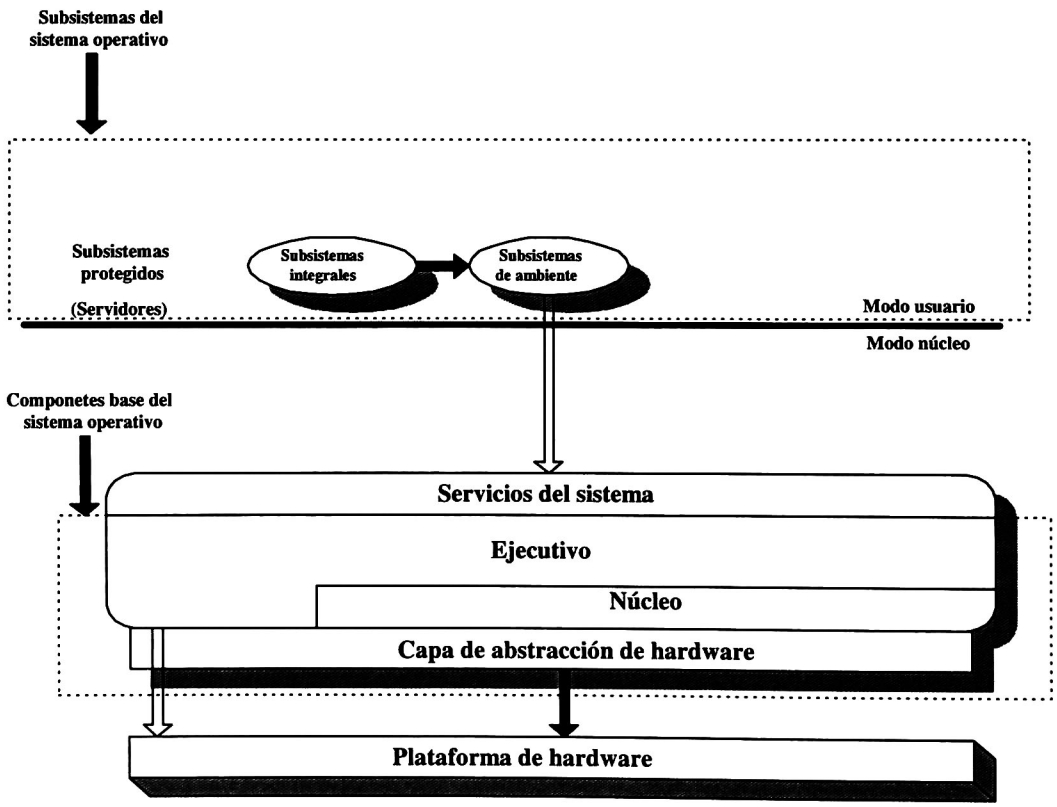


Figura D.4. Interacción entre los subsistemas que operan en modo usuario con el resto de la arquitectura del sistema operativo Windows.

Los subsistemas protegidos están divididos en dos categorías que son:

- Subsistemas integrales
- Subsistemas de ambiente

A continuación se dará una descripción de estos subsistemas.

2.2.1 Subsistemas integrales

Un subsistema integral desempeña algunas funciones necesarias del sistema. Algunas responsabilidades de estos subsistemas son:

- Definir la política de seguridad para el sistema
- Las cargas del administrador de control de servicios, supervisores, las descargas confiables de componentes del sistema tales como servicios y manejadores
- El localizador de RPC (Remote Procedure Call – Llamada de Procedimiento Remoto) da soporte a aplicaciones distribuidas que usan RPC

2.2.2 Subsistemas de ambiente

El trabajo de un subsistema de ambiente es proporcionar una interfaz de programación y ejecución para programas de aplicación. Actualmente Windows proporciona los siguientes subsistemas:

- El subsistema Win32 implementa la interfaz de programación modo nativo de Windows. Este subsistema es crucial para la operación de Windows. Las tareas de este subsistema son:
  a) Es propietario de la pantalla, teclado y ratón, éste maneja la consola y las E/S de la interfaz gráfica de usuario. Esto incluye las E/S tanto para los otros subsistemas como para las aplicaciones de usuarios
  b) El subsistema Win32 implementa la interfaz gráfica de usuario vista por los programadores y usuarios
  c) Expone la interfaz de aplicación del Win32 API que usan los programas de aplicación y otros subsistemas para interactuar con el ejecutivo
- El subsistema VMD (Virtual Disk Operating System Machine – Máquina Virtual del Sistema Operativo de Disco) permite aplicaciones MS-DOS (Microsoft Disk Operating System – Sistema Operativo de Microsoft) de 16 bits
- El subsistema WOW (Windows On Windows) soporta aplicaciones de 16 bits

- El subsistema POSIX (Portable Operanting System for Unix – Sistema Operativo Portable para Unix) proporciona una interfaz de aplicación para programas desarrollados bajo el estándar POSIX 1003.1
- El subsistema OS/2 crea un ambiente para aplicaciones OS/2 de 16 bits

Con esto hemos concluido con la descripción general de la arquitectura de Windows.

ACRÓNIMOS

ANSI	Instituto Nacional Americano de Estándares, *American National Standards Institute.*
API	Interfaz de Aplicación, *Application Interface.*
ATM	Modo de Transferencia Asíncrona, *Asynchronous Transfer Mode.*
BOP	Protocolo Orientado a Bit, *Bit Oriented Protocol.*
CPU	Unidad Central de Procesamiento, *Central Processing Unit.*
DLL	Librería de Enlace Dinámico, *Dinamic Link Library.*
DMA	Acceso Directo a Memoria, *Direct Memory Access.*
E/S	Entrada/Salida
FDL	Sección de Enlace de Datos, *Facility Data Link.*
HAL	Capa de Abstracción de Hardware, *Hardware Abstraction Layer.*
IEEE	Instituto de Eléctrica e Ingenieros Electrónicos, *Institute of Electrical and Electronics Engineers.*
ISO	Organización Internacional de Estándares, *International Standards Organization.*
LAPD	Procedimiento de Acceso de Enlace en el canal D, *Link Access Procedure on the D channel.*
LAN	Red de Área Local, *Local Area Network.*
LPC	Llamada a Procedimiento Local, *Local Procedure Call.*
LLC	Control de Enlace Lógico, *Logical Link Control.*
MAC	Control de Acceso al Medio, *Media Access Control.*
MOP	Protocolo Orientado a Mensage, *Message Oriented Protocol.*
MSDOS	Sistema Operativo de Disco de Microsoft, *Microsoft Disk Operating System.*
NDIS	Especificación de Interfaz de Manejador de Red, *Network Driver Interface Specification.*
NIC	Tarjeta de Interfaz de Red, *Network Interface Card.*
OSI	Interconexión de Sistemas Abiertos, *Open Systems Interconnection.*

POSIX Sistema Operativo Portable para Unix, *Portable Operating System for Unix.*

RAI Indicación de Alarma Remota, *Remote Alarm Indication .*

RPC Llamada a Procedimiento Remoto, *Remote Procedure Call.*

TDI Interfaz de Manejador de Transporte, *Transport Driver Interface.*

UML Lenguaje Unificado de construcción de Modelos, *Unified Modeling Language.*

VDM Máquina Virtual del Sistema Operativo de Disco, *Virtual Disk Operating System Machine.*

WOW Windows en Windows, *Windows On Windows.*

BIBLIOGRAFÍA

[1] ANSI T1.403. 1995. Telecomunications. Network to customer installation. DS1 metallic interface.

[2] ANSI T1.231. 1997. Telecomunications. Digital Hierarchy. Layer 1 in service digital transmission performance monitoring.

[3] BAKER Art, The windows NT device driver book: a guide for programmers, Prentice Hall 1997, 1ª. Edición.

[4] BOOCH Grady, Análisis y diseño orientado a objetos con aplicaciones, Addison-Wesley/Diaz de Santos 1996, 1ª. Edición.

[5] COLEMAN Derek, ARNOLD Patrick, BODOFF Stephanie, DOLLIN Chris, GILCHRIST Helena, HAYES Fiona, JEREMAES Paul, Object oriented development, the fusion method, Prentice Hall 1994, 1ª. Edición.

[6] COMER Duglas E., Redes globales de información con internet y TCP/IP. Principios básicos, protocolos y arquitectura, Prentice Hall 1998, 3ª. Edición.

[7] COPLIEN James O., SCHMIDT Douglas C., Pattern languajes of program design, Addison-Wesley Longman 1998, 4ª. Edición.

[8] DEITEL Harvey M., Introducción a los sistemas operativos, Addison-Wesley Iberoamericana 1993, 2ª.Edición.

[9] FOWLER Martín, Analysis patterns reusable object models, Addison-Wesley Longman 1999, 8ª. Edición.

[10] FOWLER Martín, SCOTT Kendall, UML gota a gota, Addison-Wesley Longman 1999, 1ª. Edición.

[11] HOFMEISTER Christine, NORD Robert, SONI Dilip, Applied software architecture, Addison-Wesley Longman 2000, 2ª. Edición.

[12] HOPCROFT John E., ULLMAN Jefrey D., Introducción a la teoría de autómatas, lenguajes y computación, Cecsa 1996, 2ª. Edición.

[13] ITU-T Recommendation Q921, ISDN user network. Data layer specification.

[14] KARTALOPOULOS Stamatios V, Understanding SONET/SDH and ATM communications networks for the next millennium, IEEE Press 1999, 1ª. Edición.

[15] LARMAN Craig, UML y patrones, introducción al análisis y diseño orientado a objetos, Prentice Hall 1999, 1ª. Edición.

[16] RUBINI Alessandro, Linux device drivers, O'reilly & Associates Inc. 1998, 1ª Edición.

[17] SCHMULLER Joseph, Aprendiendo UML en 24 horas, Prentice Hall 2000, 1ª. Edición.

[18] TANENBAUM Andrew S., WOODHULL Albert S., Sistemas operativos, diseño e implementación, Prentice Hall 1998, 2ª. Edición.

[19] HERMOSILLO J., FIGUEROA M., SILVA J., TORRES D., GARCIA A., Desarrollo e implementación de un protocolo para ciertas aplicaciones de voz y datos. XX Congreso internacional académico de ingenieria eléctrica, ELECTRO 98. Instituto tecnológico de Chihuahua. Chihuahua, Chih. Mexico, 26-30 octubre 1998. pp 93-98, ISSN 1405-2172.

[20] VLISSIDES John M., COPLIEN James O., KERTH Norman L., Pattern Languajes of program design 2, Addison-Wesley Longman 1998, 4ª. Edición.

[21] YOURDON Edward, Análisis estructurado moderno, Prentice Hall 1993, 1ª. Edición.

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL IPN
UNIDAD GUADALAJARA

El Jurado designado por la unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, aprobó la Tesis "Arquitectura y diseño de una plataforma para el desarrollo de aplicaciones para la evaluación de circuitos de alta integración" del Sr. ARTURO JOSE GARCIA GARCIA, el día 5 de Diciembre de 2001.

Dr. Arturo del Sagrado Corazón Sánchez Carmona
Investigador Cinvestav 3 B
CINVESTAV DEL IPN
Guadalajara

Dr. Manuel Edgardo Guzmán Rentería
Investigador Cinvestav 3A
CINVESTAV DEL IPN
Guadalajara

Dr. Deni Librado Torres Román
Investigador Cinvestav 3 A
CINVESTAV DEL IPN
Guadalajara