UNIVERSIDAD NACIONAL DE INGENIERÍA

Facultad de Ingeniería Mecánica

"Diseño e Implementación de un Sistema Cliente - Servidor de OPC Autónomo"

TESIS

Para optar el Título Profesional de:

INGENIERO MECATRÓNICO

KEVIN SCOTT ARZAPALO CAMAVILCA

Promoción 2001 – I

Lima – Perú

2003

Dedico este trabajo a mi padre *Antonio Arzapalo Campos*, a mi hermana y a mi madre que siempre la llevo conmigo.

ÍNDICE

CAPITULO II:

PRÓLOGO

Este presente trabajo consta de cinco capítulos, de los cuales el primer capítulo viene hacer el análisis del problema generado al tratar de desarrollar aplicaciones para integrar sistemas, luego se propone usar un estándar para el desarrollo de las mismas. El estándar propuesto es OPC (Ole para Control de Procesos), también se muestra sus interfaces disponibles, las ventajas obtenidas al usar el estándar y la comparación con otras tecnologías existentes en la industria de la comunicación industrial.

El segundo capítulo viene a ser el desarrollo seguido para la implementación del software OPC Client o Cliente de OPC. Es importante leer los Apéndices A y B inicialmente antes de entrar a este capítulo, ya que muchos términos son expuestos por primera vez y es importante saberlos con anterioridad. Este capítulo comienza con la definición de los objetos encargados de la funcionalidad de OPC, para luego proceder con el diseño e implementación de cada funcionalidad provista por OPC. El diseño se basa en la descomposición orientada a objetos y en algoritmos.

El capítulo tercero muestra la interfaz de usuario que ofrece el software Cliente de OPC desarrollado en el capítulo anterior, luego se muestra todas las funcionalidades y herramientas que se obtiene al usar dicho software. También se podrá ver en detalle de todas las operaciones que ofrece OPC para el acceso de datos, los cuales fueron implementados en el Cliente de OPC. Finalmente se tendrá la funcionalidad de OPC desarrollado para el ambiente de Excel.

El capítulo cuarto consta de todo el procedimiento realizado para la elaboración del Servidor de OPC para el microcontrolador PIC 16F873, este procedimiento consta de la definición del hardware, definición del protocolo, implementación del algoritmo de comunicación y finalmente la implementación de las funcionalidades de OPC para el Servidor de OPC. Para su mejor entendimiento es necesario leer conjuntamente con este capítulo el Apéndice C.

El quinto capítulo muestra las aplicaciones en la industria donde se pueden aplicar los dos productos desarrollados en los capítulos anteriores (Cliente y Servidor de OPC). Estas aplicaciones mostrarán en forma general como es posible utilizar e introducir, en especial el producto para el Cliente de OPC, a la industria de la comunicación industrial, con el fin de integrar los niveles existentes en una planta industrial. Las limitaciones y especificaciones técnicas de cada producto desarrollado también son incluidas en este capítulo. Este capítulo finaliza con unas pruebas de rendimiento realizadas tanto al Cliente y al Servidor de OPC.

INTRODUCCIÓN

Los cambios acelerados en el control industrial y en la tecnología de la automatización de los últimos años, son consecuencia de la demanda que exige la industria para afrontar nuevos retos. Uno de estos retos es tener un flujo de información a todo nivel, es decir entre la oficina, fábrica y la producción; en otras palabras viene a ser la integración de estos niveles.

Esta integración se puede dar por medio de integraciones horizontales (el cual permite la comunicación entre los componentes distribuidos) e integraciones verticales, para optimizar los procesos de planeamiento de producto, desarrollo, manufactura y ventas. Esta optimización es realizada a través de un consistente flujo y disponibilidad de datos de los niveles de campo (sensores, actuadores, etc.), de célula y control (PLC, DCS, SCADA, etc), hacia los niveles de oficina (Administrativa y Producción) y viceversa.

La importancia de este trabajo esta basado en estos requerimientos de integración de la industria, contando para ello con el desarrollo de un sistema que permita tanto la integración horizontal como la vertical, basado en una interface estándar aceptado por la gran mayoría de fabricantes de aplicaciones para la industrial de la automatización (Ole para Control de Procesos - OPC). El sistema mencionado consta de un Cliente de OPC, el cual permitirá acceder a los datos de todos los servidores de OPC disponibles en la industria. Además, dicho sistema consta de un Servidor de OPC para un microcontrolador PIC 16F873, el cual será usado como una fuente de datos.

El propósito del presente trabajo es obtener un software que se pueda aplicar a la industria de la comunicación industrial, para realizar integraciones (horizontales y verticales) que permitan la optimización de todos los recursos. Conocer la nueva tecnología en la comunicación industrial para realizar futuros trabajos, también es el propósito de este trabajo.

Con este trabajo se espera demostrar el buen rendimiento de la comunicación obtenida al usar OPC para distintas configuraciones existentes, sus limitaciones y restricciones serán mostradas en el desarrollo de la presente tesis. El método de trabajo utilizado para el desarrollo del software se basa inicialmente en su diseño, para dicho propósito se usa la descomposición orientada a objetos, el cual permitirá obtener su algoritmo para finalmente proceder con la implementación.

CAPÍTULO I

PERSPECTIVA GENERAL DEL PROBLEMA

Este capítulo trata de analizar un problema que es muy frecuente al implementar aplicaciones para integrar sistemas, ya que esta tarea involucra muchos sistemas de diferentes fabricantes y de diferente índole. Un ejemplo de este problema se puede dar al tratar de acceder a la información dentro de una planta industrial (cantidad de materia prima consumida, cantidad de merma, etc.), pero este acceso no es rápido ni fácil. Esta tarea se puede llevar a cabo, pero al tener muchos sistemas este problema se complica y eleva el costo de su realización.

En la figura 1.1 se puede observar todos los niveles jerárquicos acorde con su velocidad de transmisión y tiempo de respuesta, que se puede encontrar dentro de cualquier red industrial. Esto implica que el intercambio de información puede tener varias redes de comunicación dentro de una misma planta industrial. Es por eso que el problema de integración se complica aún más.

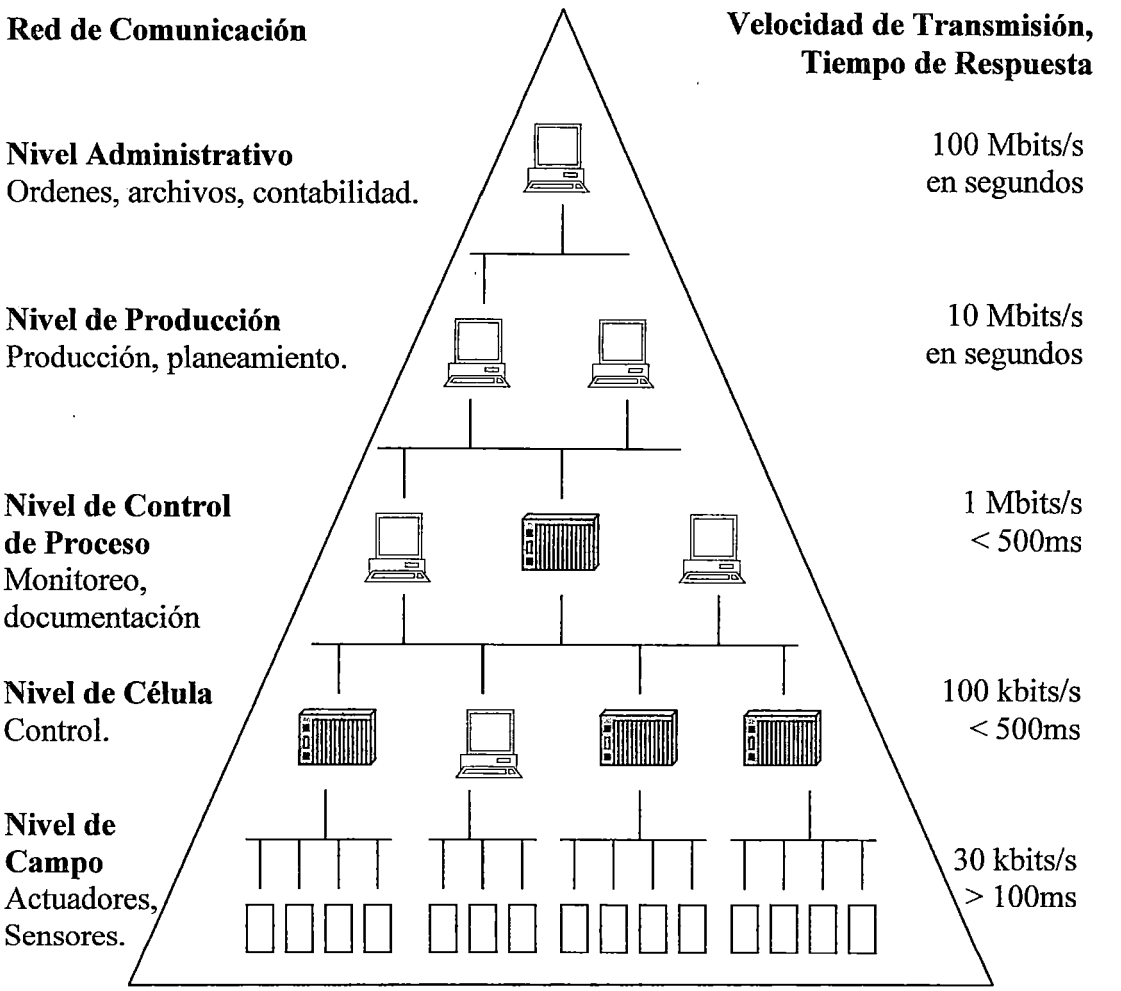


Figura 1.1 Intercambio jerárquico de información en una red de comunicación industrial

1.1 Análisis del Problema de Integración

El problema principal radica en que un software de aplicación debería fácilmente comunicarse con los dispositivos digitales de una planta y además con otras aplicaciones existentes, pero este contexto no se da a menudo.

Por lo general se encuentra que los sistemas propietarios no se pueden comunicar con otros sistemas que son muy comunes y la ausencia de cualquier estándar genera, que los fabricantes desarrollen hardware propietario y soluciones para software propietario.

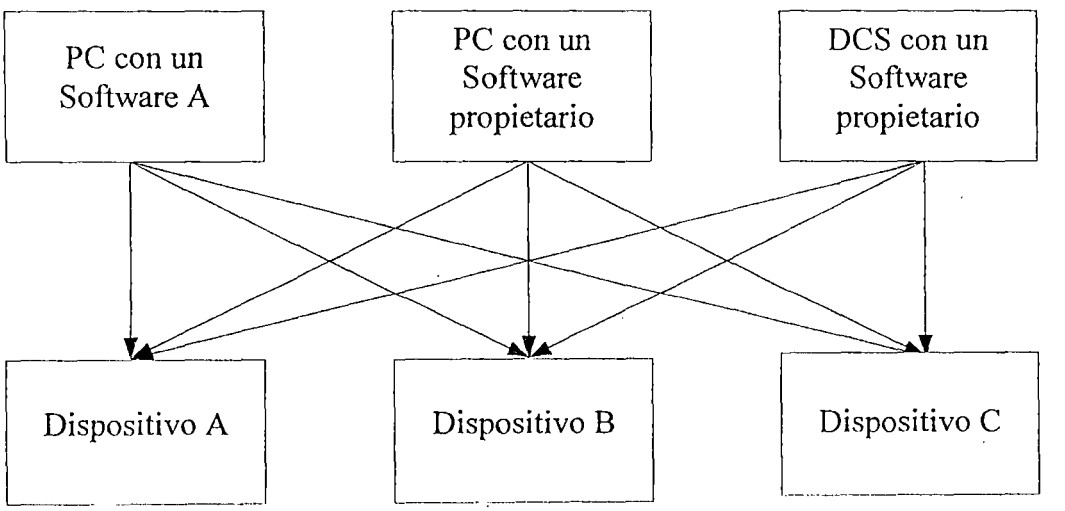


Figura 1.2 Esquema de la comunicación usando un sistema propietario no estándar

Una solución podría ser crear *drivers* para los clientes e interfaces para cada dispositivo, como se muestra en la figura 1.2, pero la gran variedad de tipos de dispositivos de control y paquetes de software que necesitan comunicarse hacen que esta solución no sea práctica.

1.1.1 Planteo de la Solución

La solución, tal como se muestra en la figura 1.3, es implementar aplicaciones sobre la base de un estándar que provea una tecnología real de software de fácil conexión y desconexión, para el control de procesos y la automatización de la fábrica donde cada sistema, cada dispositivo y cada *driver* puedan conectarse y comunicarse. Basado en tal estándar se hace posible el concepto de integración total, verdaderamente abierto y de fácil comunicación entre sistemas y dispositivos.

El nombre de ese estándar es OPC, con este estándar se evita el alto costo de múltiples servidores propietarios, *drivers* e interfaces que eran necesarios para la comunicación de sistemas en el pasado.

Este estándar será la base para el desarrollo de este presente proyecto, ya que las dos aplicaciones diseñadas e implementadas en los capítulos posteriores obtendrán todas las características que ofrece OPC y cumplirán con el mismo propósito, el cual es la integración total de sistemas.

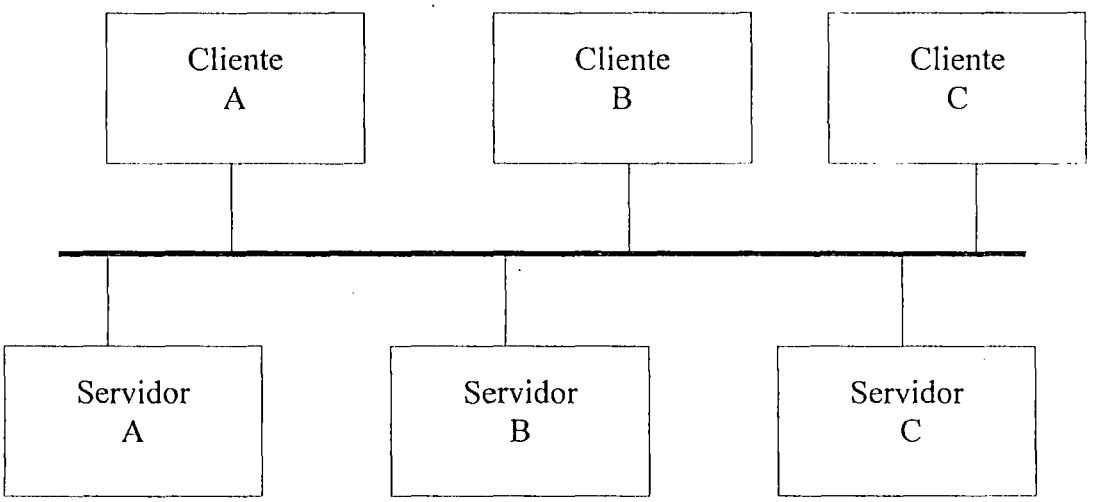


Figura 1.3 Esquema de la comunicación usando un sistema estándar

1.2 OLE para Control de Procesos OPC

Basado en OLE de Microsoft (ahora *ActiveX*), en las tecnologías COM (*Modelo de Objeto Componente*) y DCOM (*Modelo de Objeto Componente Distribuido*), OPC consiste en un grupo de interfaces, propiedades y métodos para ser usados en el control de proceso y aplicaciones de automatización. OPC provee una interface común para la comunicación con diversos dispositivos de control de proceso (ver figura 1.4).

El propósito de OPC es obligar a los proveedores de la industria de la automatización poner a todos sus *drivers* y dispositivos una forma estándar.

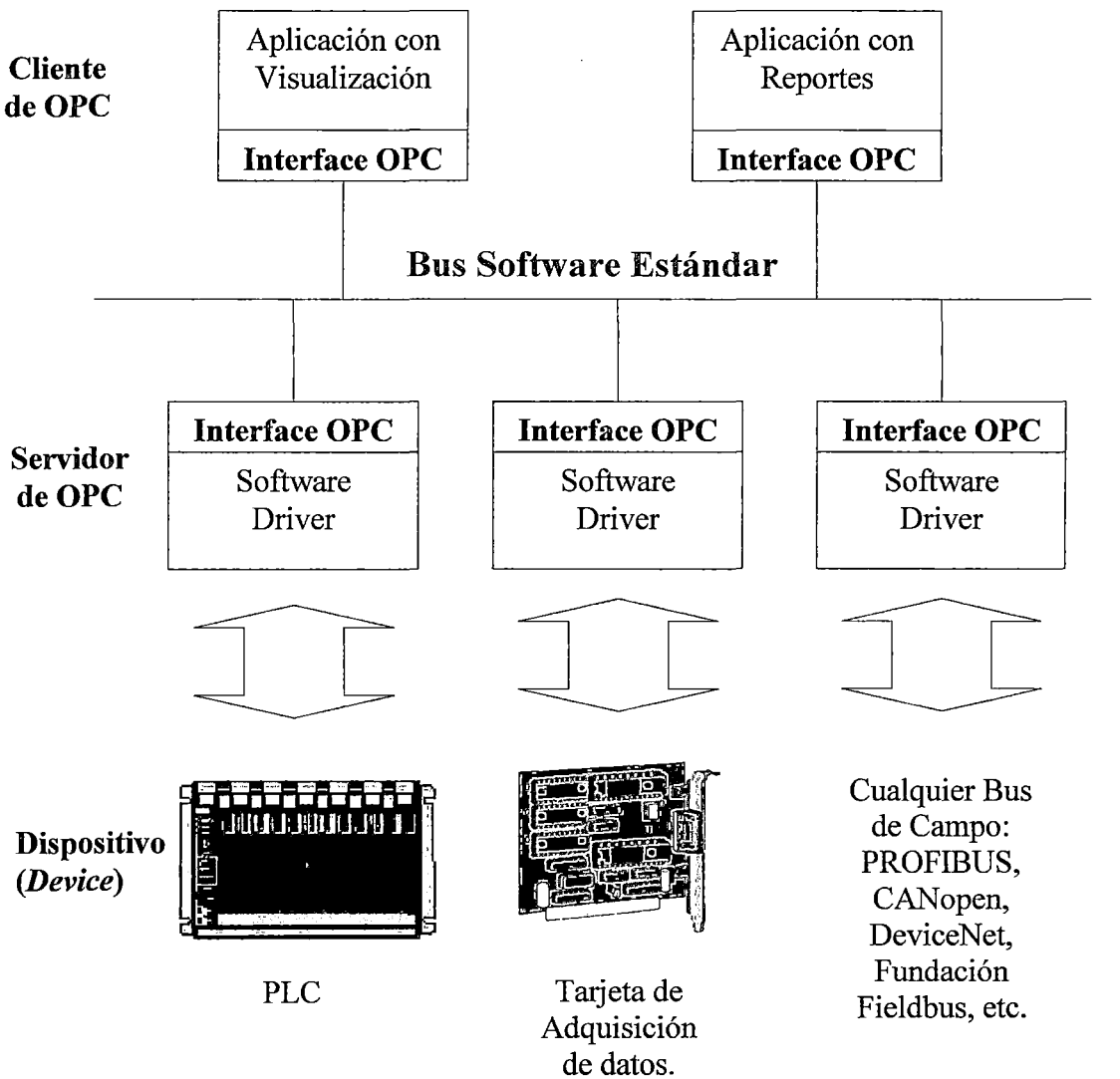


Figura 1.4 Estructura de un *Bus Software* estándar

Esencialmente, OPC define una interface común que permite que el desarrollo de la interface sea hecho una vez y luego sea fácilmente vuelto a usar.

El estándar de OPC requiere de proveedores de hardware para proveer la colección y distribución de datos (ver figura 1.4 PLC, tarjetas de adquisición de datos, etc.). Estos están diseñados para acceder a los datos internos del dispositivo de manera eficiente. Estos dispositivos luego se volvieron ***Servidores de OPC***, los cuales proveen constantemente los datos a los ***Clientes de OPC*** (ver figura 1.4).

1.2.1 Perspectiva General de OPC

OPC es la base tecnológica para el conveniente y eficiente lazo de componentes de automatización con un control del hardware y dispositivos de campo. Además, OPC provee la condición para la integración de productos de oficina y sistemas de información como son los niveles Administrativo y de Producción con los niveles de Control de Procesos, Célula y de Campo (ver figura 1.1).

Hoy en día OPC está basado en DCOM, por tal razón todas las definiciones de DCOM, como objetos, interfaces, métodos, etc, son aplicados a OPC.

El estándar de OPC está basado en especificaciones técnicas de acceso gratuito que definen un grupo de interfaces estándar para diferentes campos de aplicación en la tecnología de la automatización.

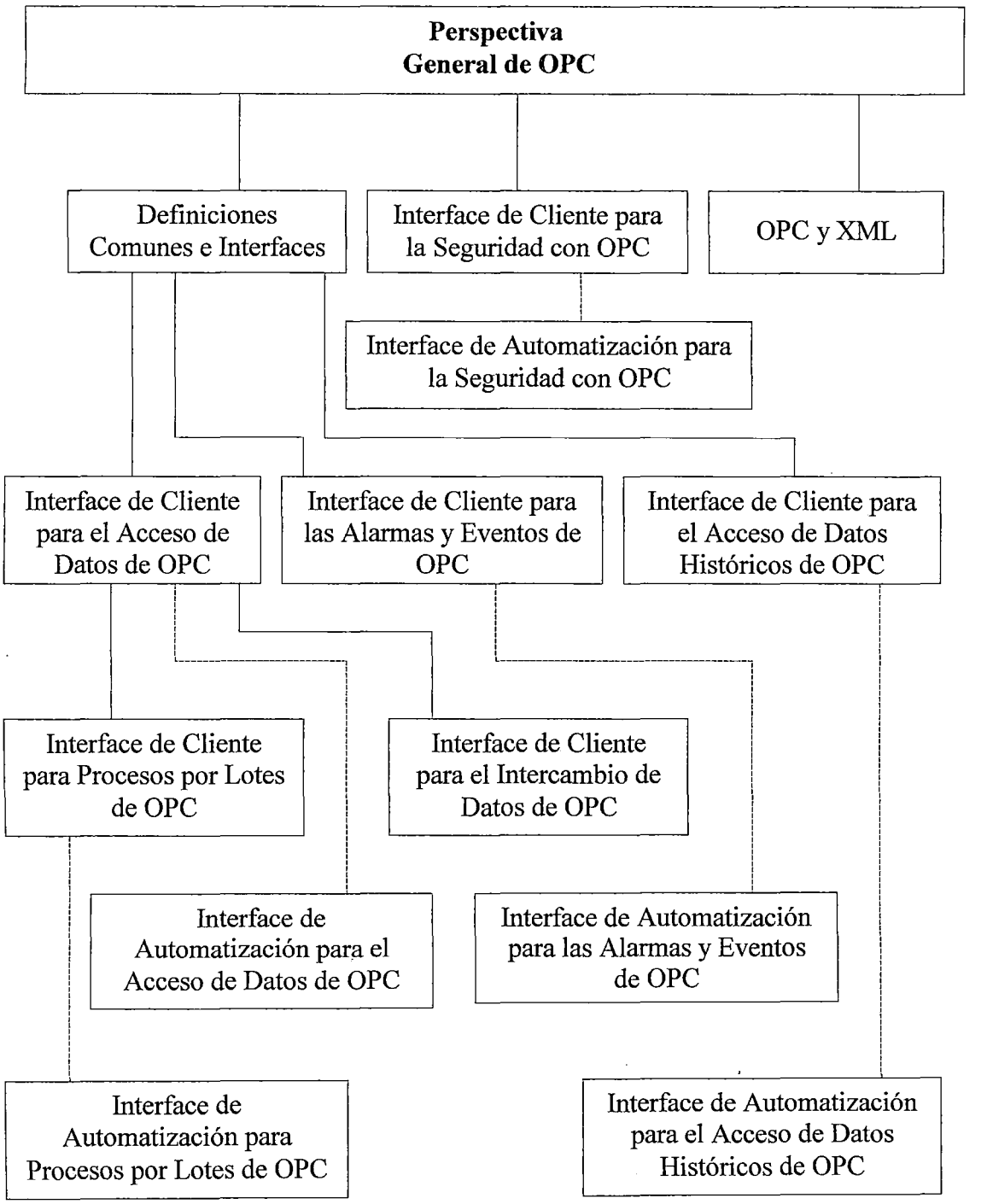


Figura 1.5 Perspectiva general de todas las especificaciones de OPC

Especificación	Contenido	Última Publicación
OPC Overview (Perspectiva General de OPC)	Descripción general de los campos de aplicación de las especificaciones de OPC.	versión 1.00
OPC Common Definitions and Interfaces (Definiciones Comunes e Interfaces de OPC)	Definición de los temas concernientes a un número de especificaciones.	versión 1.00
OPC Data Access Specification (Acceso de Datos de OPC)	Definición de una interface para la lectura y escritura de los datos en tiempo real.	versión 2.05
OPC Alarms and Events Specification Especificación (Alarmas y Eventos de OPC)	Definición de una interface para el monitoreo de eventos.	versión 1.02
OPC Historical Data Access Specification (Especificación Acceso de Datos Históricos de OPC)	Definición de una interface para el acceso de datos históricos.	versión 1.1
OPC Batch Specification (Especificación para Procesos por Lotes de OPC)	Definición de una interface para el acceso de datos para un proceso por lotes o tipo *Batch*. Esta basado en la Especificación *OPC Data Access Specification.*	versión 2.0
OPC Security Specification (Seguridad de OPC)	Definición de una interface para la configuración y utilización de políticas de seguridad.	versión 1.0
OPC and XML (OPC y XML)	Integración de OPC y XML para la construcción de aplicaciones Web.	versión 0.60
OPC Data eXchange (DX) (Intercambios de Datos de OPC)	Comunicación entre servidores en pleno proceso.	En construcción
OPC Command Execution Interface (Ejecución de Interfaces y Comandos de OPC)	Definición de una interface para la eliminación de comandos y supervisión de su ejecución.	versión 0.10

Tabla 1.1. Especificaciones de OPC con sus contendidos y su última publicación

La figura 1.5 muestra las especificaciones actualmente disponibles, aquellos que están en construcción y su relación entre ellos. La tabla 1.1 muestra el contenido de las diferentes especificaciones y la última versión publicada por la Fundación OPC (grupo encargado en realizar las especificaciones de OPC).

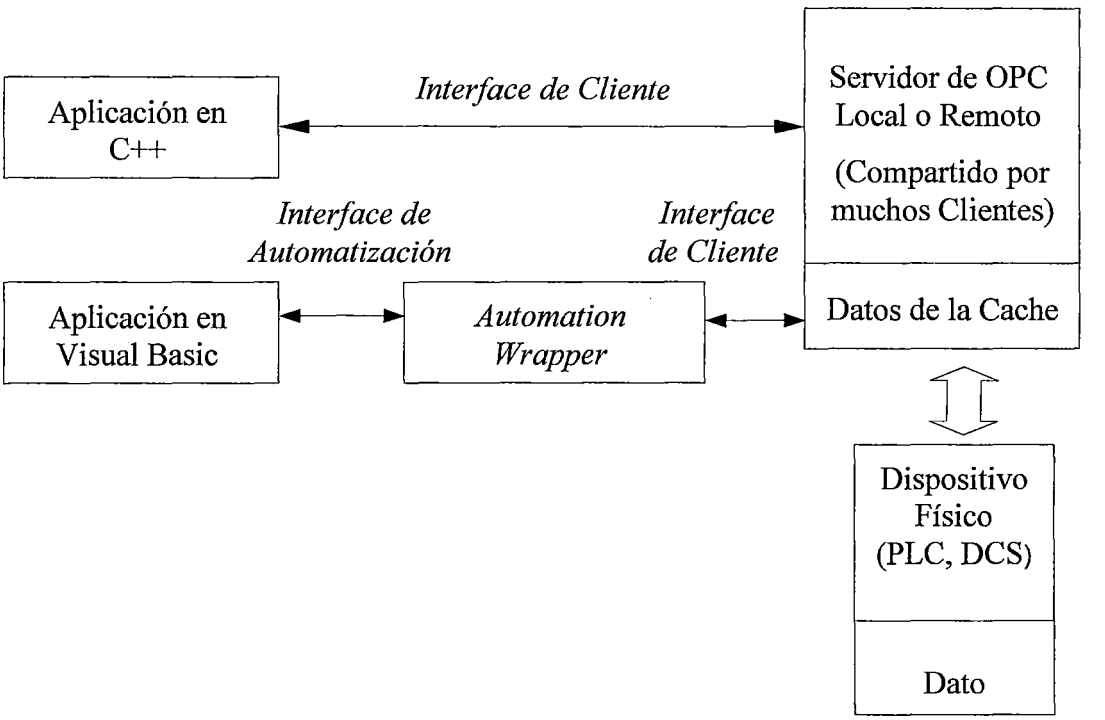


Figura 1.6 Formas de acceso a un Servidor de OPC

Los clientes de OPC pueden acceder a los servidores de OPC usando dos tipos de interfaces: *OPC Custom Interface* (Interface de Cliente) u *OPC Automation Interface* (Interface de Automatización). Tal como se puede apreciar en la figura 1.6.

1.2.1.1 Interface de Cliente (*Custom Interface*)

Esta interface está definida en las especificaciones de OPC y es usado en lenguajes de programación que soportan el concepto de punteros de funciones, como C/C++.

1.2.1.2 Interface de Automatización (*Automation Interface*)

La Interface de Automatización está definida en las especificaciones de OPC y es usado para lenguajes de programación como Visual Basic o Delphi los cuales no usan ningún puntero de función.

El *Automation Wrapper* mostrado en la figura 1.6 es provisto por la Fundación de OPC, el cual es un DLL que llama a la Interface de Cliente desde los métodos de la Interface de Automatización.

1.2.2 Ventajas Obtenidas al usar el Estándar OPC

La tabla 1.2 describe las ventajas que se pueden obtener al usar OPC, en el desarrollo de las aplicaciones Cliente y Servidor descritas en los siguientes capítulos.

Categoría	Ventajas
Fabricantes de Hardware	El producto puede ser usado por todos los sistemas compatibles a OPC que existen en el mercado. El tiempo invertido para el soporte técnico del hardware es reducido porque solamente se tiene un producto para ello.
Fabricantes de Software	El software desarrollado puede ser usado con todos los dispositivos y protocolos de comunicación existentes en el mercado que están basados en OPC. Los fabricantes ya no tienen que desarrollar soluciones específicas (*drivers* específicos), ya que la interface es estándar y permite la interoperabilidad, y no se necesita familiarizarse con las especificaciones de otros dispositivos y protocolos de comunicación. El tiempo invertido para el soporte técnico es considerablemente reducido porque muchos productos (*drivers* específicos) ya no son utilizados.
Integrador de Sistemas	La flexibilidad en la elección de productos es considerablemente alto. El tiempo necesario para integración y entrenamiento es reducido considerablemente, porque OPC provee una interface estándar el cual es similar entre todos los productos.
Usuario	OPC provee una flexibilidad adicional (distribución de componentes, uso de nuevas tecnologías, facilidad para escoger entre productos, etc.) durante el diseño del sistema integral ya que los productos de varios fabricantes pueden ser combinados. Por consecuencia se obtendrá en el futuro a una reducción notable de los costos con el crecimiento de la calidad y comodidad para el usuario.

Tabla 1.2 Ventajas obtenidas al usar OPC

1.2.3 Comparación y Evaluación entre otras Tecnologías

La tabla 1.3 realiza una comparación entre OPC, DDE y los *drivers* de un producto específico. Las tablas 1.2 y 1.3 muestran los fundamentos necesarios que fueron tomados en cuenta para la utilización del estándar OPC para el desarrollo del presente proyecto.

	***Drivers* de un Producto Específico**	**DDE**	**OPC**
Estandarización	No	No	Sí
Interoperabilidad	No	Limitado para el estándar DDE o para el respectivo dialecto (ejemplo, FastDDE, AdvancedDDE).	Sí
Rendimiento	Muy alto rendimiento porque las soluciones son optimizadas para una específica aplicación.	DDE es basado en el intercambio de mensajes. Esto es lento. Los dialectos de DDE mejoran la eficiencia ya que estos ponen más datos en cada mensaje.	Muy alto rendimiento.
Acceso Remoto	No	Sí (NetDDE)	Sí (DCOM)
Intercambio Comercial	El intercambio comercial es reducido.	Los servidores de DDE desaparecerán en el futuro.	La participación de las soluciones con OPC dominará en el futuro.

Tabla 1.3 Comparación de OPC con otras tecnologías (*continúa...*)

	Drivers de un Producto Específico	DDE	OPC
Interacción	Generalmente no está disponible.	No	Sí, OPC define un largo número de métodos, el cual puede ser usado para el intercambio adicional de información entre el cliente y el servidor.
Complejidad	Muchos fabricantes proveen herramientas (*toolkits*), para crear *drivers* para un correspondiente sistema muy fácilmente. Un nuevo sistema requiere otro tiempo adicional para aprenderlo.	Las herramientas (*toolkits*) son disponibles por el fabricante. La programación de aplicaciones DDE sin una herramienta es muy complicada.	Aprender la programación de DCOM lleva mucho tiempo de entrenamiento, Hay herramientas (*toolkits*) que pueden ayudar a minimizar el tiempo.
Disponibilidad para otras plataformas	Hay solamente un mínimo de sistemas disponibles para diferentes plataformas.	DDE es solamente disponible para sistemas operativos Windows.	Los componentes OPC están limitados para sistemas donde DCOM es disponible. La nueva especificación OPC y XML que están en construcción, adicionalmente permitirán que el usuario de los componentes de OPC pueda trabajar en sistemas operativos sin DCOM.

Tabla 1.3 (*Continuación de la página 17.*) Comparación de OPC con otras tecnologías

CAPITULO II

DISEÑO E IMPLEMENTACIÓN DEL CLIENTE DE OPC

Uno de los objetivos planteados para el presente proyecto es el desarrollo de un software cliente de OPC, con algunas de las siguientes características: el software debe proporcionar toda la funcionalidad de OPC referida al acceso de datos, debe ser amigable con el usuario y debe de estar dirigida a una variedad de usuarios, es decir que debe ser muy genérico.

Teniendo en cuenta estas características iniciales se ha desarrollado el software llamado "OPC Client". En este capítulo se tendrá una visión clara de los procedimientos seguidos para el desarrollo del presente software y para la implementación de la funcionalidad de OPC para el acceso de datos. El software OPC Client usa la Interface de Automatización (*Automation Interface*, ver Apéndice A) para la interacción con un servidor de OPC.

2.1 Diseño e Implementación de la Funcionalidad de OPC

2.1.1 Diseño de los Objetos Encargados de la Funcionalidad de OPC

La programación del software OPC Client está orientada a objetos, los objetos permiten el manejo de toda la funcionalidad del acceso de datos de la Interface de Automatización de OPC con la interfaz de usuario del software. El modelo de los objetos diseñados para este fin es mostrado en la figura 2.1 y sus respectivas descripciones en la tabla 2.1.

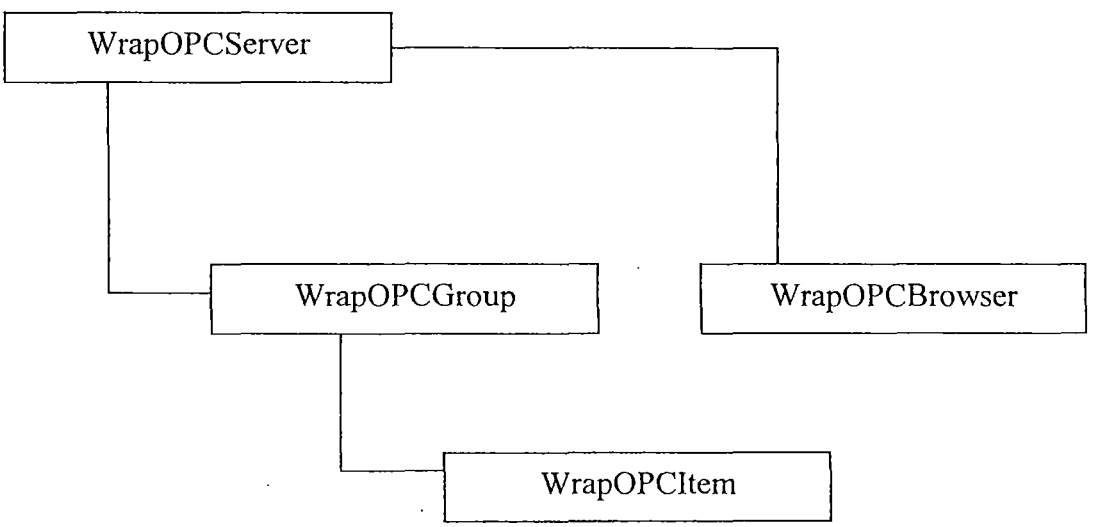


Figura 2.1 Modelo de la jerarquía de los objetos creados para el software OPC Client

Objeto	Descripción
WrapOPCServer	Este objeto se crea antes de poder usar los demás objetos. Además, encapsula toda la funcionalidad de los objetos *OPCServer* y *OPCGroups* de la Interface de Automatización. El objeto *WrapOPCServer* contiene la colección de objetos *OPCGroups* y permite crear un objeto *WrapOPCBrowser*.
WrapOPCGroup	Encapsula toda la funcionalidad de los objetos *OPCGroup* y *OPCItems* de la Interface de Automatización. Además, contiene una colección de objetos *OPCItems* que están referidos a un objeto *OPCGroup*.
WrapOPCItem	Encapsula toda la funcionalidad del objeto *OPCItem* de la Interface de Automatización. Este objeto contiene las definiciones del ítem como: valor actual, información del estado y tiempo de actualización.
WrapOPCBrowser	Encapsula toda la funcionalidad del objeto *OPCBrowser* de la Interface de Automatización. Este objeto se encarga de la búsqueda de los nombres de los items dentro del *namespace*. Existe un sólo objeto *WrapOPCBrowser* por cada objeto *WrapOPCServer* creado.

Tabla 2.1 Descripción de los objetos del software OPC Client

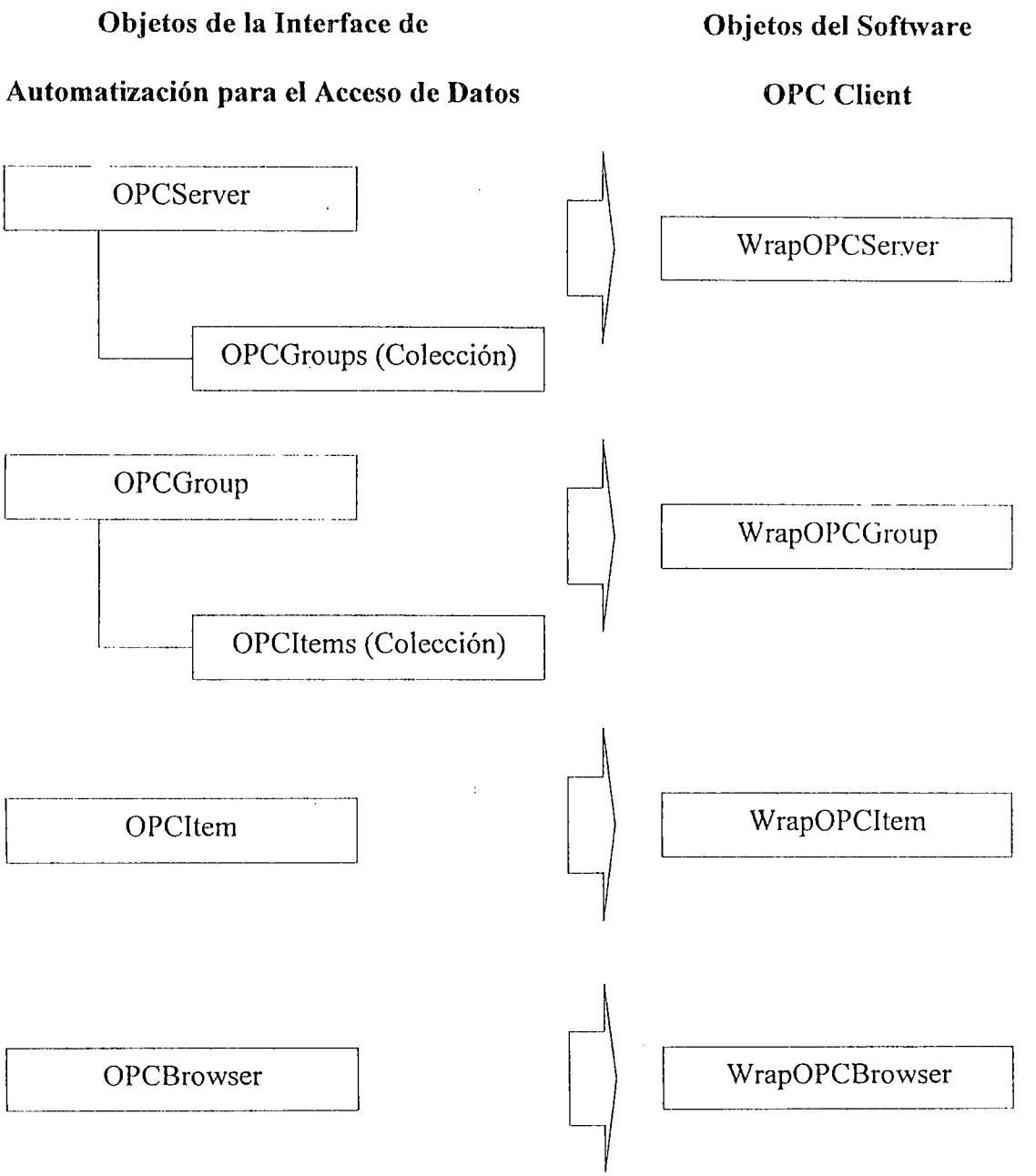


Figura 2.2 Interrelación entre los objetos creados y los objetos de la Interface de Automatización

La figura 2.2 muestra la relación que existe entre los objetos creados para el software OPC Client con los objetos de la Interface de Automatización (ver modelo de los objetos de la Interface de Automatización del Apéndice A). Los objetos creados para el software OPC Client son implementados en módulos de clase que llevan el mismo nombre.

2.1.1.1 Definición del objeto *WrapOPCServer*

El objeto *WrapOPCServer* es definido en la figura 2.3, donde se puede ver claramente todas sus variables privadas, funciones privadas y públicas.

Todas las características de cada elemento del objeto son definidas en las tablas 2.2, 2.3 y 2.4 donde las variables privadas, funciones privadas y públicas respectivamente, son descritas a detalle.

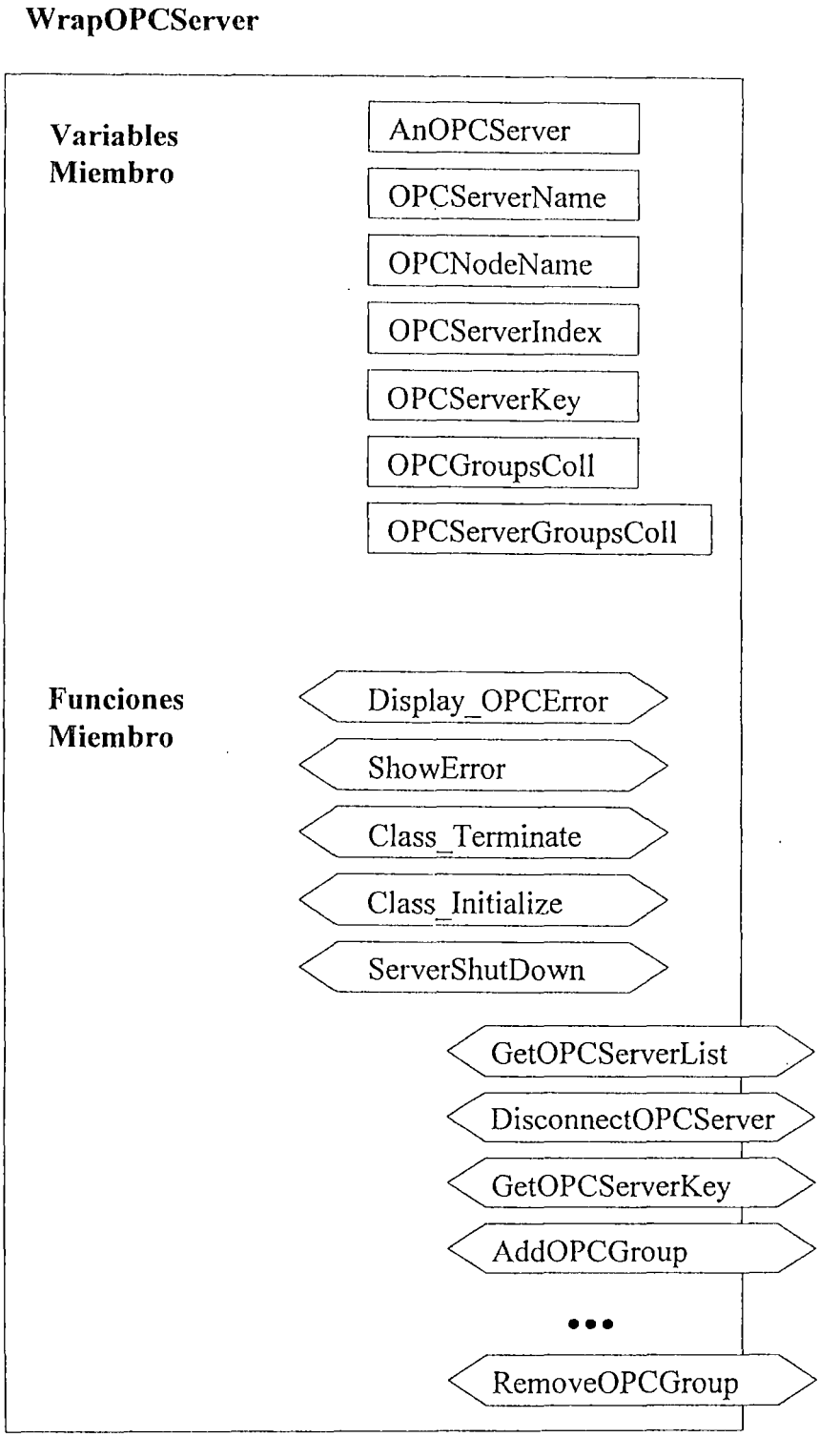


Figura 2.3 Diagrama del objeto *WrapOPCServer* (notación de Ege)

Nombre	Descripción
AnOPCServer	Objeto *OPCServer* (ver Apéndice A) declarado usando *WithEvents* para trabajar con los eventos del objeto.
OPCServerName	El nombre del servidor (*ProgID,* ver Apéndice A). Esta cadena es ingresada hacia la función *ConnectOPCServer* de este módulo.
OPCNodeName	El nombre del nodo de la maquina remota. Esta cadena es ingresada como un parámetro a la función *ConnectOPCServer* de este módulo.
OPCServerIndex	Variable numérica, que sirve para crear una clave única para cada objeto *WrapOPCServer*.
OPCServerKey	Esta clave se usa como una referencia para cada objeto *WrapOPCServer*. Además, es usado por la interfaz de usuario como una clave única para el control *TreeView* (ver sección 2.2).
OPCGroupsColl	Este contiene la colección de los objetos *OPCGroups* de la Interface de Automatización. Se usa para adicionar nuevos grupos hacia el servidor de OPC. Este es utilizado en la función *AddOPCGroup* de este módulo.
OPCServerGroupsColl	Es una nueva colección de objetos *WrapOPCGroup*. El objeto *WrapOPCGroup* envuelve los métodos y propiedades del objeto *OPCGroup* y de la colección *OPCItems* de la Interface de Automatización.

Tabla 2.2 Descripción de las variables privadas del objeto *WrapOPCServer*

Nombre	Descripción
Display_OPCError	Este procedimiento muestra cualquier error generado por OPC, COM o Visual Basic, que es encontrado por el *manejador de excepciones* de cada función del objeto *WrapOPCServer* (ver manejo del error de la especificación "Data Access Automation Interface Standard" versión 2.02).
ShowError	Esta función devuelve una expresión del error generado por OPC, COM o Visual Basic, que es encontrado por el *manejador de excepciones* de cada función del objeto *WrapOPCServer* (ver manejo del error de la especificación "Data Access Automation Interface Standard" versión 2.02). Este error se mostrará por medio del control *ListView* "*lvListViewH*" (ver sección 2.2).
Class_Terminate	Cuando el objeto *WrapOPCServer* es borrado esta función borrar todas las conexiones que existen dentro del objeto, es decir grupos e items.
Class_Initialize	Crea el nuevo objeto *OPCServer* cuando el objeto *WrapOPCServer* es inicializado.
ServerShutDown	Este evento se genera cuando el servidor de OPC esta siendo cerrado. Este evento notifica del cierre producido y procede a desconectarse con el servidor de OPC y todos sus objetos creados. Este evento solamente es generado cuando el servidor de OPC soporta la especificación Acceso de Datos 2.0 de OPC.

Tabla 2.3 Descripción de las funciones privadas del objeto *WrapOPCServer*

Nombre	Descripción
GetOPCServerList	Este procedimiento provee una lista de *ProgID* (ver Apéndice A) presentes en una máquina local o remota.
ConnectOPCServer	Esta función maneja la conexión con un servidor de OPC.
DisconnectOPCServer	Esta función maneja la desconexión de un servidor de OPC.
GetOPCQueryAvailableLocaleIDs	Esta función obtiene los idiomas (*LocaleIDs*, ver Apéndice B) disponibles para esta sección servidor/cliente, este valor será usado por la función *GetOPCErrorString*.
GetOPCQueryAvailableProperties	Esta función retorna una lista de identificadores de las propiedades (*PropertyID*, ver Apéndice B) y descripciones para las propiedades que están disponibles para un ítem.
GetOPCItemProperties	Esta función retorna la lista de propiedades del ítem, ingresando como parámetro el identificador de la propiedad (*PropertyID*, ver Apéndice B), obtenida por la función *GetOPCQueryAvailableProperties*.
OPCLookupItemIDs	Esta función retorna una lista de identificadores del ítem (*ItemID*, ver Apéndice A), si son disponibles, por cada identificador de la propiedad (*PropertyID*, ver Apéndice B) ingresado.
GetOPCServerKey	Esta función retorna la clave del objeto *WrapOPCSerer*.
GetOPCServerGroupCollection	Esta función retorna una referencia hacia la colección *OPCServerGroupsColl* (colección de objetos *WrapOPCServer*).
GetOPCServerIndex	Esta función retorna el índice del objeto *WrapOPCServer*.
GetOPCServerName	Esta función retorna el nombre del servidor de OPC (*ProgID*, ver Apéndice A).
GetOPCNodeName	Esta función retorna el nombre del nodo de la maquina remota, donde se encuentra el servidor de OPC.
GetOPCStartTime	Esta función obtiene la propiedad *StartTime* (ver Apéndice B) del servidor de OPC.

Tabla 2.4 Descripción de las funciones públicas del objeto *WrapOPCServer*

(*continúa...*)

Nombre	Descripción
GetOPCCurrentTime	Obtiene la propiedad *CurrentTime* (ver Apéndice B) del servidor de OPC.
GetOPCLastUpdateTime	Obtiene la propiedad *LastUpdateTime* (ver Apéndice B) del servidor de OPC.
GetOPCMajorVersion	Obtiene la propiedad *MajorVersion* (ver Apéndice B) del servidor de OPC.
GetOPCMinorVersion	Obtiene la propiedad *MinorVersion* (ver Apéndice B) del servidor de OPC.
GetOPCBuildNumber	Obtiene la propiedad *Build* (ver Apéndice B) del servidor de OPC.
GetOPCLocaleID	Obtiene la propiedad *LocaleID* (ver Apéndice B) del servidor de OPC.
GetOPCErrorString	Esta función retorna una cadena con la descripción del error que es ingresado como un código numérico.
GetOPCPublicGroupNames	Esta función obtiene los nombres de los grupos públicos existentes en el servidor de OPC.
GetOPCVendorInfo	Obtiene la información del fabricante del servidor de OPC.
GetOPCServerState	Obtiene la propiedad *ServerState* (ver Apéndice B) del servidor de OPC.
SetOPCLocaleID	Establece la propiedad *LocaleID* (ver Apéndice B) del servidor de OPC.
GetOPCServerBrowseObject	Esta función retorna un objeto *OPCBrowser* del objeto *OPCServer* (ver Apéndice A), con el objetivo de verificar si el servidor soporta el objeto *OPCBrowser* y si es así, se crea un nuevo objeto *WrapOPCBrowser*.
SetOPCDefaultGroupIsActive	Esta función establece la condición inicial de la propiedad *DefaultGroupIsActive* (ver Apéndice B) del objeto *OPCGroups* de la Interface de Automatización. Esta función es utilizada dentro de las funciones *AddOPCGroup* y *ConnectOPCServer*.

Tabla 2.4 (*Continuación de la página 27.*) Descripción de las funciones públicas del objeto *WrapOPCServer* (*continúa...*)

Nombre	Descripción
SetOPCDefaultGroupUpdateRate	Esta función establece la propiedad *UpdateRate* (ver Apéndice B) por defecto del objeto *OPCGroups* de la Interface de Automatización.
SetOPCDefaultGroupDeadBand	Esta función establece la propiedad *Deadband* (ver Apéndice B) por defecto del objeto *OPCGroups* de la Interface de Automatización.
GetOPCDefaultGroupIsActive	Esta función obtiene la condición inicial de la propiedad *DefaultGroupIsActive* (ver Apéndice B) del objeto *OPCGroups* de la Interface de Automatización. Esta función es utilizada dentro de las funciones *AddOPCGroup* y *ConnectOPCServer*.
GetOPCDefaultGroupUpdateRate	Esta función obtiene la propiedad *UpdateRate* (ver Apéndice B) por defecto del objeto *OPCGroups* de la Interface de Automatización.
GetOPCDefaultGroupDeadBand	Esta función obtiene la propiedad *Deadband* (ver Apéndice B) por defecto del objeto *OPCGroups* de la Interface de Automatización.
AddOPCGroup	Esta función maneja la adición de un nuevo grupo al servidor de OPC.
RemoveOPCGroup	Esta función maneja la eliminación de un grupo perteneciente a un servidor de OPC.
AddOPCPublicGroup	Esta función se conecta con un grupo público (ver Apéndice A).

Tabla 2.4 (*Continuación de la página 28.*) Descripción de las funciones públicas del objeto *WrapOPCServer*

2.1.1.2 Definición del objeto *WrapOPCBrowser*

El objeto *WrapOPCBrowser* es definido en la figura 2.4, en esta gráfica se puede ver todas las variables privadas, funciones privadas y públicas del objeto.

Las descripciones en detalle de las variables privadas, funciones privadas y públicas del objeto *WrapOPCBrowser*, están definidas en las tablas 2.5, 2.6 y 2.7 respectivamente.

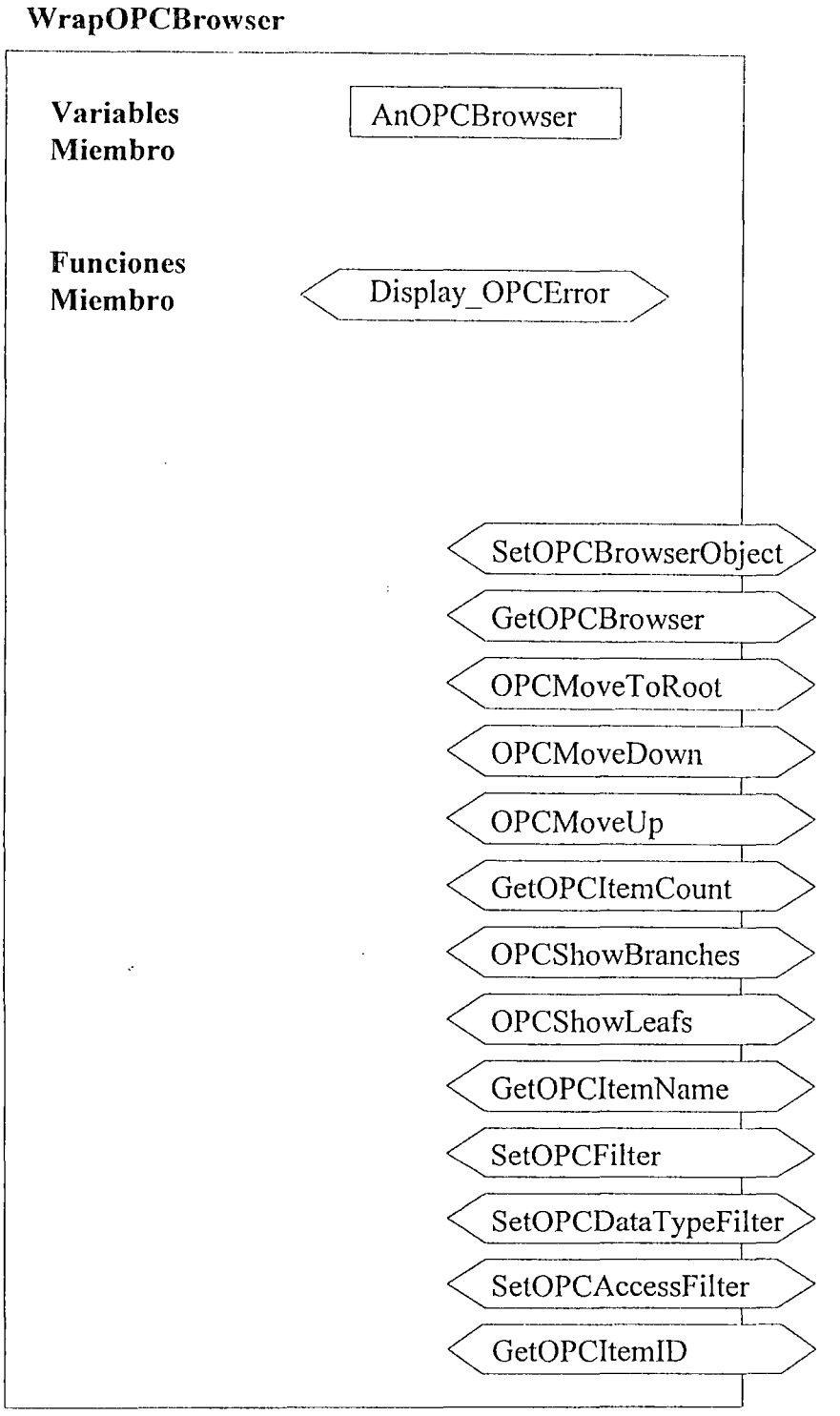


Figura 2.4 Diagrama del objeto *WrapOPCBrowser* (notación de Ege)

Nombre	Descripción
AnOPCBrowser	Objeto *OPCBrowser* (ver Apéndice A).

Tabla 2.5 Descripción de la variable privada del objeto *WrapOPCBrowser*

Nombre	Descripción
Display_OPCError	Este procedimiento muestra cualquier error generado por OPC, COM o Visual Basic, que es encontrado por el *manejador de excepciones* de cada función del objeto *WrapOPCBrowser* (ver manejo del error de la especificación "Data Access Automation Interface Standard" versión 2.02).

Tabla 2.6 Descripción de la función privada del objeto *WrapOPCBrowser*

Nombre	Descripción
GetOPCBrowserOrganization	Esta función permite establecer el formato del *namespace* disponible en el servidor de OPC (Apéndice B, tabla B 16).
OPCMoveToRoot	Esta función mueve el buscador hasta la posición inicial o raiz del *namespace*.
OPCMoveDown	Esta función mueve la posición del buscador del *namespace* hacia abajo de una rama.
OPCMoveUp	Esta función mueve el buscador del *namespace* un nivel de la posición actual. Se usa esta función en conjunción con la función *OPCMoveDown*, para poder navegar dentro del *namespace*.
GetOPCItemCount	Esta función retorna dos valores dependiendo de la función de llamada que le precede. Si la función *OPCShowBranches* es llamada previamente, el valor retornado será el número de ramas disponibles debajo de la posición actual, dentro del *namespace* del servidor de OPC. Si la función *OPCShowLeafs* es llamada previamente, el valor retornado será el número de hojas, items o *tags* disponibles debajo de la posición actual del *namespace*.
OPCShowBranches	Esta función ocasionará que el servidor obtenga la colección de ramas disponibles debajo de la posición actual del *namespace*. Se puede determinar luego cuantas ramas existen llamando la función *GetOPCItemCount*. Si hay ramas disponibles ellos puede ser recuperados al usar la función *GetOPCItemName*.
OPCShowLeafs	Esta función ocasionará que el servidor de OPC obtenga la colección de hojas, items o *tags* disponibles debajo de la posición actual del *namespace*. Luego se puede determinar cuantas son llamando a la función *GetOPCItemCount*. Si existen algunos disponibles, se pueden recuperar al usar la función *GetOPCItemName*.
GetOPCItemName	Esta función retorna el nombre del ítem de la colección. Esta colección es cargada al llamar cualquiera de las funciones *OPCShowBranches* o *OPCShowLeafs*.

Tabla 2.7 Descripción de las funciones públicas del objeto *WrapOPCBrowser*

(*continúa...*)

Nombre	Descripción
SetOPCFilter	Esta función permite aplicar un filtro a las hojas o ramas obtenidas al llamar a la función *OPCShowLeafs* o *OPCShowBranches*, respectivamente. Este filtro está basado en la propiedad *ItemID* (ver Apéndice B) del ítem.
SetOPCDataTypeFilter	Esta función permite aplicar un filtro a los items llamados por la función *OPCShowLeafs*. Este filtro está basado en la propiedad *DataType* (ver Apéndice B) de ítem.
SetOPCAccessFilter	Esta función permite filtrar el número de items que retornan al llamar la función *OPCShowLeafs*, este filtro es basado en la propiedad *AccessRights* (ver Apéndice B) del ítem.
GetOPCItemID	Esta función retorna una cadena con el identificador del ítem *ItemID* (ver Apéndice B).
SetOPCBrowserObject	Esta función simplemente ingresa un objeto *OPCBrowser* que será usado por un instante por el objeto *WrapOPCBrowser*. Esto es llevado a cabo al usar la función *GetOPCServerBrowseObject* de la clase *WrapOPCServer*.

Tabla 2.7 (*Continuación de la página 33.*) Descripción de las funciones públicas del objeto *WrapOPCBrowser*

2.1.1.3 Definición del objeto *WrapOPCGroup*

En la figura 2.5 se define el objeto *WrapOPCGroup*, donde se puede ver claramente todas sus variables privadas, funciones privadas y públicas.

Todas las características de cada elemento del objeto *WrapOPCGroup* son definidas en las tablas 2.8, 2.9 y 2.10, donde las variables privadas, funciones privadas y públicas respectivamente, son descritas a detalle.

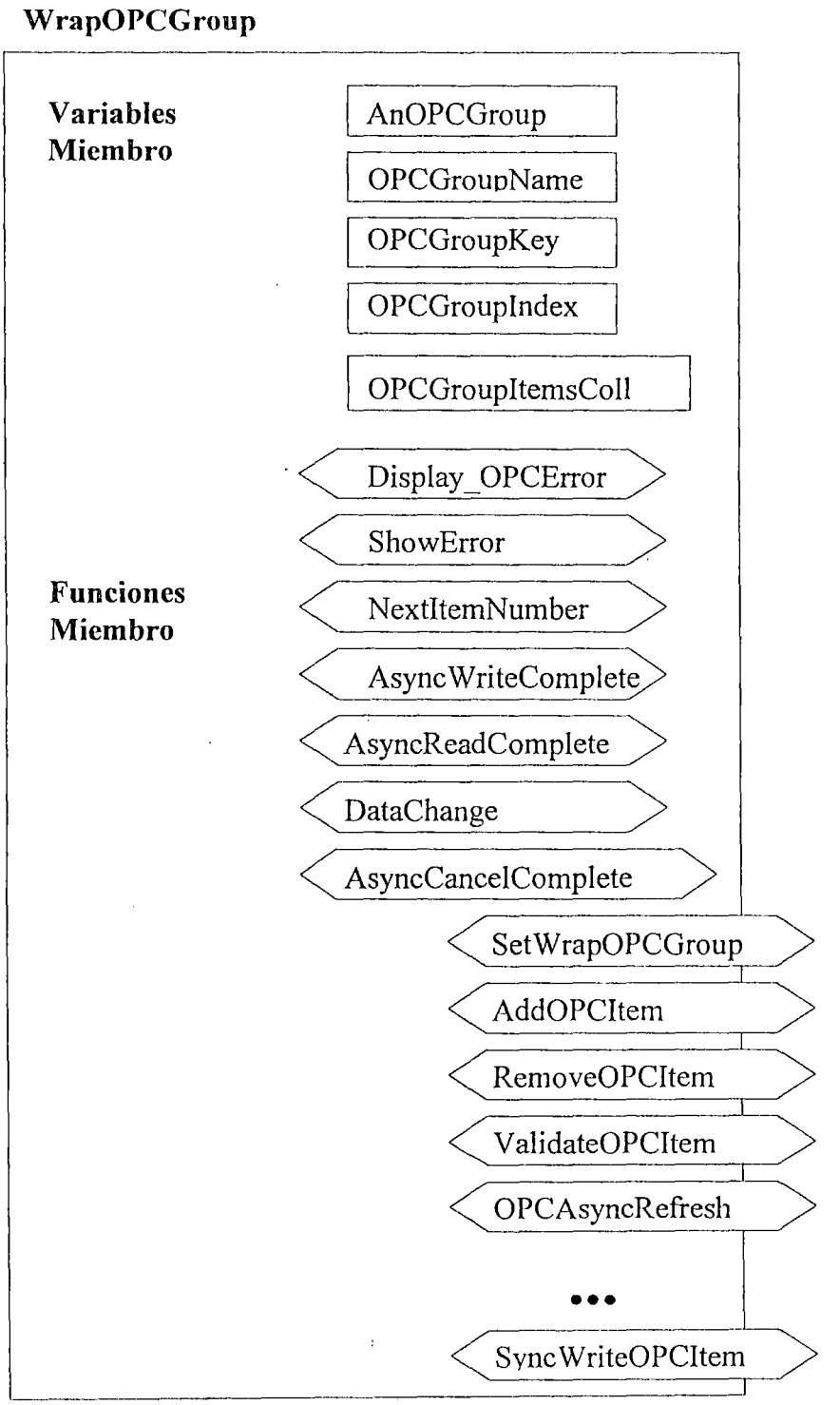


Figura 2.5 Diagrama del objeto *WrapOPCGroup* (notación de Ege)

Nombre	Descripción
OPCGroupName	Contiene el nombre actual del grupo.
OPCGroupKey	Este es una clave usado para referenciar a un objeto *WrapOPCGroup*. También es usado como clave para el control *TreeView* (ver sección 2.2). Con esta clave y la función *GetOPCServerGroupCollection* del objeto *WrapOPCServer* se puede obtener directamente el acceso en un específico instante al objeto *WrapOPCGroup*. Esta clave es generada por la función *AddOPCGroup* del objeto *WrapOPCServer*.
AnOPCGroup	Objeto *OPCGroup* (ver Apéndice A) es declarado usando *WithEvents* para trabajar con los eventos del objeto de la Interface de Automatización.
OPCGroupItemsColl	Nueva colección para almacenar los objetos *WrapOPCItem* que son añadidos a este objeto *WrapOPCGroup*.
OPCGroupIndex	Contiene el índice del objeto *WrapOPCGroup*. Además, es usado para crear la clave única *OPCGroupKey*.

Tabla 2.8 Descripción de las variables privadas del objeto *WrapOPCGroup*

Nombre	Descripción
Display_OPCError	Este procedimiento muestra cualquier error generado por OPC, COM o Visual Basic, que es encontrado por el *manejador de excepciones* de cada función (ver manejo del error de la especificación "Data Access Automation Interface Standard" versión 2.02).
ShowError	Esta función devuelve una expresión del error generado por OPC, COM o Visual Basic, que es encontrado por el *manejador de excepciones* de cada función (ver manejo del error de la especificación "Data Access Automation Interface Standard" versión 2.02). Este error se mostrará en el control *ListView "lvListViewH"* (ver sección 2.2).
NextItemNumber	Esta función crea un índice único cada vez que se llama a la función *AddOPCItem.*
DataChange	Este evento es producido cuando el valor o la propiedad *Quality* (ver Apéndice B) de un ítem dentro de un grupo ha cambiado. Este evento no trabajará más rápido que el parámetro *UpdateRate* (ver Apéndice B) del grupo. Solamente items que son activos, y cuyo grupo es activo se notificarán por este evento.
AsyncReadComplete	Este evento es generado cuando una lectura asíncrona ha terminado (la lectura asíncrona es llevado a cabo al usar el procedimiento *AsyncReadOPCItem*).
AsyncWriteComplete	Este evento actúa cuando una escritura asíncrona ha terminado (la escritura asíncrona es llevado a cabo al usar el procedimiento *OPCAsyncWrite*).
AsyncCancelComplete	Este evento actúa cuando una cancelación asíncrona ha terminado (la cancelación asíncrona es llevado a cabo al usar el procedimiento *OPCAsyncCancel*).

Tabla 2.9 Descripción de las funciones privadas del objeto *WrapOPCGroup*

Nombre	Descripción
SetWrapOPCGroup	Este procedimiento inicializa el objeto *WrapOPCGroup*. Este es usado por la función *AddOPCGroup* de la clase *WrapOPCServer*.
GetOPCGroupName	Esta función obtiene el nombre del objeto *OPCGroup* (ver Apéndice A) seleccionado.
GetOPCGroupKey	Esta función simplemente retorna la clave del objeto *WrapOPCGroup*.
GetOPCGroupIndex	Esta función simplemente retorna el índice del objeto *WrapOPCGroup*.
GetOPCGroupItemsCollection	Esta función simplemente retorna la colección de objetos *WrapOPCItem*.
SetOPCGroupName	Esta función establece el nombre del objeto *OPCGroup* (ver Apéndice A) seleccionado.
SetOPCGroupActiveState	Esta función permiten establecer la propiedad *IsActive* (ver Apéndice B) del objeto *OPCGroup*. Cuando el grupo está desactivado no podrá recibir las notificaciones del servidor de OPC.
GetOPCGroupActiveState	Esta función permite obtener la propiedad *IsActive* (ver Apéndice B) del objeto *OPCGroup*.
SetOPCGroupDeadBand	Esta función permiten establecer la propiedad *DeadBand* (ver Apéndice B) del objeto *OPCGroup*. Esta propiedad permite limitar el porcentaje de variación del valor del ítem que retornará el servidor de OPC.
GetOPCGroupDeadBand	Esta función permiten obtener la propiedad *DeadBand* (ver Apéndice B) del objeto *OPCGroup*.
SetOPCGroupUpdateRate	Esta función permiten establecer la propiedad *UpdateRate* (ver Apéndice B) del objeto *OPCGroup*. La propiedad *UpdateRate* puede ser usada para controlar y mejorar el rendimiento de la comunicación, el menor valor para esta propiedad es 0, el cual permite que el servidor de OPC notifica los cambios de los datos lo más rápido posible.
GetOPCGroupUpdateRate	Esta función permiten obtener la propiedad *UpdateRate* (ver Apéndice B) del objeto *OPCGroup*.
OPCAsyncRefresh	Este procedimiento genera un evento para actualizar los valores de todos los items activos dentro de un grupo. Los resultados son retornados vía el evento *DataChange* asociado al objeto *OPCGroup*.

Tabla 2.10 Descripción de las funciones públicas del objeto *WrapOPCGroup*

(*continúa...*)

Nombre	Descripción
AsyncReadOPCItem	Este procedimiento lee en forma asíncrona uno o más items de un grupo. Los resultados son retornados vía el evento *AsyncReadComplete* del objeto *OPCGroup*.
OPCAsyncWrite	Realiza la escritura asíncrona de uno o más items dentro de un grupo. Los resultados son retornados vía el evento *AsyncWriteComplete* del objeto *OPCGroup*.
SyncReadOPCItem	Este procedimiento lee en forma síncrona el valor, la calidad y el tiempo de actualización (*Value, Quality y Timestamp*) de uno o más items de un grupo.
SyncWriteOPCItem	Realiza la escritura en forma síncrona de uno o más items de un grupo. Los valores son escritos en el dispositivo. El procedimiento no retornará hasta que verifique que el dispositivo ha aceptado o rechazado el dato.
SetOPCItemsDefaultDataType	Esta función permite establecer la propiedad *DataType* (ver Apéndice B) por defecto para la adición de items hacia un grupo.
SetOPCItemsDefaultActive	Esta función permite obtener la propiedad *DataType* (ver Apéndice B) por defecto para la adición de items hacia un grupo.
AddOPCItem	Esta función se encarga de la adición de un ítem al grupo. Cuando se adiciona un ítem, se configuran algunos de los parámetros del mismo usando las funciones *SetOPCItemsDefaultActive* y *SetOPCItemsDefaultDataType*.
RemoveOPCItem	Esta función maneja la eliminación de un ítem del grupo. El ítem primeramente es eliminado del objeto *OPCItems*, luego el ítem es removido de la colección *OPCGroupItemsColl*, el cual es manejado por la clase *WrapOPCGroup*.
ValidateOPCItem	Con esta función se puede probar los parámetros de un ítem antes de ser adicionado a un grupo, estos parámetros son: *Item ID, DataType*, y *ActiveState*. Si todos estos parámetros son adecuados el servidor de OPC no retornará errores.

Tabla 2.10 (*Continuación de la página 39.*) Descripción de las funciones públicas del objeto *WrapOPCGroup*

2.1.1.4 Definición del objeto *WrapOPCItem*

El objeto *WrapOPCItem* es definido en la figura 2.6, donde se puede ver claramente todas sus variables privadas, funciones privadas y públicas.

Todas las características de cada elemento del objeto son definidas en las tablas 2.11, 2.12 y 2.13 donde las variables privadas, funciones privadas y públicas respectivamente, son descritas a detalle.

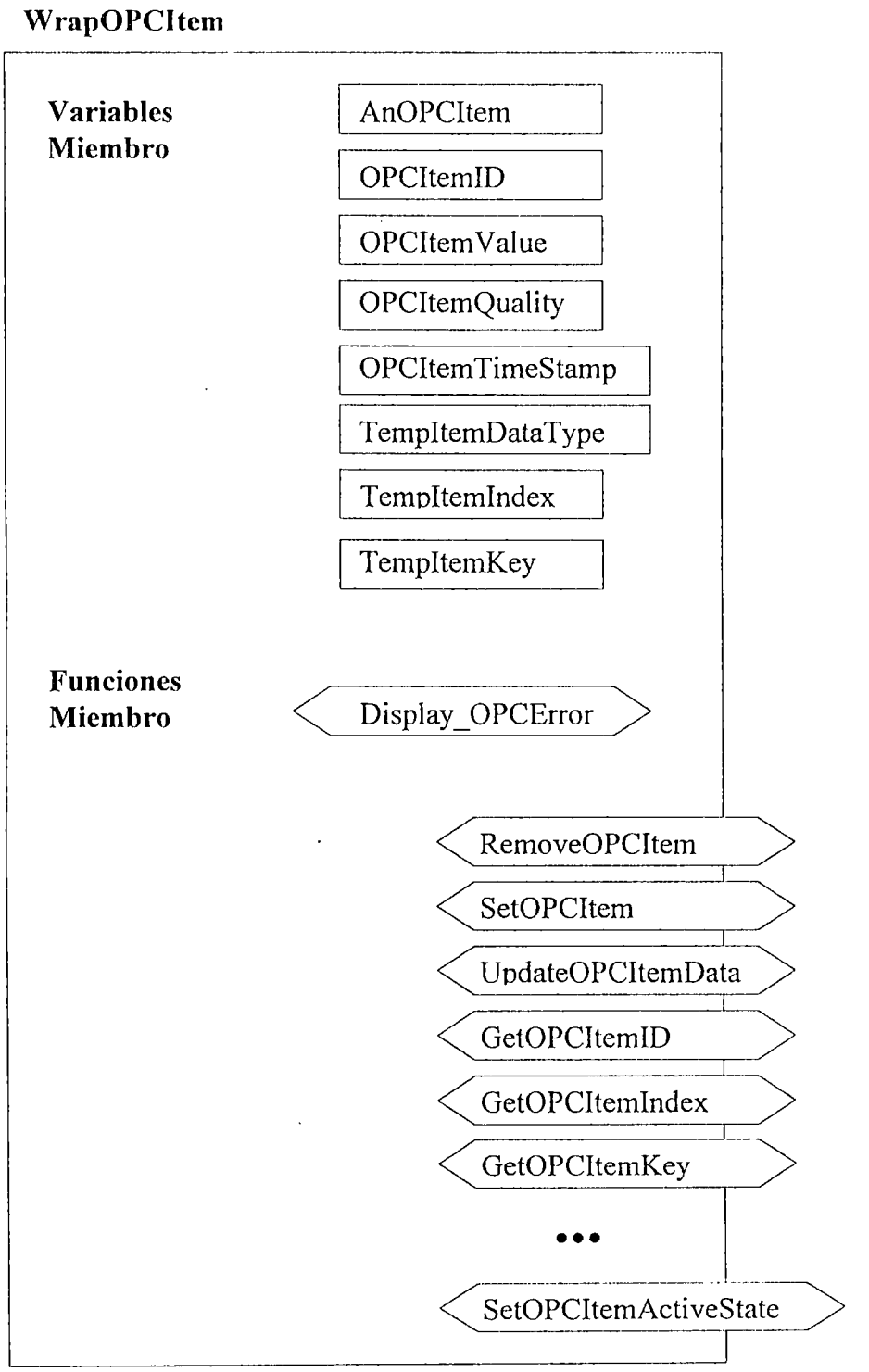


Figura 2.6 Diagrama del objeto *WrapOPCItem* (notación de Ege)

Nombre	Descripción
AnOPCItem	Objeto *OPCItem* (ver Apéndice A), el cual es retornado cuando el ítem es adicionado al grupo, al usar la función *AddOPCItem* que perteneciente a la clase *WrapOPCGroup*.
OPCItemID	Guarda el identificador del ítem (*ItemID*, ver Apéndice A).
OPCItemValue	Guarda el valor del ítem, el cual es una variable tipo *Variant* (este tipo de variable es usado para mantener cualquier valor del dato que es retornado por el servidor de OPC).
OPCItemQuality	Guarda la propiedad *Quality* (ver Apéndice B) del ítem, un valor de &HCO (192) indica que el valor del ítem es correcto y no tiene errores.
OPCItemTimeStamp	Guarda la propiedad *TimeStamp* (ver Apéndice B) del ítem, el cual contiene el tiempo de la ultima operación de lectura para este ítem.
TempItemDataType	Esta variable guarda temporalmente la propiedad *DataType* (ver Apéndice B) del ítem.
TempItemIndex	Esta variable guarda temporalmente el índice del ítem.
TempItemKey	Esta variable guarda temporalmente la clave del objeto *WrapOPCItem*.

Tabla 2.11 Descripción de las variables privadas del objeto *WrapOPCItem*

Nombre	Descripción
Display_OPCError	Este procedimiento muestra cualquier error generado por OPC, COM o Visual Basic, que es encontrado por el *manejador de excepciones* de cada función (ver manejo del error de la especificación "Data Access Automation Interface Standard" versión 2.02).

Tabla 2.12 Descripción de las funciones privadas del objeto *WrapOPCGroup*

Nombre	Descripción
GetOPCItemID	Esta función retorna una cadena con el identificador del ítem *ItemID* (ver Apéndice B).
GetOPCItemValue	Esta función retorna el valor del objeto *OPCItem*.
GetOPCItemQuality	La función retorna el parámetro *Quality* (ver Apéndice B) del objeto *OPCItem*.
GetOPCItemTimeStamp	Esta función retorna el parámetro *TtimeStamp* (ver Apéndice B) del objeto *OPCItem*.
GetOPCItemDataType	Esta función retorna el parámetro *DataType* (ver Apéndice B) del objeto *OPCItem*.
GetOPCItemCanonicalType	Esta función retorna el parámetro *CanonicalDataType* (ver Apéndice B) del objeto *OPCItem*.
GetOPCItemServerHandle	Esta función retorna el parámetro *ServerHandle* (ver Apéndice B) para el objeto *OPCItem*.
GetOPCItemActiveState	Esta función retorna el parámetro *IsActive* (ver Apéndice B) del objeto *OPCItem*.
SetOPCItemActiveState	Este procedimiento permite activar o desactivar un especifico ítem usando la propiedad *IsActive* (ver Apéndice B) del objeto *OPCItem*.
SetOPCItem	Este procedimiento es llamado por la función *AddOPCItem* del objeto *WrapOPCGroup*, para iniciar el objeto *WrapOPCItem* y establecer los siguientes parámetros del ítem: *ItemID, Index y DataType* (ver Apéndice B).
UpdateOPCItemData	Este procedimiento es llamado por el evento *DataChange* del objeto *WrapOPCGroup*. Este procedimiento simplemente actualiza las variables *OPCItemValue*, *OPCItemQuality* y *OPCItemTimeStamp* (ver tabla 2.11) correspondientes a las propiedades *Value, Quality* y *DataType* (ver Apéndice B) del objeto *OPCItem*.
GetOPCItemIndex	Esta función retorna el índice del objeto *WrapOPCItem*.
GetOPCItemKey	Esta función retorna la clave del objeto *WrapOPCItem* que podría ser de la forma "Item 1 2 3". Donde: "1", es el índice del objeto *WrapOPCItem*. "2" el índice del objeto *WrapOPCGroup* y "3" el índice del objeto *WrapOPCServer*. Esta clave es usado por el control *TreeView* de la interfaz de usuario (ver sección 2.2).

Tabla 2.13 Descripción de las funciones públicas del objeto *WrapOPCItem*

2.1.2 Funcionalidad para la Conexión y Desconexión con un Servidor de OPC

Esta funcionalidad fue llevada a cabo siguiendo los siguientes pasos:

2.1.2.1 Descomposición Orientada a Objetos de la Funcionalidad

Las figuras 2.7 y 2.8 muestran respectivamente, el comportamiento dinámico (secuencia de sucesos) de la conexión y desconexión con un servidor de OPC.

En figura 2.7, el objeto *AWrapSr* es el encargado de llevar a cabo los sucesos. A diferencia del esquema de la figura 2.8, donde el objeto encargado es *SelectedOPCServer,* el cual representa el objeto seleccionado actualmente desde la interfaz de usuario. Las funciones de las secuencias 3 al 5 del objeto *AWrapSr* de la figura 2.7, llaman respectivamente a los métodos de las secuencias 3 al 5 del objeto *OPCGroups*. Las funciones de las secuencias 1 al 5 de los objetos *OPCServer* y *OPCGroups* (objetos de la Interface de Automatización) de la figura 2.7 y la función de la secuencia 1 de la figura 2.8, son explicadas en el Apéndice B.

La secuencia 6 de la figura 2.7 y secuencia 2 de la figura 2.8 permiten respectivamente, adicionar a la colección y eliminar de la colección el objeto actual (*AWrapSr* y *SelectedOPCServer,* respectivamente).

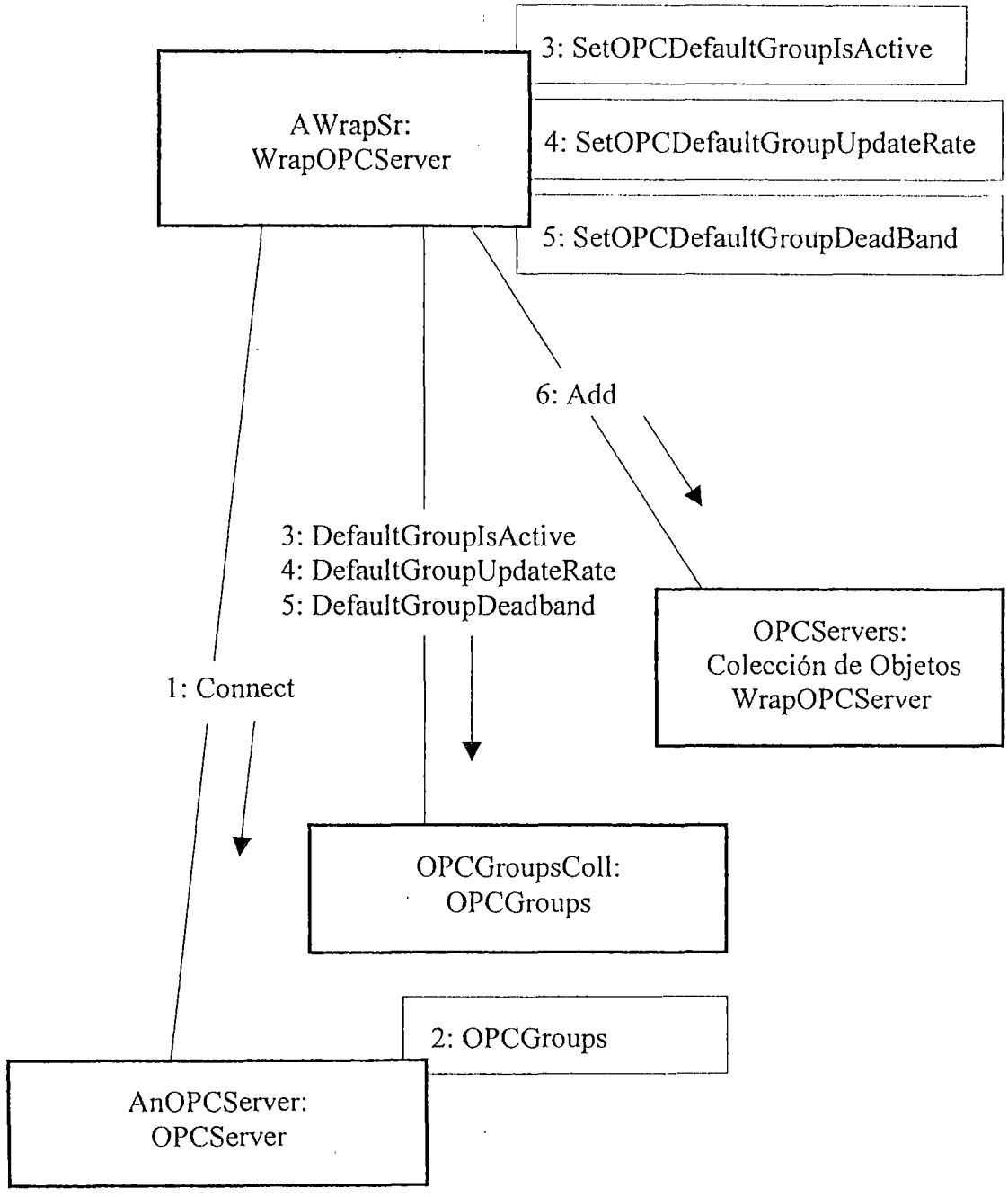


Figura 2.7 Descomposición orientada a objetos para la conexión con un servidor de OPC (objeto *OPCServer*)

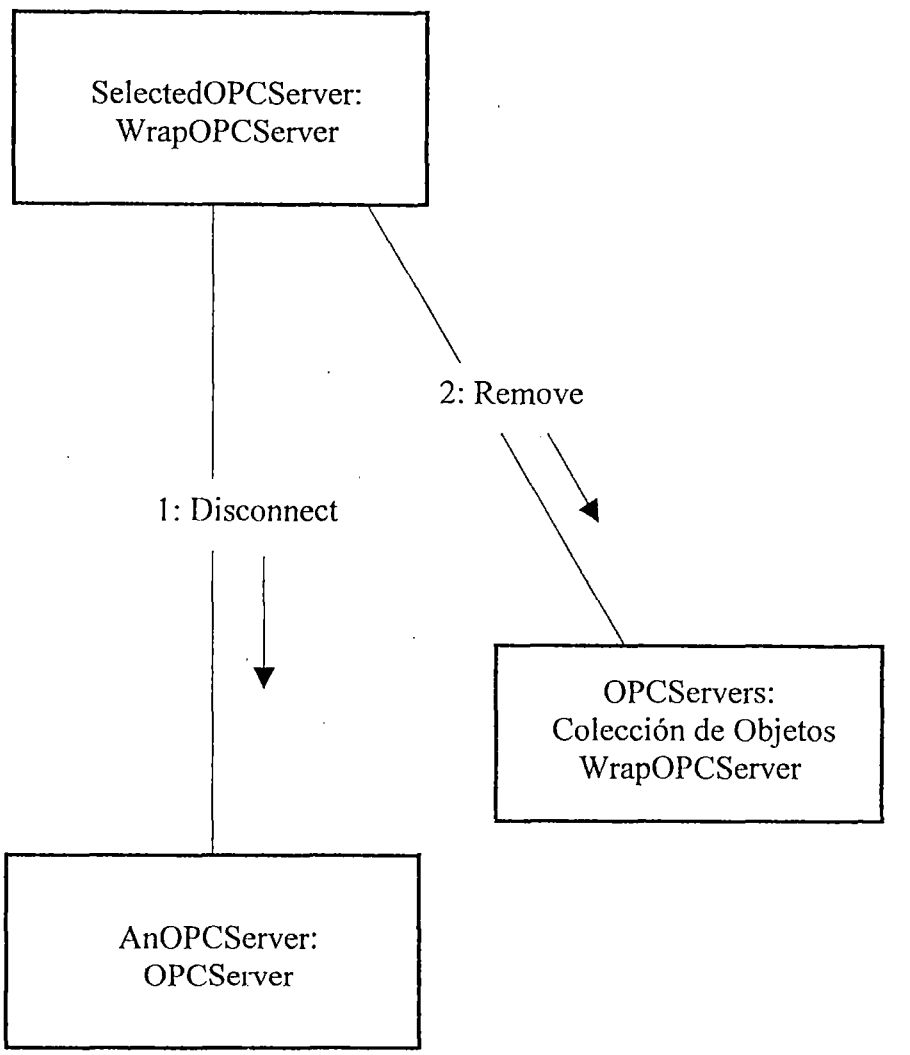


Figura 2.8 Descomposición orientada a objetos para la desconexión de un servidor de OPC (objeto *OPCServer*)

2.1.2.2 Diseño del Algoritmo

Una vez definido el comportamiento dinámico de la funcionalidad, se crea el algoritmo de la función encargada de la conexión (*ConnectOPCServer*, ver tabla 2.4). Este algoritmo es mostrado en la figura 2.9, donde se muestra la secuencia de sucesos de la figura 2.7 desde el 1 al 5.

El algoritmo de la figura 2.10 llama a la función del algoritmo de la figura 2.9 para crear un objeto *OPCServer*, pero además provee los requerimientos necesarios para la interacción con la interfaz de usuario. Este algoritmo lleva a cabo el suceso 6 de la figura 2.7. El algoritmo para la desconexión con un servidor de OPC no fue necesario, ya que la implementación es directa. La implementación se basa en la descomposición de la figura 2.8.

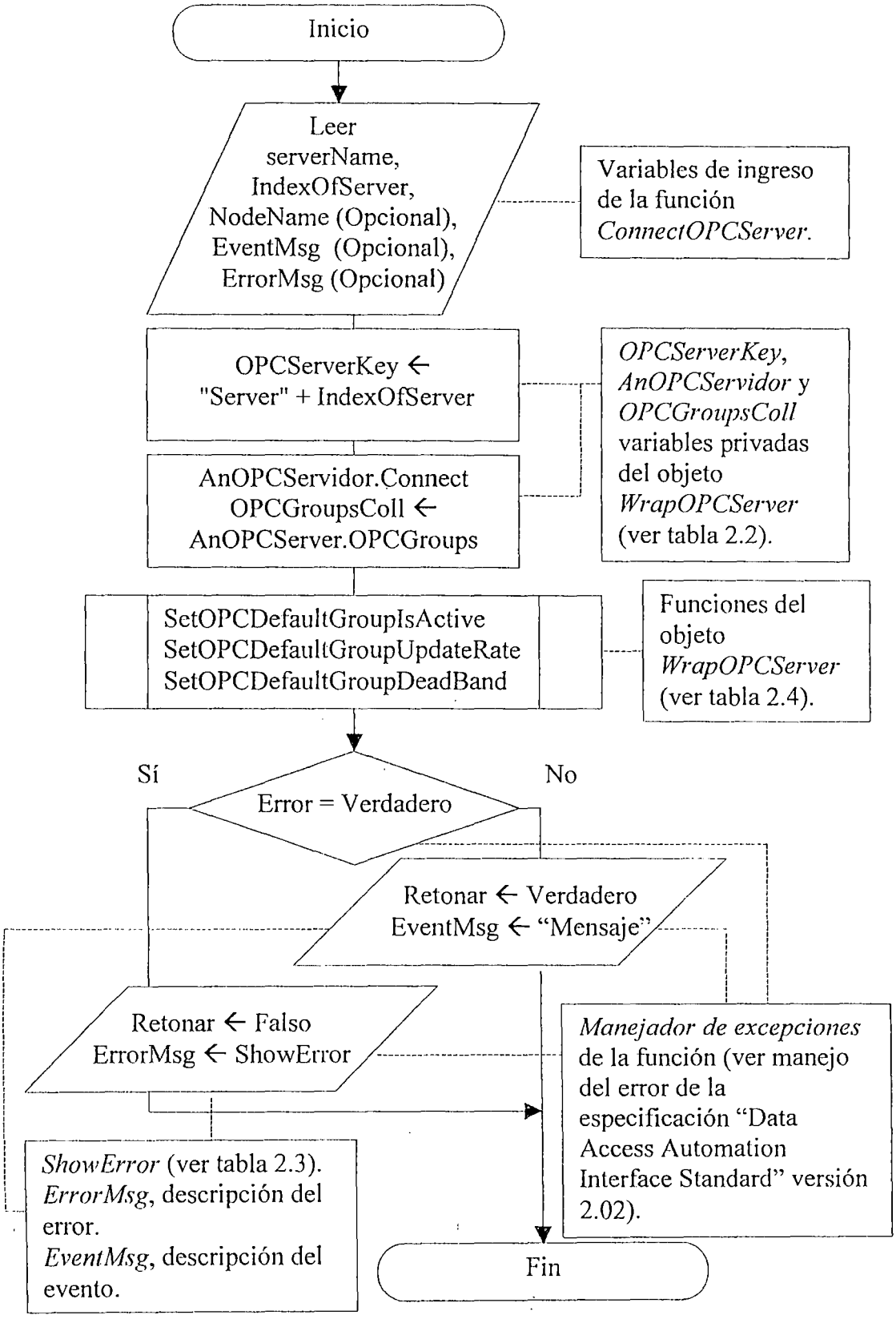


Figura 2.9 Algoritmo de la función *ConnectOPCServer* del objeto *WrapOPCServer*

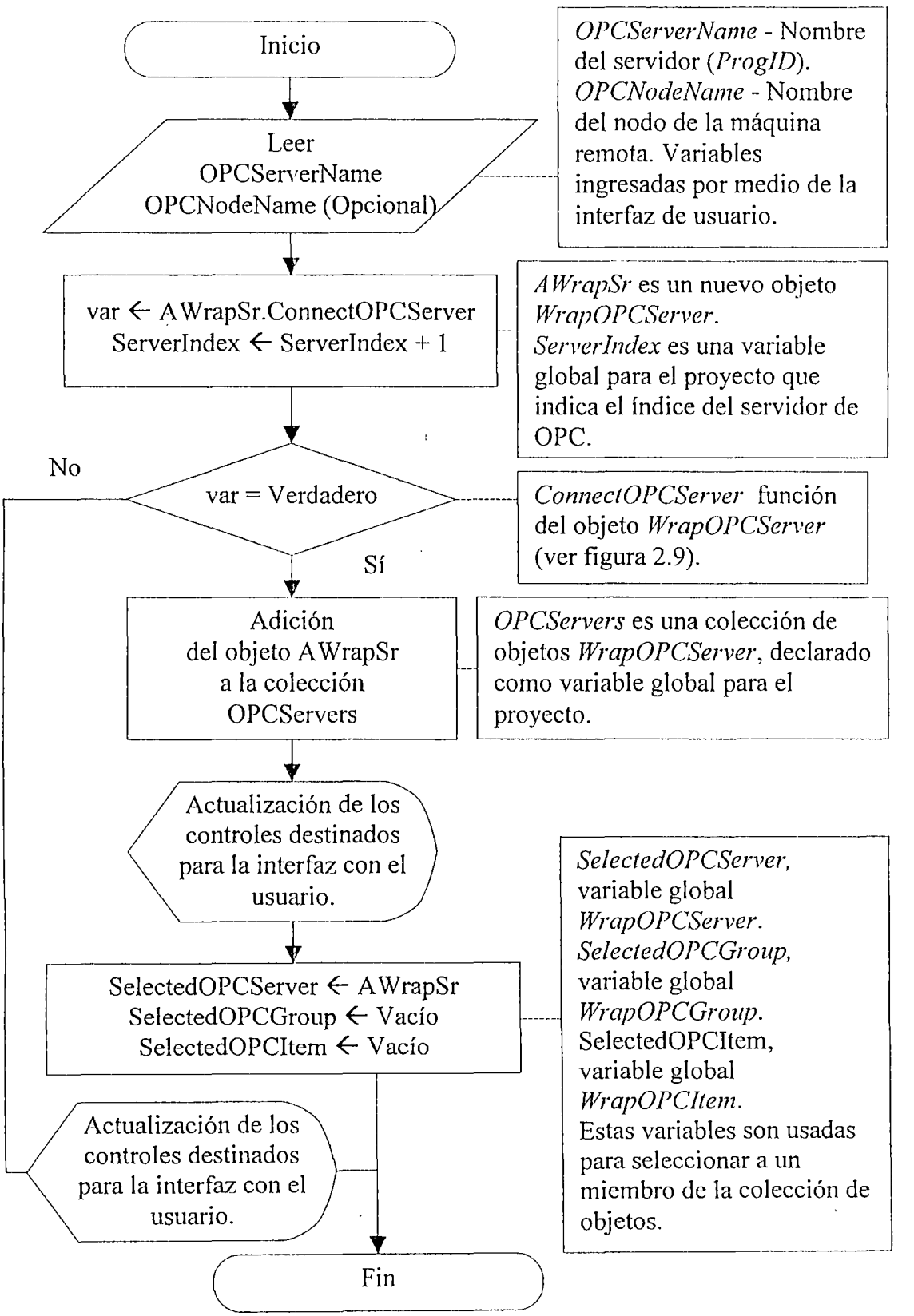


Figura 2.10 Algoritmo para la conexión con un servidor de OPC (objeto *OPCServer*) desde la interfaz de usuario

2.1.2.3 Implementación

Las listas 2.1 y 2.2 muestran los códigos desarrollados para los algoritmos de las figuras 2.9 y 2.10 respectivamente

Las líneas 2 y 18 al 22 de la lista 2.1, muestran el procedimiento para el manejador del error producido durante la ejecución del programa (ver manejo del error de la especificación "Data Access Automation Interface Standard" versión 2.02). En la línea 10 se puede apreciar la creación del la clave única para el servidor de OPC creado (objeto *OPCServer*). Las funciones de las líneas 13 al 15 pertenecen al objeto *WrapOPCServer*.

En la lista 2.2 la variable *ModMain* viene a ser un módulo principal donde se declaran todas las variables globales para todo el proyecto. Las variable como *lvListViewH* (control *ListView*), *lvTreeView* (control *TreeView*) vienen a ser controles del Visual Basic destinados para la interfaz de usuario (ver sección 2.2).

```
1.  Function ConnectOPCServer(serverName As String,
                    IndexOfServer As Integer,
                    Optional ByVal NodeName As Variant,
                    Optional ByRef EventMsg As String,
                    Optional ByRef ErrorMsg As String) As Boolean

    ' Manejo del error
2.  On Error GoTo ShowConnectOPCServerError

3.      Dim StoreName As String
4.      Dim StoreNodeName As Variant

5.      StoreName = serverName
6.      OPCServerName = StoreName
7.      OPCServerIndex = IndexOfServer
8.      StoreNodeName = NodeName
9.      OPCNodeName = StoreNodeName


10.     OPCServerKey = "Server" + Str(IndexOfServer)

11.     AnOPCServer.Connect OPCServerName, OPCNodeName

12.     Set OPCGroupsColl = AnOPCServer.OPCGroups

13.     SetOPCDefaultGroupIsActive (True)
14.     SetOPCDefaultGroupUpdateRate (100)
15.     SetOPCDefaultGroupDeadBand (0)

16.     ConnectOPCServer = True

17.     EventMsg = "Connected to Server '" + OPCServerName + "'"

18. GoTo SkipConnectOPCServerError

19. ShowConnectOPCServerError:
20.     ConnectOPCServer = False
21.     ErrorMsg = ShowError("Connect", Err.Number)
22. SkipConnectOPCServerError:

23. End Function
```

Lista 2.1 Implementación de la función *ConnectOPCServer* del objeto *WrapOPCServer*

```
1.  Sub AddSelectedOPCServerMain(OPCServerName As String,
        Optional OPCNodeName As Variant)

2.      Dim AWrapSr As WrapOPCServer
3.      Set AWrapSr = New WrapOPCServer

4.      Dim Result As Boolean
5.      Dim EventMsg As String
6.      Dim ErrorMsg As String
7.      Dim itmx As ListItem
8.      Dim SrvName As String

9.      Result = AWrapSr.ConnectOPCServer(OPCServerName,
        ModMain.ServerIndex, OPCNodeName, EventMsg, ErrorMsg)
10.     ModMain.ServerIndex = ModMain.ServerIndex + 1

11.     If (Result = True) Then
12.         With OPCServers
13.             .Add AWrapSr, AWrapSr.GetOPCServerKey
14.         End With
15.         Dim nodX As Node
16.         Set nodX = fMainForm.tvTreeView.Nodes.Add(, ,
                AWrapSr.GetOPCServerKey, OPCServerName, 4, 5)
17.             nodX.EnsureVisible
18.         Set ModMain.SelectedOPCServer = AWrapSr
19.         Set ModMain.SelectedOPCGroup = Nothing
20.         Set ModMain.SelectedOPCItem = Nothing
21.         lvListView.ListItems.Clear
22.         Set itmx = lvListViewH.ListItems.Add(, , Date, 21, 21)
23.             itmx.SubItems(1) = Time ' Visualiza el tiempo
24.             itmx.SubItems(2) = EventMsg ' Muestra el error
25.             itmx.EnsureVisible
26.     End If

27.     If (Result = False) Then
28.         Set itmx = lvListViewH.ListItems.Add(, , Date, 13, 13)
29.         itmx.SubItems(1) = Time ' Visualiza el tiempo
30.         itmx.SubItems(2) = ErrorMsg ' Muestra el error
31.         itmx.EnsureVisible
32.     End If

33. End Sub
```

Lista 2.2 Implementación del algoritmo de la figura 2.10

2.1.3 Funcionalidad para la Adición y Eliminación de Grupos de OPC

La secuencia para la implementación de esta funcionalidad fue llevada a cabo de la siguiente manera:

2.1.3.1 Descomposición Orientada a Objetos de la Funcionalidad

La figura 2.11 muestra el comportamiento dinámico que se tendrá cada vez que se desea agregar un nuevo grupo al servidor de OPC seleccionado. La secuencia de la figura 2.11 se inicia al seleccionar un servidor de OPC por medio de la interfaz de usuario, luego el objeto *SelectedOPCServer* llama a sus funciones (secuencias 1 al 3), que a su vez llaman respectivamente a los métodos de las secuencias 1 al 3 del objeto *OPCGroups* de la Interface de Automatización. La secuencia restante se muestra muy claramente en la figura 2.11.

Para la eliminación de grupos se usa la secuencia de la figura 2.12, donde se muestra claramente el procedimiento que se debe seguir. Es importante notar que al obtener la clave de un grupo por medio de la función *GetOPCGroupKey* se puede hacer referencia al objeto *OPCGroup* que se desea eliminar. Sobre la base de la figura 2.12 se puede implementar directamente esta funcionalidad.

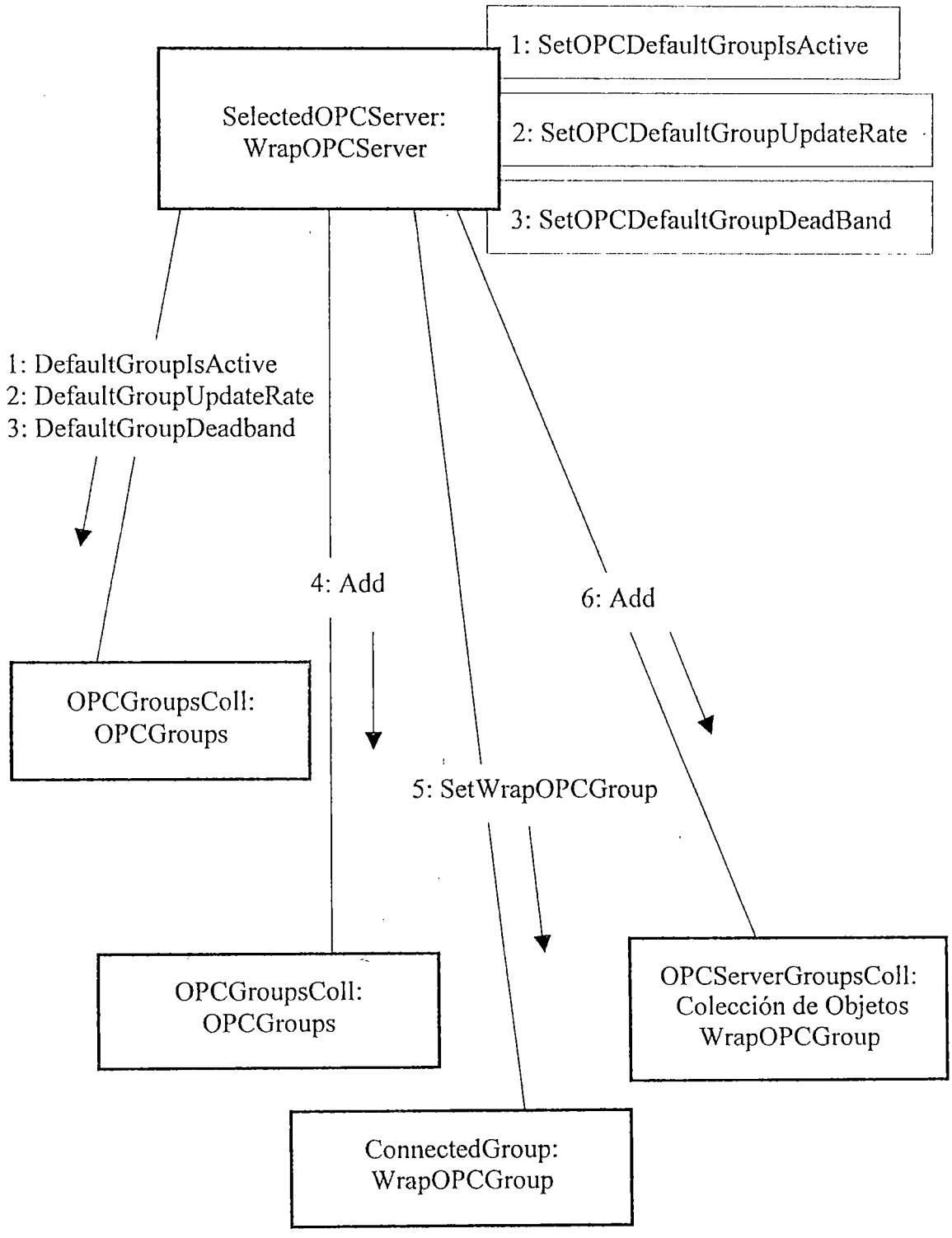


Figura 2.11 Descomposición orientada a objetos para la creación de un nuevo grupo

(objeto *OPCGroup*)

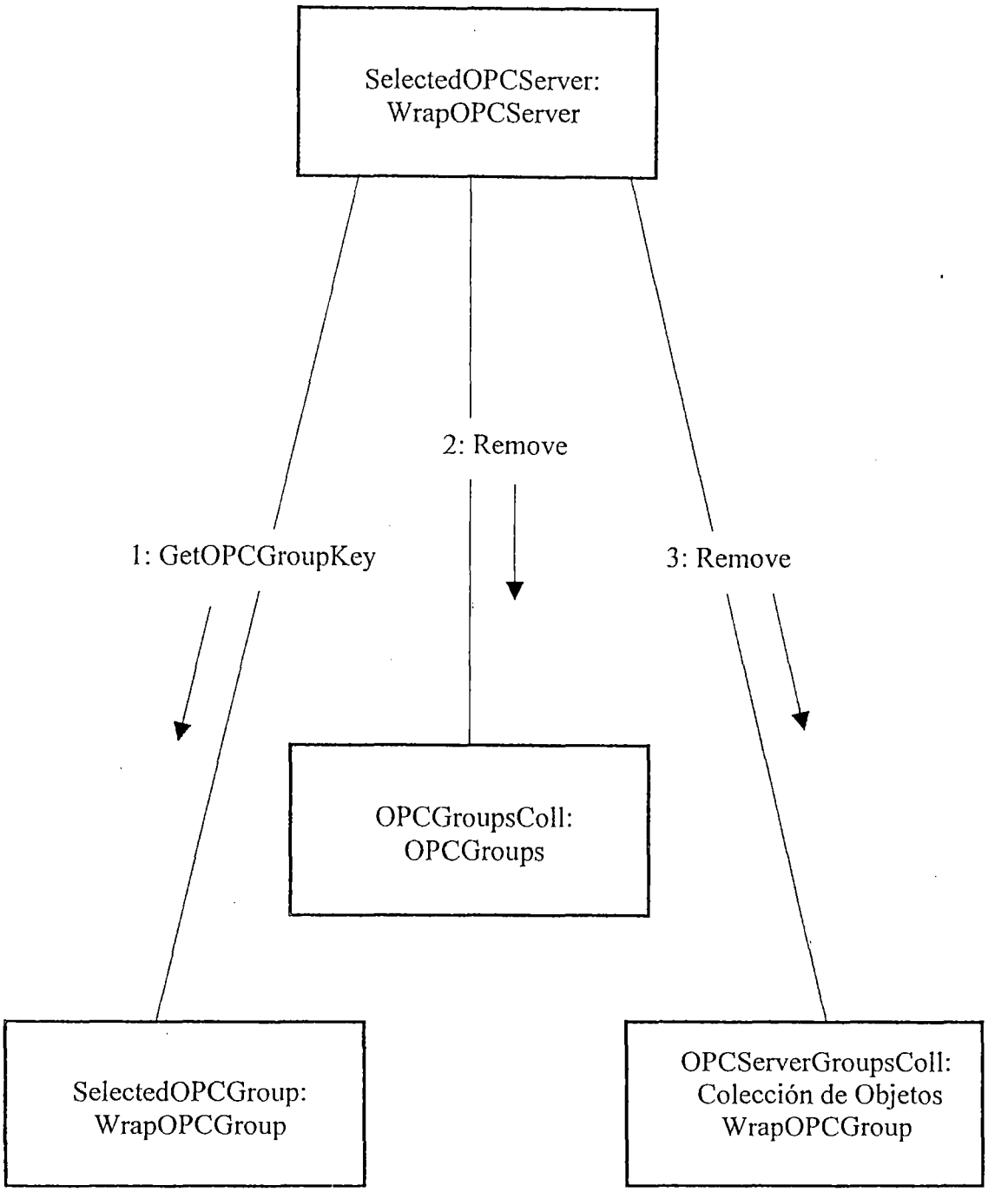


Figura 2.12 Descomposición orientada a objetos para la eliminación de un grupo

(objeto *OPCGroup*)

2.1.3.2 Diseño del Algoritmo

Sobre la base del esquema de la figura 2.11, se crea el algoritmo de la función *AddOPCGroup* (ver tabla 2.4) perteneciente al objeto *WrapOPCServer*. Este algoritmo, mostrado en la figura 2.13, es encargado de la creación de un nuevo objeto *OPCGroup*. Toda la secuencia de sucesos definidos en la figura 2.11 fue establecida en el algoritmo de la figura 2.13.

La primera y segunda condición del algoritmo de la figura 2.13 permiten crear un nombre para el grupo, cuando ni el usuario ni el servidor de OPC lo hicieron. Es importante observar que en este algoritmo se crea una clave única para el objeto *WrapOPCGroup* añadido, esta clave sirve para hacer referencia o seleccionar al objeto deseado.

El algoritmo de la función *SetWrapOPCGroup* (perteneciente al objeto *WrapOPCGroup*, ver tabla 2.10) es mostrado en la figura 2.14, la función *SetWrapOPCGroup* es llamado dentro del algoritmo de la figura 2.13.

El algoritmo de la figura 2.15 usa la función definida por el algoritmo de la figura 2.13 para crear un objeto *OPCGroup* y, además, provee la interacción con la interfaz de usuario.

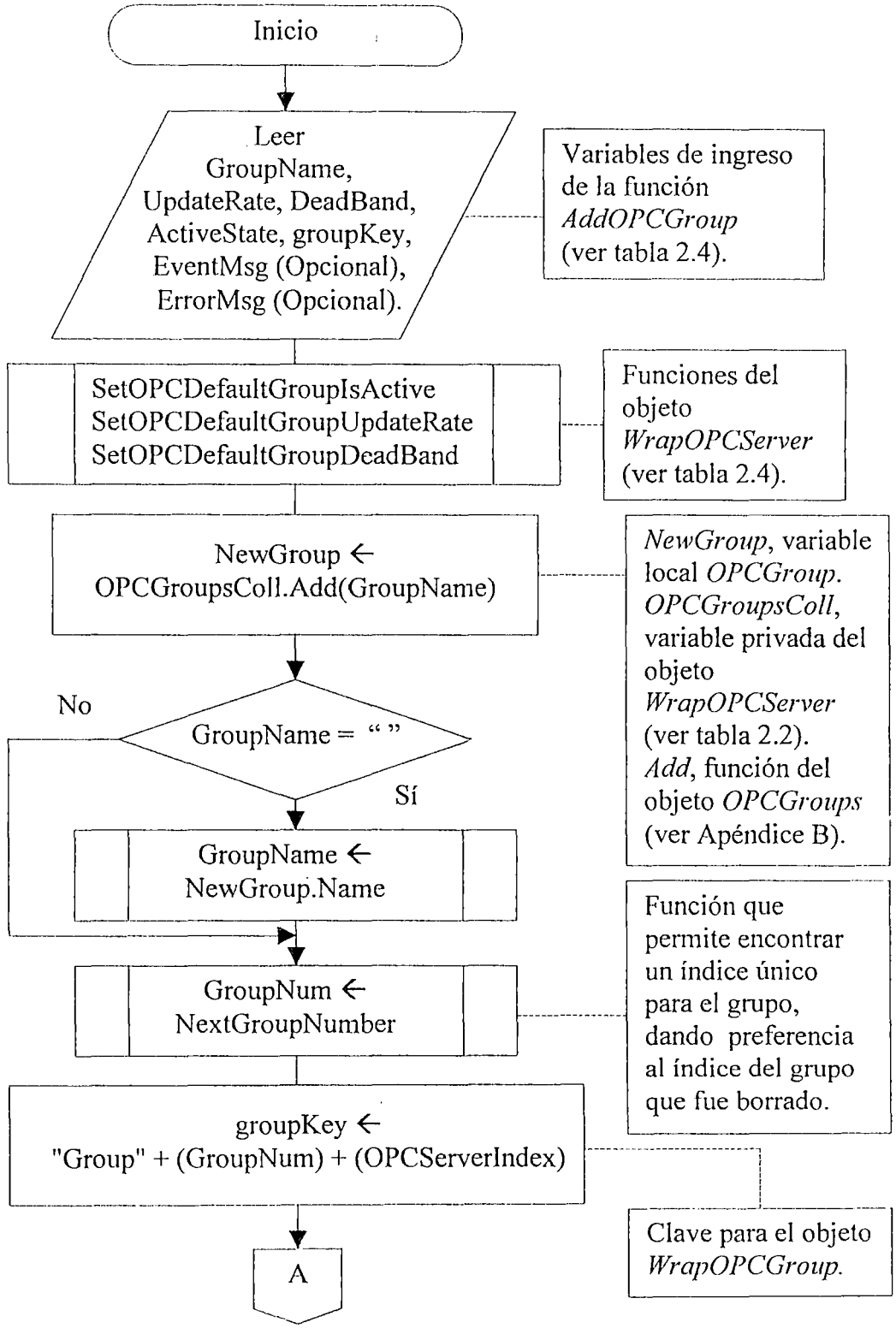


Figura 2.13 Algoritmo de la función *AddOPCGroup* del objeto *WrapOPCServer*

(*continúa...*)

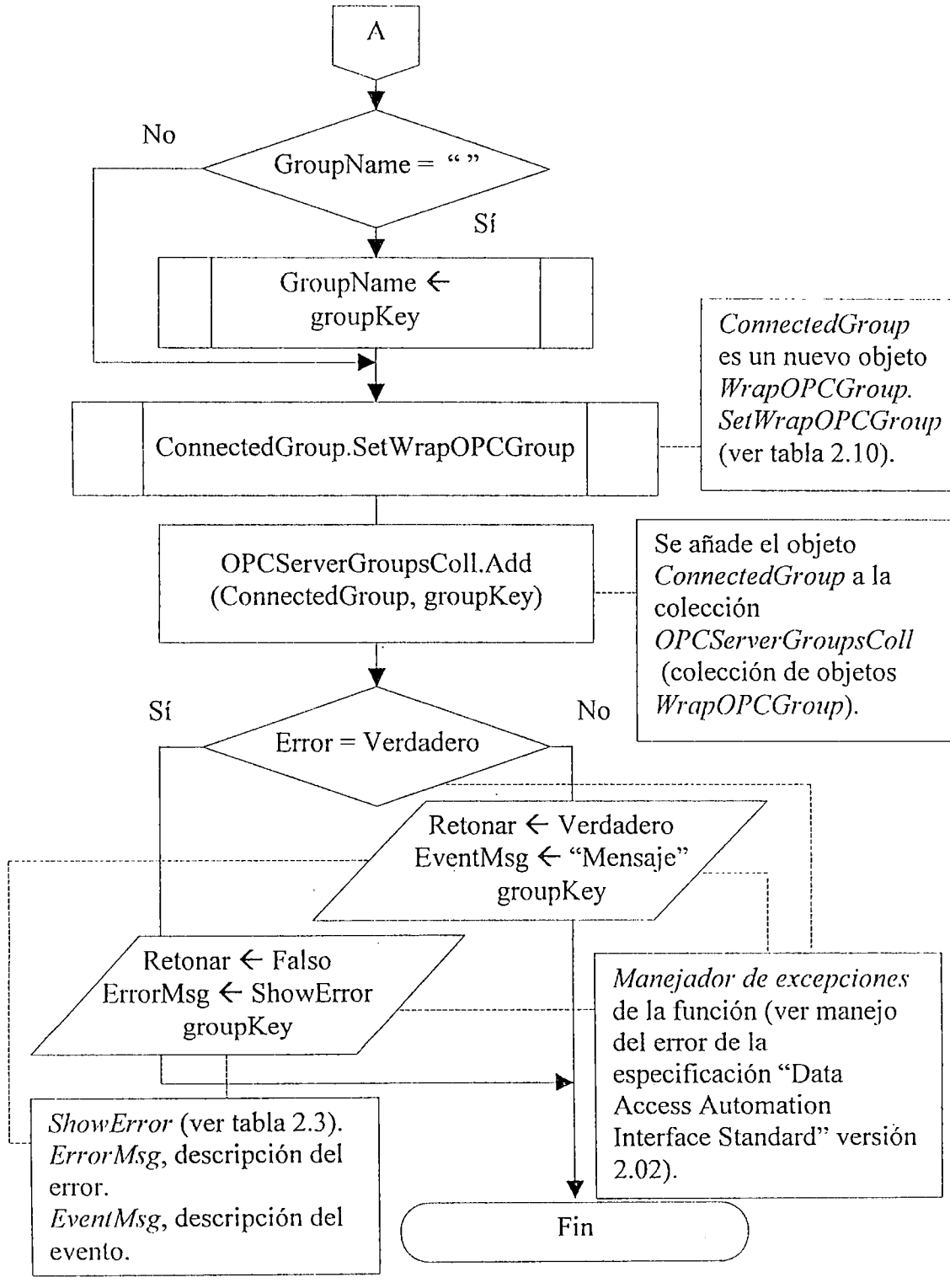


Figura 2.13 (*Continuación de la página 58.*) Algoritmo de la función *AddOPCGroup* del objeto *WrapOPCServer*

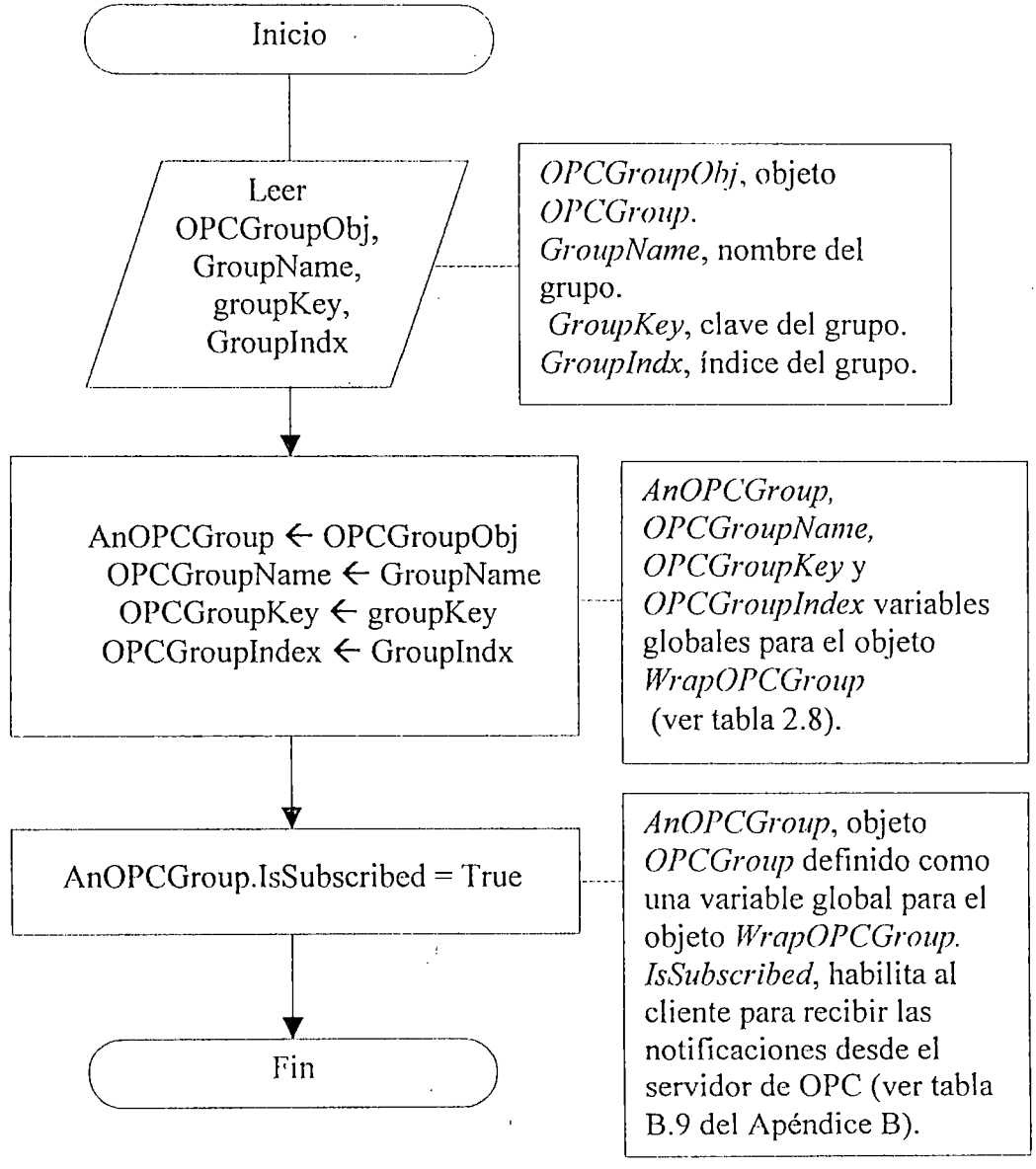


Figura 2.14 Algoritmo de la función *SetWrapOPCGroup* del objeto *WrapOPCGroup*

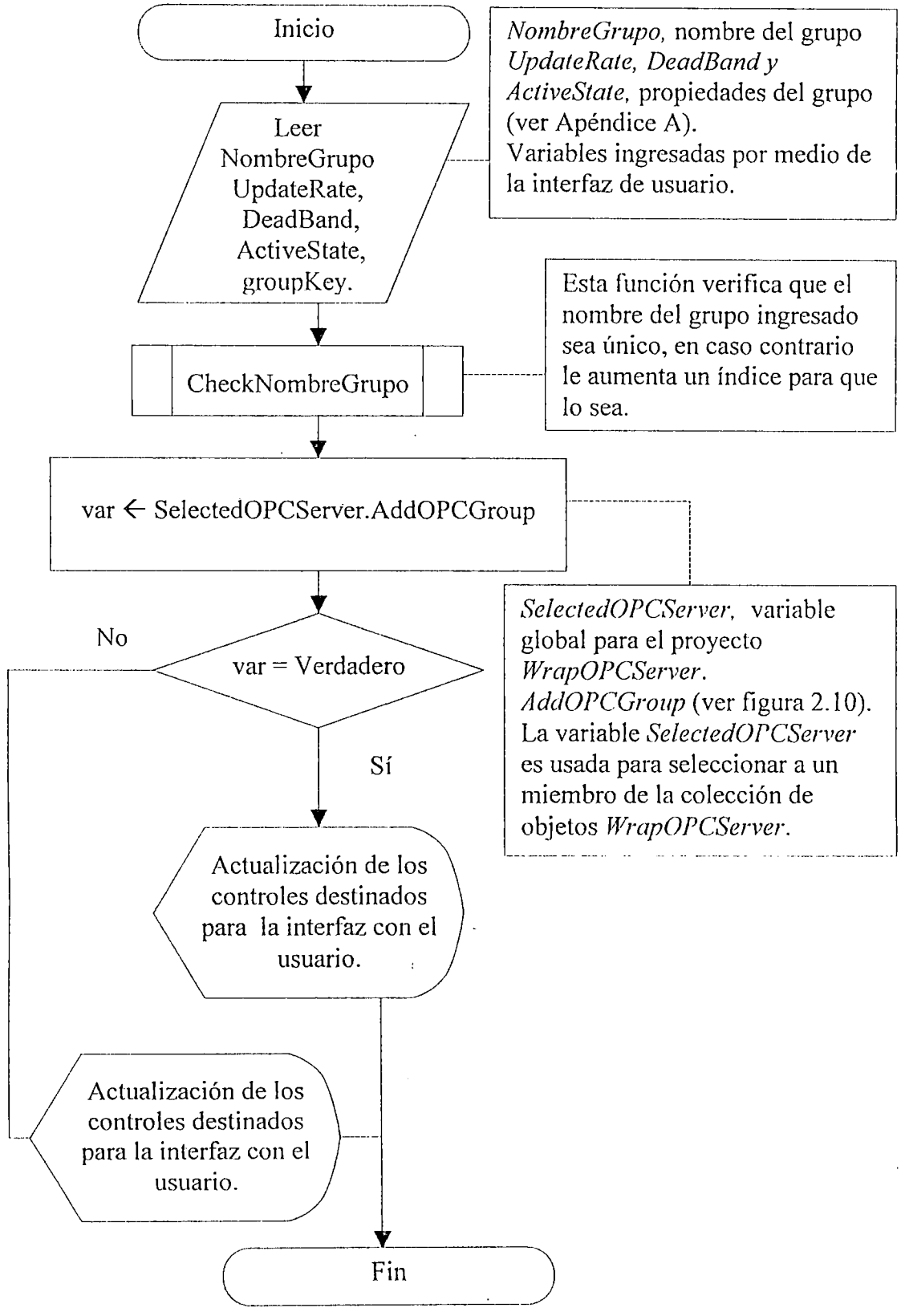


Figura 2.15 Algoritmo para la adición de un grupo (objeto *OPCGroup*) desde la interfaz de usuario

2.1.3.3 Implementación

La implementación es mostrada en las listas 2.3 y 2.4, donde se pueden apreciar los códigos desarrollados para los algoritmos de las figuras 2.13 y 2.15 respectivamente.

En las líneas 2 y 25 al 28 de la lista 2.3 se puede apreciar el manejo del error, este es similar a la lista 2.1. La primera condición de la línea 10 obtiene el nombre del grupo creado por el servidor, cuando el usuario no ha ingresado ninguno. La función de la línea 13 permite obtener un índice para el grupo, con la condición de que ésta no se repita y que tenga preferencia el índice que pertenecía a un grupo eliminado de la colección. La línea 14 define la clave única del grupo. La condición de la línea 15 define al nombre del grupo basándose en su clave, cuando el servidor no ha generado ninguna.

La lista 2.4 llama a la función implementada en la lista 2.3 y actualiza los controles del Visual Basic *lvListViewH* (control *ListView*) y *lvTreeView* (control *TreeView*) encargados de la interfaz de usuario (ver sección 2.2).

```
1.  Function AddOPCGroup(GroupName As String, UpdateRate As Long,
                DeadBand As Single, ActiveState As Boolean,
                ByRef groupKey As String,
                Optional ByRef EventMsg As String,
                Optional ByRef ErrorMsg As String) As Boolean
    ' Manejo del error
2.  On Error GoTo ShowAddOPCGroupError
3.      Dim ConnectedGroup As New WrapOPCGroup
4.      Dim NewGroup As OPCGroup
5.      Dim GroupNum As Integer
6.      SetOPCDefaultGroupIsActive (ActiveState)
7.      SetOPCDefaultGroupUpdateRate (UpdateRate)
8.      SetOPCDefaultGroupDeadBand (DeadBand)

9.      Set NewGroup = OPCGroupsColl.Add(GroupName)
10.     If GroupName = "" Then
11.         GroupName = NewGroup.Name
12.     End If

13.     GroupNum = NextGroupNumber
14.     groupKey = "Group" + Str(GroupNum) + Str(OPCServerIndex)
15.     If GroupName = "" Then
16.         GroupName = groupKey
17.     End If

18.     ConnectedGroup.SetWrapOPCGroup NewGroup, GroupName,
        groupKey, GroupNum
19.     With OPCServerGroupsColl
20.         .Add ConnectedGroup, groupKey
21.     End With

22.     AddOPCGroup = True
23.     EventMsg = "Added group '" + GroupName + "' to " +
        AnOPCServer.serverName
24.     GoTo SkipAddOPCGroupError

25. ShowAddOPCGroupError:
26.     ErrorMsg = ShowError("AddOPCGroup", Err.Number)
27.     AddOPCGroup = False
28. SkipAddOPCGroupError:

29. End Function
```

Lista 2.3 Implementación de la función *AddOPCGroup* del objeto *WrapOPCServer*

```
1.  Sub AddOPCGroupMain(ByVal GroupName As String,
        ByVal UpdateRate As Long,
        ByVal DeadBand As Single, ByVal ActiveState As Boolean)

2.      Dim groupKey As String
3.      Dim EventMsg As String
4.      Dim ErrorMsg As String
5.      Dim itmx As ListItem
6.      Dim NewGroupName As String

7.      FindNameGroup GroupName, NewGroupName
8.      GroupName = NewGroupName

9.      If ModMain.SelectedOPCServer.AddOPCGroup(GroupName,
        UpdateRate, DeadBand, ActiveState, groupKey, EventMsg, ErrorMsg)
        = True Then

10.         Dim nodX As Node    ' Declara la variable Node.
11.         Set nodX = fMainForm.tvTreeView.Nodes.Add(
                ModMain.SelectedOPCServer.GetOPCServerKey,
                tvwChild, groupKey, GroupName, 7, 8)
12.             nodX.EnsureVisible

13.         Set itmx = lvListViewH.ListItems.Add(, , Date, 21, 21)
14.             itmx.SubItems(1) = Time ' Visualiza el tiempo
15.             itmx.SubItems(2) = EventMsg ' Muestra el error
16.             itmx.EnsureVisible
17.     Else
18.         Set itmx = lvListViewH.ListItems.Add(, , Date, 13, 13)
19.             itmx.SubItems(1) = Time ' Visualiza el tiempo
20.             itmx.SubItems(2) = ErrorMsg ' Muestra el error
21.             itmx.EnsureVisible
22.     End If


23. End Sub
```

Lista 2.4 Implementación del algoritmo de la figura 2.15

2.1.4 Funcionalidad para la Adición y Eliminación de Items de OPC

Los siguientes procedimientos se llevaron a cabo para la implementación de esta funcionalidad:

2.1.4.1 Descomposición Orientada a Objetos de la Funcionalidad

El comportamiento dinámico que se tendrá cada vez que se desea agregar un nuevo objeto ítem al objeto grupo seleccionado es mostrado en la figura 2.16. En la figura 2.17 se muestra la secuencia necesaria que se necesita cada vez que se desea eliminar un objeto ítem seleccionado.

Las funciones de objeto *SelectedOPCGroup* (*WrapOPCGroup*) mostradas en las secuencias 1 y 2 de la figura 2.16, llaman respectivamente a las funciones de las secuencias 1 y 2 del objeto *OPCGroup* de la figura 2.16. El suceso 5 de la figura 2.16 adiciona el nuevo objeto *ItemToAdd* (*WrapOPCItem*) a la colección de objetos *WrapOPCItem* (*OPCGroupItemsColl*).

La función *GetOPCItemServerHandle* del objeto *OPCItemToRemove* (*WrapOPCItem*) llama al método *ServerHandle* del objeto *AnOPCItem* (*OPCItem*), en la secuencia de sucesos de la figura 2.17

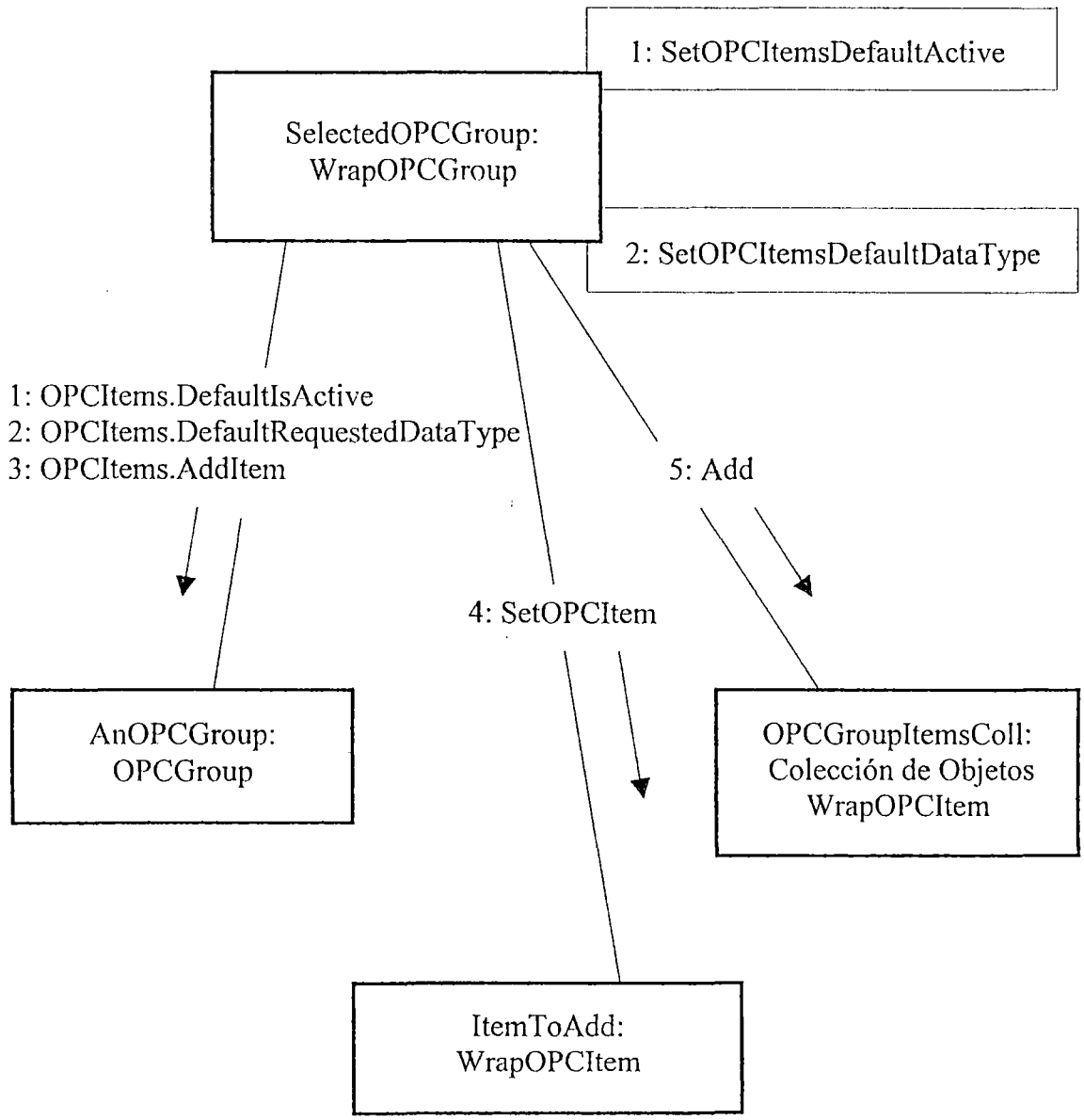


Figura 2.16 Descomposición orientada a objetos para la creación de un nuevo ítem (objeto *OPCItem*)

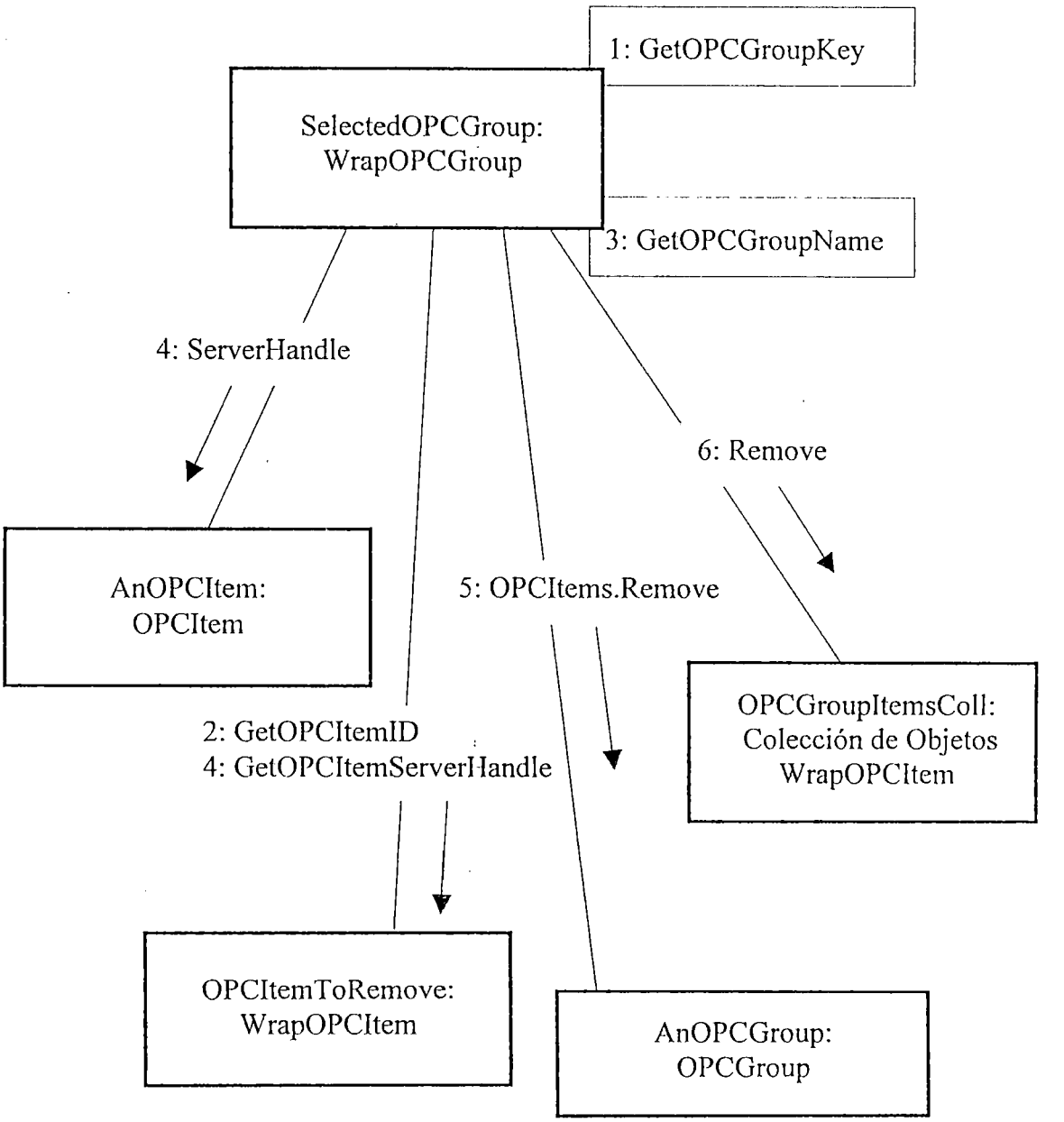


Figura 2.17 Descomposición orientada a objetos para la eliminación de un ítem

(objeto *OPCItem*)

2.1.4.2 Diseño del Algoritmo

El algoritmo de la función *AddOPCItem* de la figura 2.18 se encargada de la creación de un nuevo ítem. Este algoritmo surge sobre la base del esquema de la figura 2.16.

El algoritmo de la figura 2.18 usa la función *SetOPCItem,* cuyo algoritmo es mostrado en la figura 2.19. El algoritmo de la figura 2.20 usa la función definida por el algoritmo de la figura 2.18 para crear un objeto *OPCGItem* y, además provee la interacción con la interfaz de usuario.

El algoritmo definido por la figura 2.20 es llamado por medio de la interfaz de usuario, cada vez que se desea adicionar un nuevo ítem al grupo seleccionado y además, actualiza los controles destinados para la interfaz de usuario (ver sección 2.2).

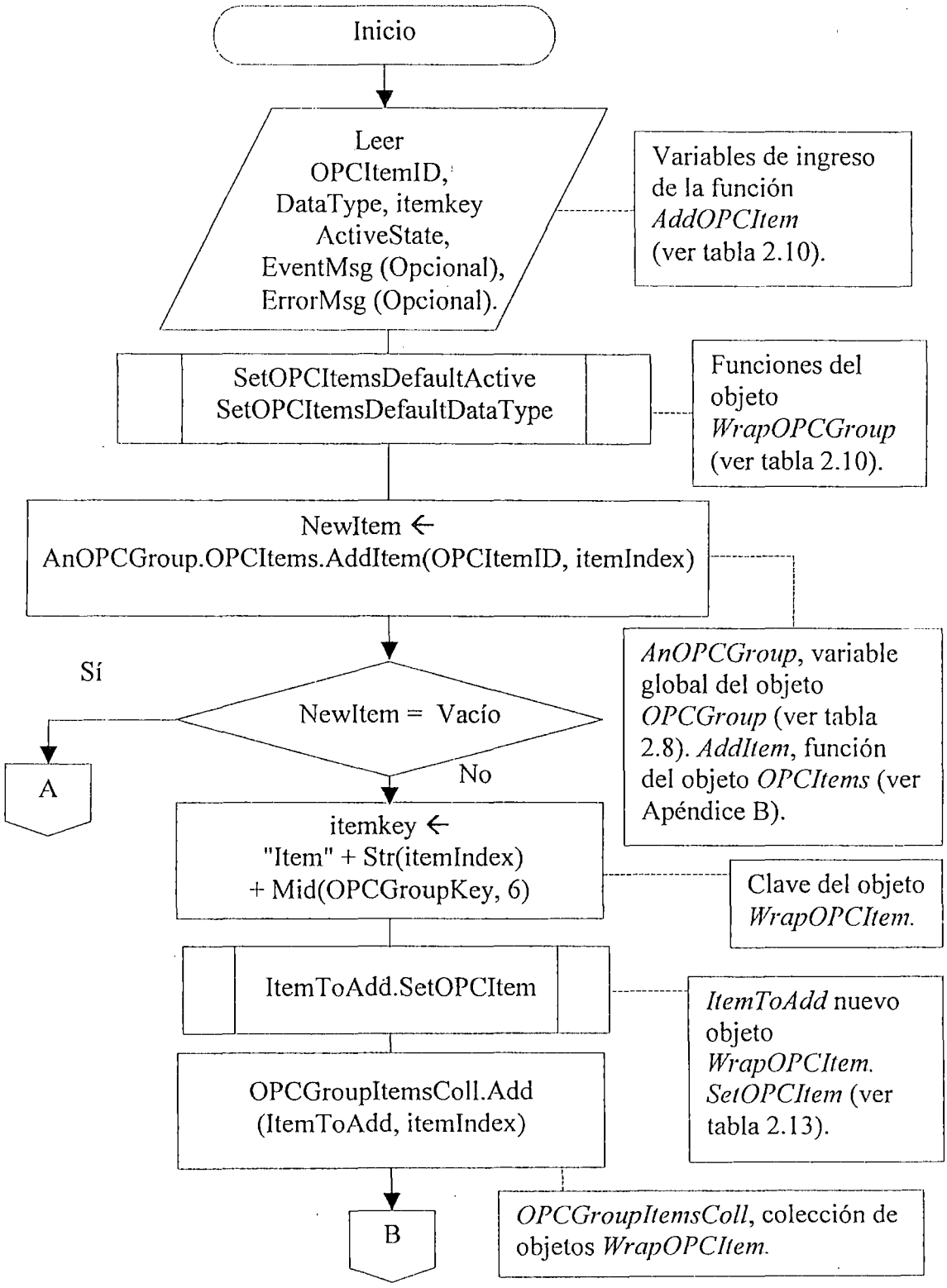


Figura 2.18 Algoritmo de la función *AddOPCItem* del objeto *WrapOPCGroup*

(*continúa...*)

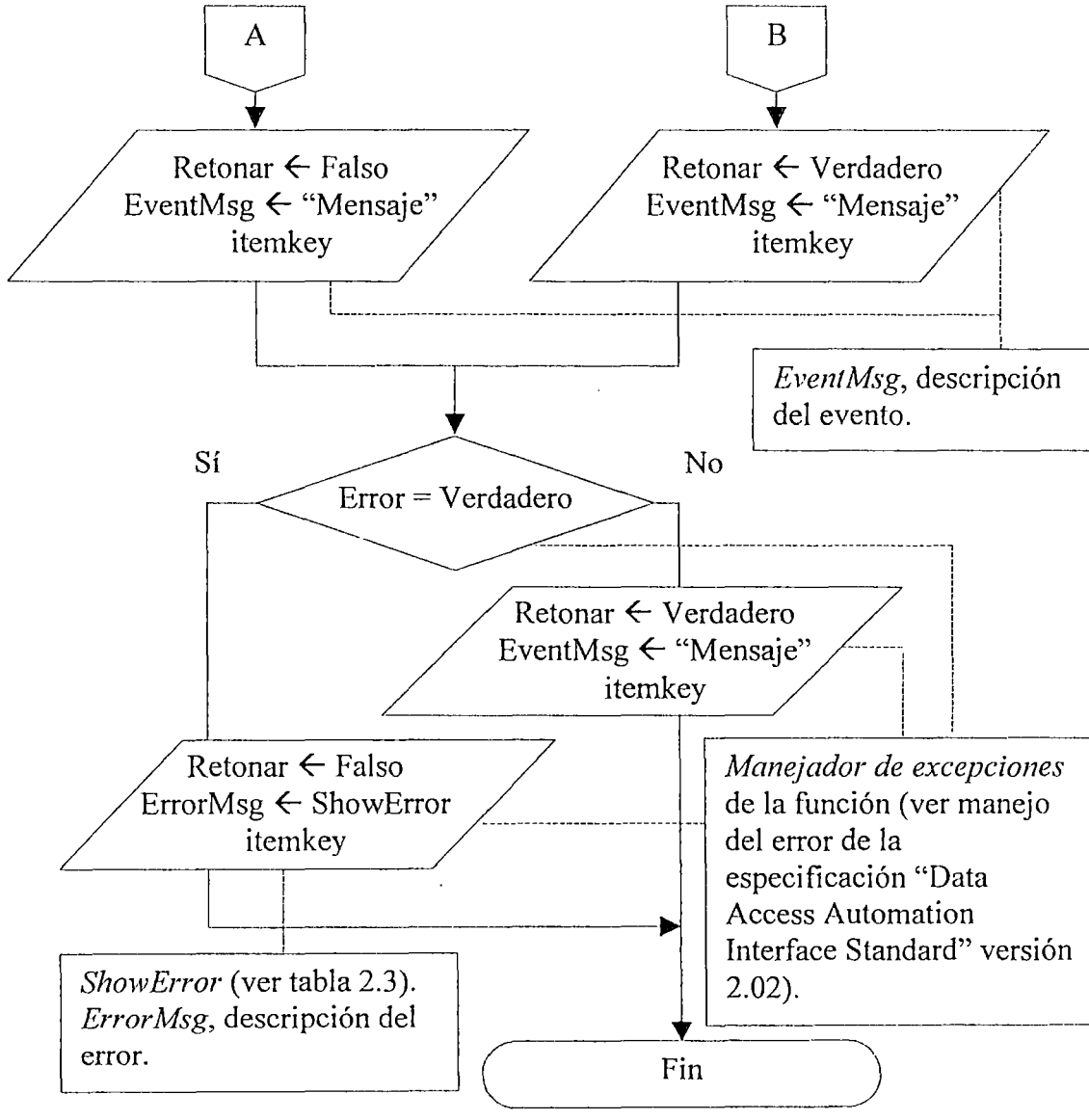


Figura 2.18 (*Continuación de la página 69.*) Algoritmo de la función *AddOPCItem* del objeto *WrapOPCGroup*

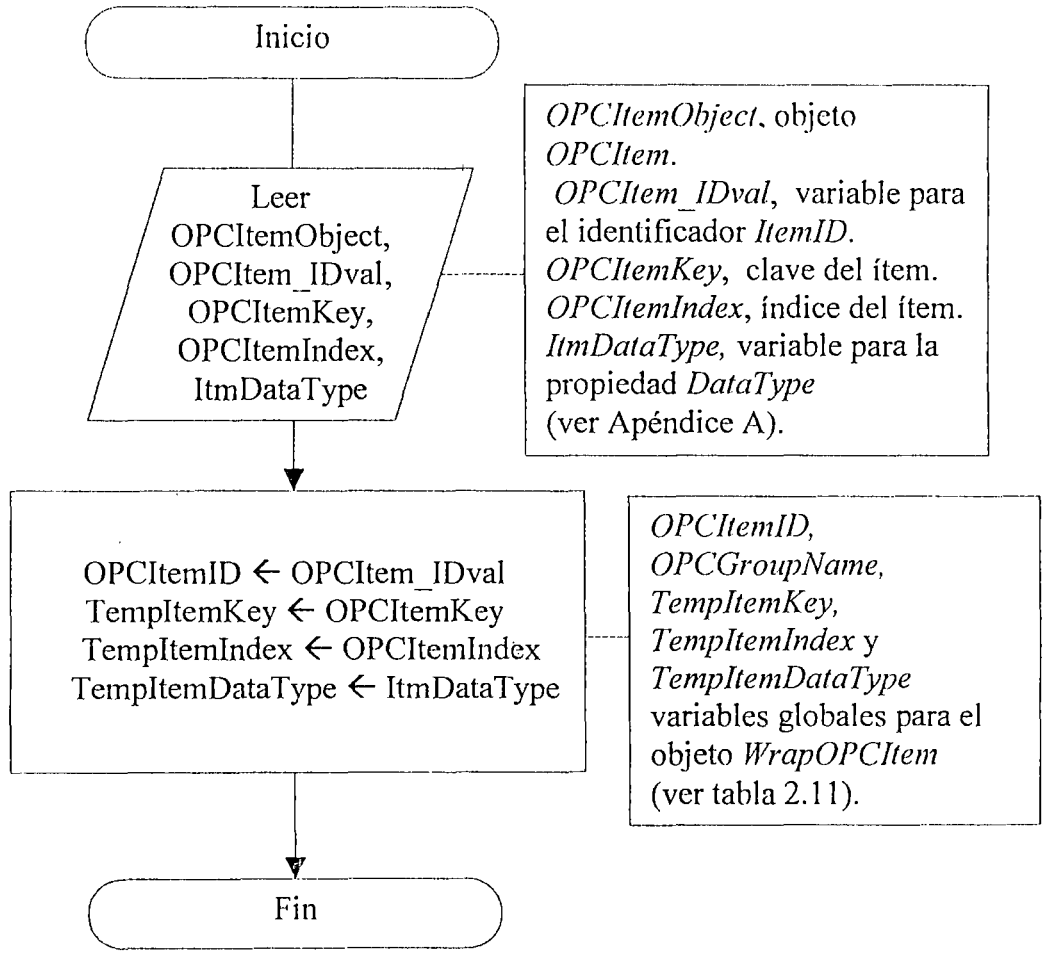


Figura 2.19 Algoritmo de la función *SetOPCItem* del objeto *WrapOPCItem*

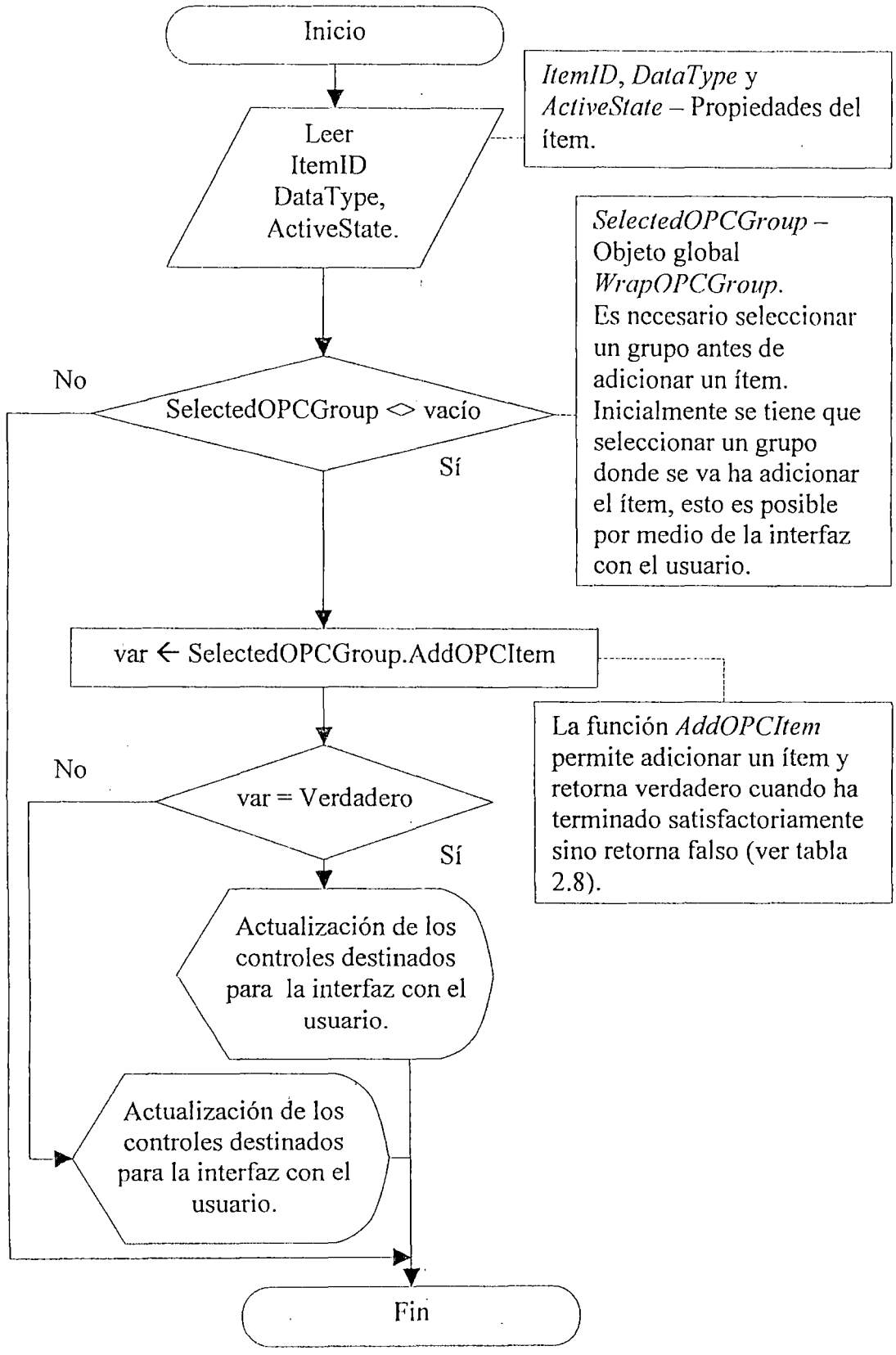


Figura 2.20 Algoritmo para la adición de un ítem (objeto *OPCItem*) desde la interfaz de usuario

2.1.4.3 Implementación

El desarrollo del código correspondiente al algoritmo de la figura 2.18 se muestra en la lista 2.5. Las líneas 2 y 25 al 28 de la lista 2.5 corresponden al manejo del error en tiempo de ejecución (ver manejo del error de la especificación "Data Access Automation Interface Standard" versión 2.02) .

La función *SetOPCItemsDefaultActive* de la línea 7 de la lista 2.5 es usada para llamar al método *DefaultIsActive* del objeto *OPCItems*. Este método es usado para configurar el parámetro *ActiveState* del objeto *OPCItems*. En forma similar la función *SetOPCItemsDefaultDataType* de la línea 8 de la lista 2.5 llama al método *DefaultRequestedDataType* del objeto *OPCItems*. Este método es llamado con el objetivo de configurar el parámetro *DataType* del objeto *OPCItems*.

En la línea 13 de la lista 2.5 se puede ver la definición de la clave única del ítem, el cual se usará para hacer referencia a un elemento de la colección de objetos *WrapOPCItem*.

La implementación del algoritmo de la figura 2.20 es mostrada en la lista 2.6, donde se puede apreciar que las variables *tvTreewView*, *lvListViewH* y *lvListView* son controles del Visual Basic usados para interactuar con la interfaz de usuario (ver sección 2.2).

```
1.  Function AddOPCItem(ByVal OPCItemID As String,
                ByVal DataType As Integer, ByVal ActiveState As Integer.
                ByRef itemkey As String,
                Optional ByRef EventMsg As String,
                Optional ByRef ErrorMsg As String) As Boolean

    ' Manejo del error
2.  On Error GoTo ShowOPCItemAddError

3.      Dim ItemToAdd As New WrapOPCItem
4.      Dim NewItem As OPCItem
5.      Dim itemIndex As Integer
6.      Dim TmpOPCItemID As String

7.      SetOPCItemsDefaultActive (ActiveState)
8.      SetOPCItemsDefaultDataType (DataType)
9.      TmpOPCItemID = OPCItemID

10.     itemIndex = NextItemNumber
11.     Set NewItem = AnOPCGroup.OPCItems.AddItem(OPCItemID,
                                                  itemIndex)
12.     If Not NewItem Is Nothing Then
13.         itemkey = "Item" + Str(itemIndex) + Mid(OPCGroupKey, 6)
14.         ItemToAdd.SetOPCItem NewItem, TmpOPCItemID,
                        itemKey, itemIndex, DataType

15.         With OPCGroupItemsColl
16.             .Add ItemToAdd, itemKey
17.         End With
18.         EventMsg = "Added item " & OPCItemID & " to the group '"
            & AnOPCGroup.Name & "'"
19.         AddOPCItem = True
20.     Else
21.         EventMsg = "The item " & OPCItemID & " can not be added to
            the goup " & AnOPCGroup.Name
22.         AddOPCItem = False
23.     End If
24.     GoTo SkipAddItemError
25. ShowOPCItemAddError:
26.     ErrorMsg = ShowError("AddOPCItem", Err.Number)
27.     AddOPCItem = False
28. SkipAddItemError:

29. End Function
```

Lista 2.5 Implementación de la función *AddOPCGroup* del objeto *WrapOPCServer*

```
1.  Function AddOPCItemMain(ByVal ItemID As String,
                ByVal DataTypeSelected As Integer,
                ByVal ActiveState As Integer)

2.  Dim itemKey As String
3.  Dim EventGC As String
4.  Dim ErrorGC As String
5.  Dim itmx As ListItem
6.  Dim nodeX As Node
7.  Dim varKey As String
8.  If Not ModMain.SelectedOPCGroup Is Nothing Then
9.      If ModMain.SelectedOPCGroup.AddOPCItem(ItemID,
        DataTypeSelected, ActiveState, itemKey, EventGC, ErrorGC) =
        False Then
10.         If ErrorGC <> "" Then
11.             Set itmx = lvListViewH.ListItems.Add(, , Date, 13, 13)
12.             itmx.SubItems(1) = Time
13.             itmx.SubItems(2) = ErrorGC
14.             itmx.EnsureVisible
15.         Else
16.             Set itmx = lvListViewH.ListItems.Add(, , Date, 13, 13)
17.             itmx.SubItems(1) = Time
18.             itmx.SubItems(2) = EventGC
19.             itmx.EnsureVisible
20.         End If
21.           AddOPCItemMain = False
22.     Else
23.         varKey = ModMain.SelectedOPCGroup.GetOPCGroupKey
24.         Set itmx = lvListView.ListItems.Add(,itemKey, ItemID, 10, 10)
25.         itmx.SubItems(2) = ""
26.         itmx.SubItems(4) = "Bad"
27.         Set nodeX = tvTreeView.Nodes.Add(varKey, tvwChild,
            itemKey, Mid(ItemID, InStrRev(ItemID, ".") + 1, Len(ItemID)
            - InStrRev(ItemID, ".")), 10, 10)
28.         nodeX.EnsureVisible
29.         Set itmx = lvListViewH.ListItems.Add(, , Date, 21, 21)
30.         itmx.SubItems(1) = Time
31.         itmx.SubItems(2) = EventGC
32.         itmx.EnsureVisible
33.         AddOPCItemMain = True
34.     End If
35. End If
```

Lista 2.6 Implementación del algoritmo de la figura 2.20

2.1.5 Funcionalidad para la Creación del objeto *WrapOPCBrowser*

Los siguientes pasos fueron tomados en cuenta para la implementación de esta funcionalidad:

2.1.5.1 Descomposición Orientada a Objetos de la Funcionalidad

El esquema mostrado en la figura 2.21 se llevará a cabo cada vez que el usuario intenta acceder al *namespace* del servidor de OPC seleccionado.

El comportamiento dinámico mostrado en la figura 2.21 no especifica cuando el servidor de OPC no soporta el objeto *OPCBrowser*, por lo tanto este requerimiento es mostrado en el diseño del algoritmo.

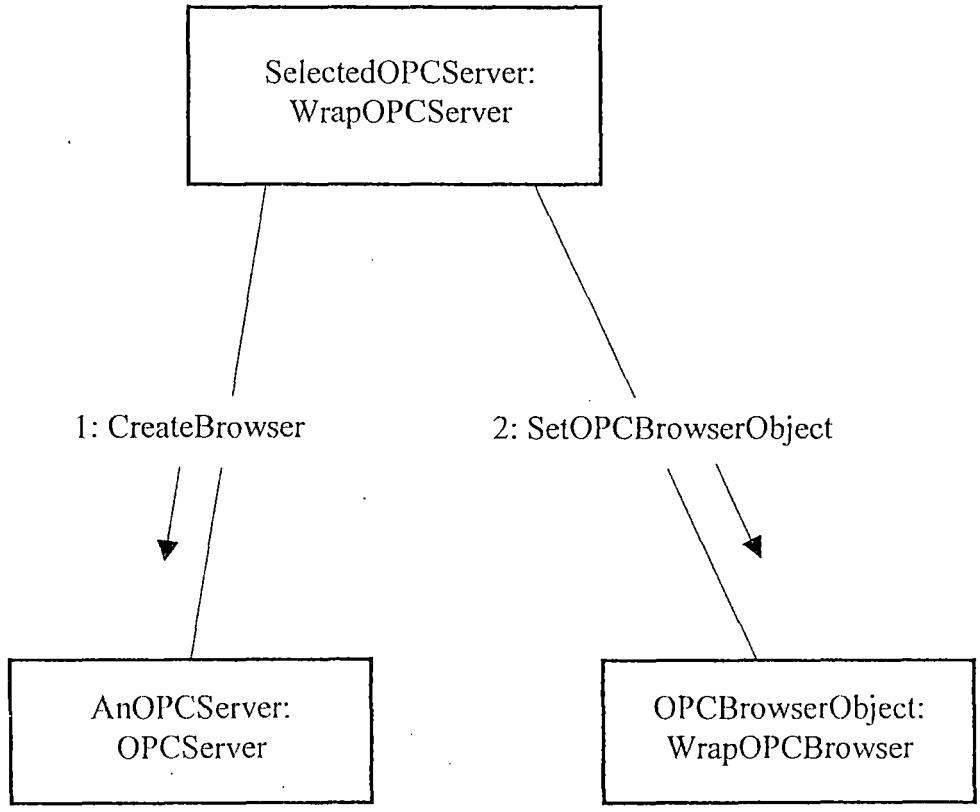


Figura 2.21 Descomposición orientada a objetos para la creación del objeto *OPCBrowser*

2.1.5.2 Diseño del Algoritmo

En base del esquema de la figura 2.21, se crea el algoritmo de la función *GetOPCServerBrowseObject,* encargada de la creación de un objeto *OPCBrowser*. Este algoritmo es mostrado en la figura 2.22. Es importante notar que este algoritmo define el caso cuando un servidor de OPC no soporta el objeto *OPCBrowser*, por lo tanto no se debe crear un *namespace*.

2.1.5.3 Implementación

La lista 2.7 muestra el código desarrollado para el algoritmo de la figura 2.22. Tal como se puede apreciar en las líneas 2 y 13 al 16 el manejo del error en tiempo de ejecución es similar a las funciones mostradas anteriormente.

Cuando el servidor de OPC no soporta el objeto *OPCBrowser* la función *GetOPCServerBrowseObject* implementada en la lista 2.7 devuelve la palabra clave "*Nothing*" (línea 9 de la lista 2.7), que indica que no hace referencia a ningún objeto.

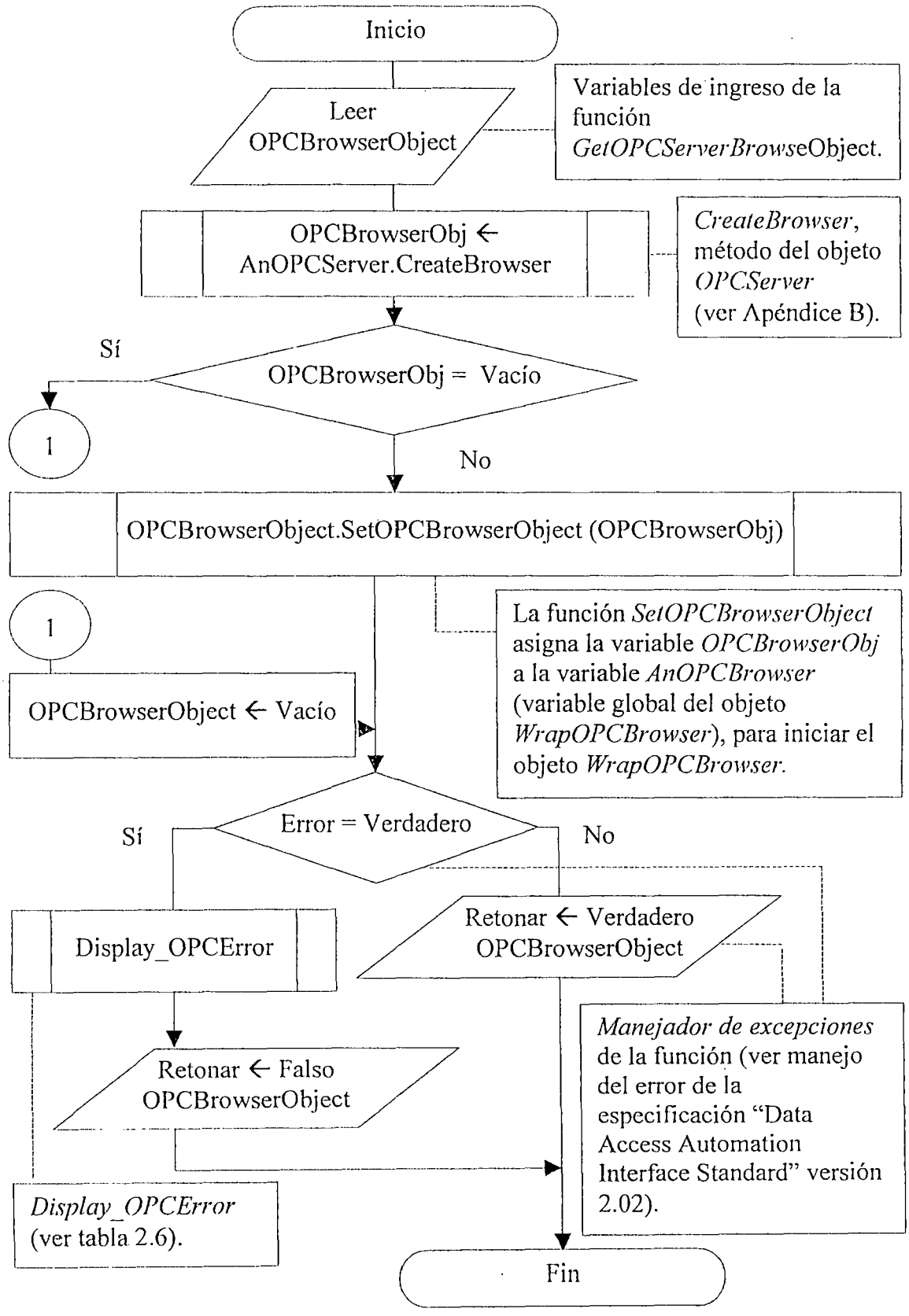


Figura 2.22 Algoritmo de la función *GetOPCServerBrowseObject* del objeto *WrapOPCServer*

```
1.  Function GetOPCServerBrowseObject(
            ByRef OPCBrowserObject As WrapOPCBrowser) As Boolean

    ' Manejo del error
2.  On Error GoTo ShowGetOPCServerBrowseObjectError

3.      Dim OPCBrowserObj As OPCBrowser


4.      Set OPCBrowserObj = AnOPCServer.CreateBrowser

5.      If Not OPCBrowserObj Is Nothing Then
6.          Set OPCBrowserObject = New WrapOPCBrowser
7.          OPCBrowserObject.SetOPCBrowserObject OPCBrowserObj
8.      Else
9.          Set OPCBrowserObject = Nothing
10.     End If

11.     GetOPCServerBrowseObject = True
12.     GoTo SkipGetOPCServerBrowseObjectError

13. ShowGetOPCServerBrowseObjectError:
14.     Call Display_OPCError("GetOPCServerBrowseObject",
        Err.Number)
15.     GetOPCServerBrowseObject = False
16. SkipGetOPCServerBrowseObjectError:

17. End Function
```

Lista 2.7 Implementación del algoritmo de la figura 2.22

2.1.6 Funcionalidad para el *Namespace*

Esta funcionalidad consta en poder acceder a toda la información provista por el *namespace* del servidor de OPC. Este acceso se debe dar por medio de la interfaz de usuario. A continuación se tendrá el análisis de esta funcionalidad por medio de dos herramientas (Descomposición Orientada a Objetos y Diagrama de Flujo).

2.1.6.1 Descomposición Orientada a Objetos de la Funcionalidad

El comportamiento dinámico para esta funcionalidad es mostrado claramente en las figuras 2.23, 2.24 y 2.25.

La descomposición de la figura 2.23 es llamada cada vez que se inicia la búsqueda dentro del *namespace*. *StartObject* es un objeto propio del Visual Basic (objeto *Tab* del Control *TabStrip*, ver manual de usuario del Visual Basic), usado para iniciar la búsqueda dentro del *namespace*. Este objeto es usado por la interfaz de usuario para iniciar la búsqueda.

Las funciones *GetOPCServerKey* y *GetOPCServerName* de la figura 2.23 son utilizadas para mostrar y crear las ramas que se verán por medio de la interfaz de usuario. Tal como se puede apreciar en la figura 2.23, la función *GetOPCServerName* llama al método *serverName* (perteneciente al objeto *OPCServer*, ver Apéndice B) para obtener el nombre del servidor de OPC.

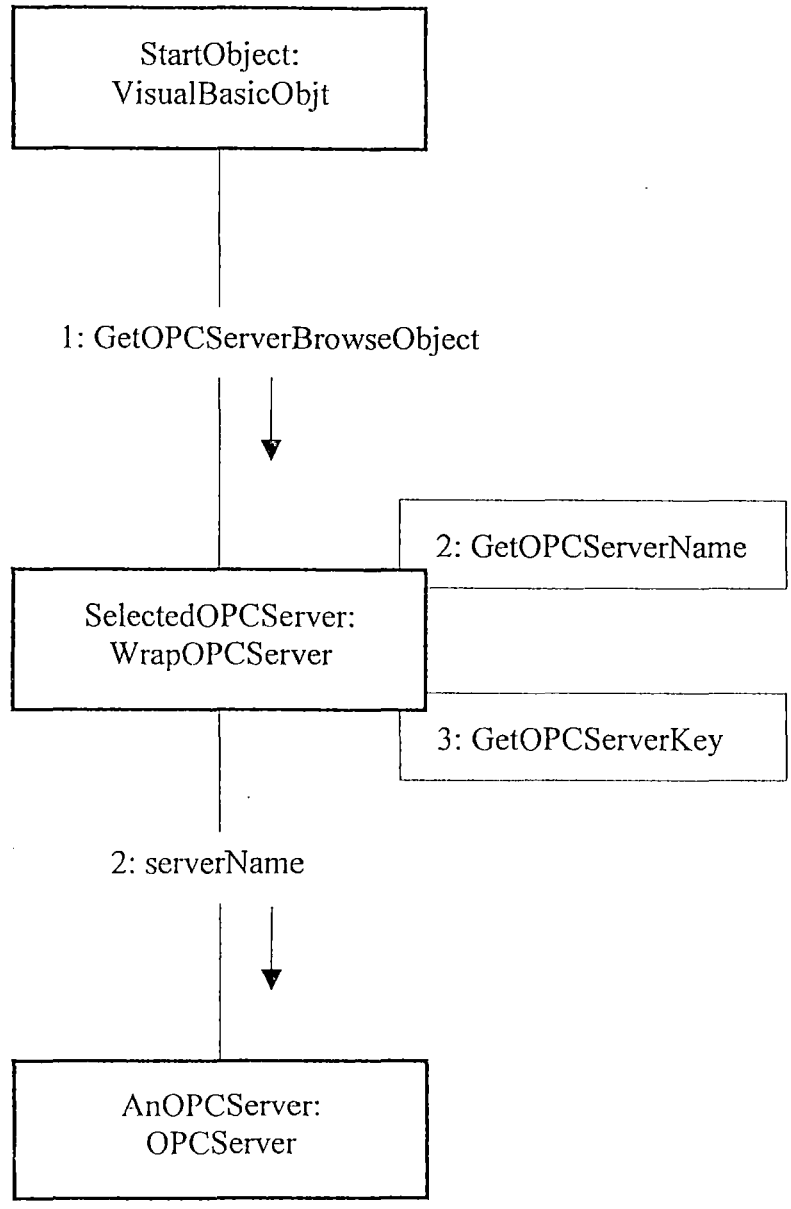


Figura 2.23 Descomposición orientada a objetos para el inicio de la búsqueda dentro del *namespace*

Una vez iniciado la búsqueda, es decir iniciado el objeto *OPCBrowser*, se lleva a cabo la secuencia de sucesos de la figura 2.24

En la figura 2.24 se puede apreciar el objeto *BrowserObject,* el cual viene ha ser el objeto *Node* del control *TreeView tvBranchView* (ver sección 2.2, para más información ver el manual de usuario del Visual Basic).

Para el inicio de la secuencia de la figura 2.24, se usa el evento *Expand* del control *TreeView tvBranchView.* Este control permite navegar dentro del *namespace* e interactuar con la interfaz de usuario de manera fácil.

La secuencia de sucesos comienza con la función *SetBrowsePosition.* La descomposición de la función *SetBrowsePosition* es mostrada en la figura 2.25, el cual permite ubicarse dentro del *namespace*, de acuerdo al nodo actual, en la posición establecida por la interfaz de usuario

Una vez situado en la posición requerida se llama las ramas existentes (*OPCShowBranches*) en la posición actual y se muestran por medio de la interfaz de usuario (usando las funciones *GetOPCBrowserOrganization*, *GetOPCItemCount* y *GetOPCItemName,* ver tabla 2.7), pero inicialmente se aplica la función *SetOPCFilter* (ver tabla 2.7*)* para filtrar las ramas que no son necesarias. Una vez mostrados las ramas existentes de la posición actual, el siguiente paso consiste en realizar la misma lógica para mostrar todas las hojas o items pertenecientes al nodo actual.

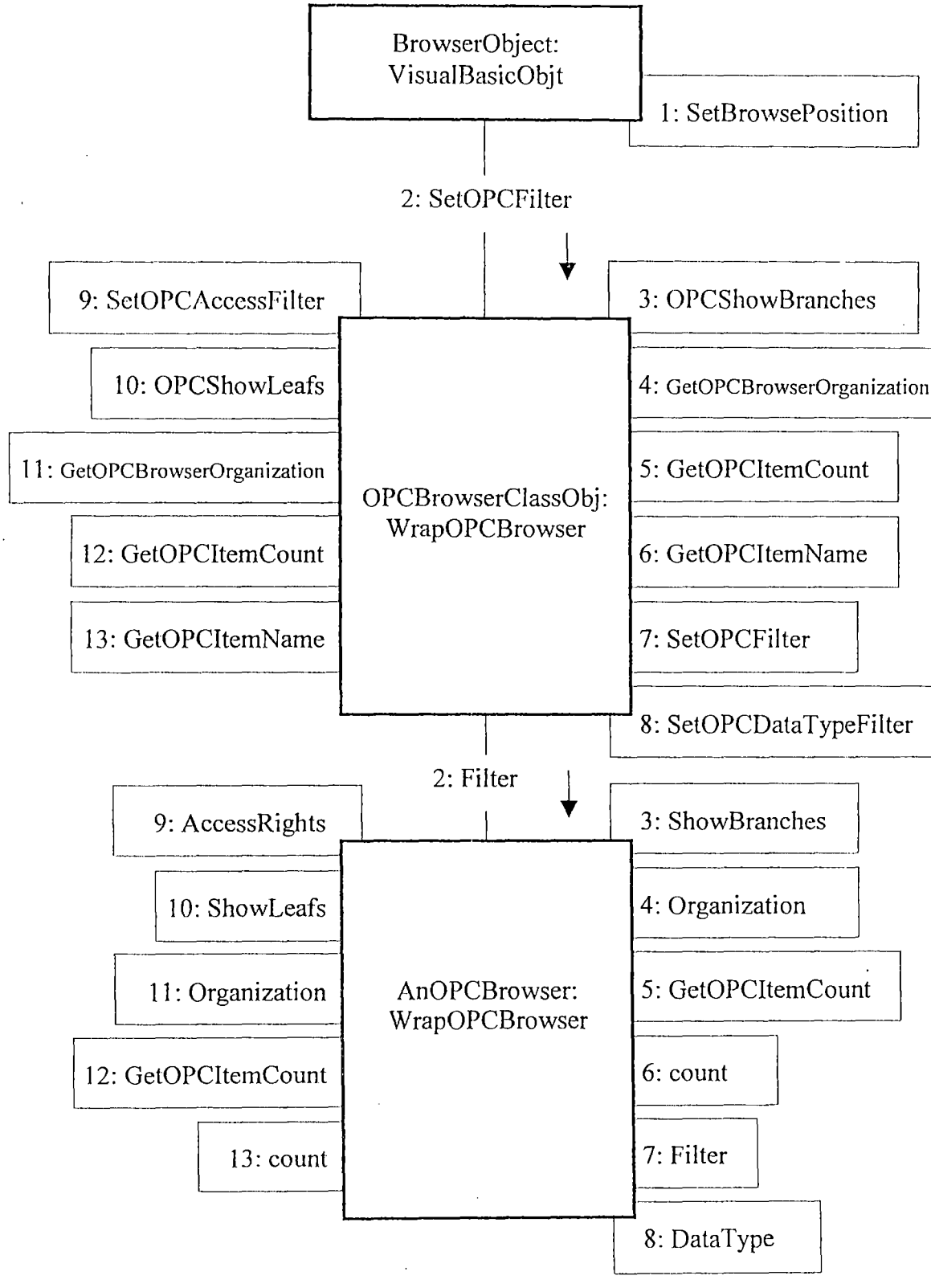


Figura 2.24 Descomposición orientada a objetos para la búsqueda dentro del *namespace*

La secuencia de sucesos de la función *SetBrowsePosition* es mostrada en la figura 2.25, donde se puede observar que inicialmente se llama a la función *OPCMoveToRoot* (ver tabla 2.7). Esta función se utiliza para comenzar desde la raíz del *namespace* y luego buscar la posición pedida dentro del nodo actual, por medio de la función *OPCMoveDown* (ver tabla 2.7). Esta última función se llamará tantas veces que sea necesario hasta posicionarse adecuadamente.

Una explicación más clara se puede obtener al usar un diagrama de bloques para esta función (ver figura 2.27).

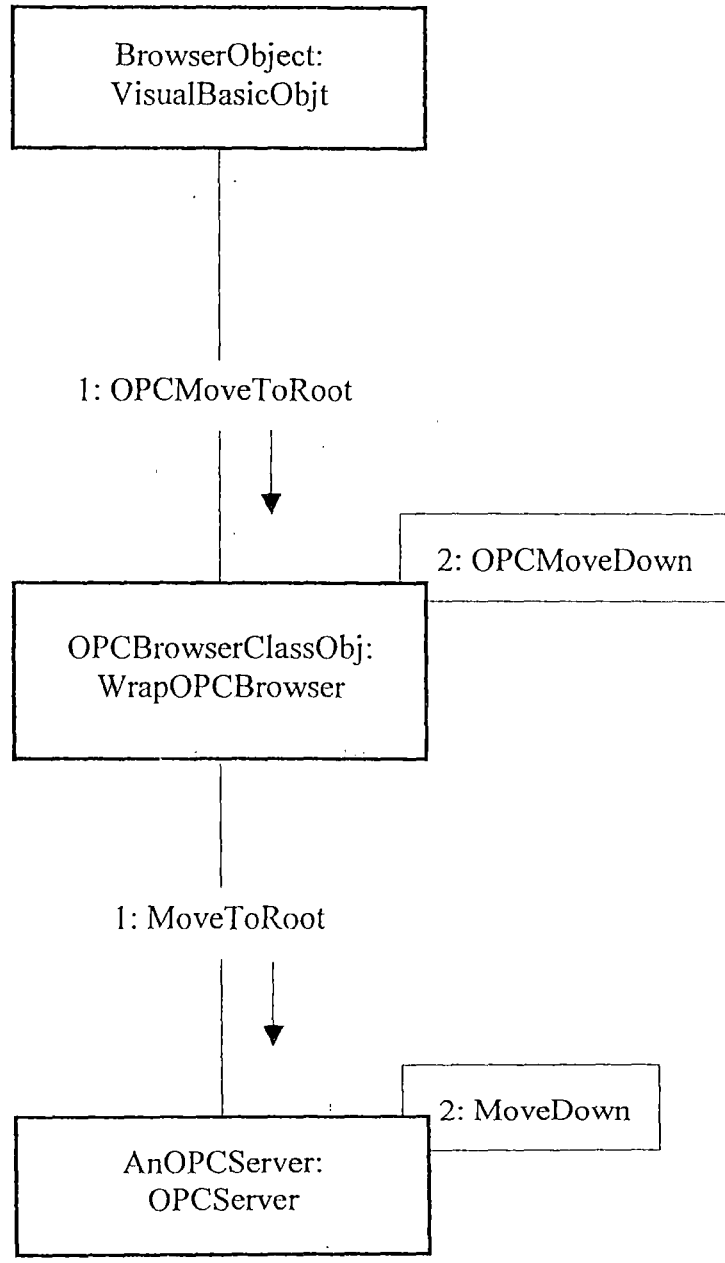


Figura 2.25 Descomposición orientada a objetos de la función *SetBrowsePosition,* para la ubicación específica dentro del *namespace*

2.1.6.2 Diseño del Algoritmo

Los algoritmos de las figuras 2.26, 2.27 y 2.28 muestran en forma clara el procedimiento necesario para poder interactuar dentro del *namespace*.

El algoritmo de la figura 2.26 se debe llamar cada vez que se inicia la búsqueda dentro del *namespace*. Es importante observar que la creación de los *nodos virtuales* en el control *TreeView tvBranchView* (ver sección 2.2) de la interfaz de usuario permite tener un nodo para expandir. Los *nodos virtuales* son luego eliminados, tal como se muestra en la figura 2.27.

El evento *Expand* de objeto *Nodo* del control *TreeView tvBranchView* (ver sección 2.2) de la interfaz de usuario es mostrado en el algoritmo de la figura 2.27, en este evento se mostrarán todos las ramas e items que se encuentra en la posición actual.

El algoritmo de la figura 2.27 usa la función *SetBrowsePosition,* cuyo algoritmo es mostrado en la figura 2.28. Esta función permite que la posición provista por la interfaz de usuario (control *TreeView tvBranchView*, ver sección 2.2) sea la misma para el *namespace*.

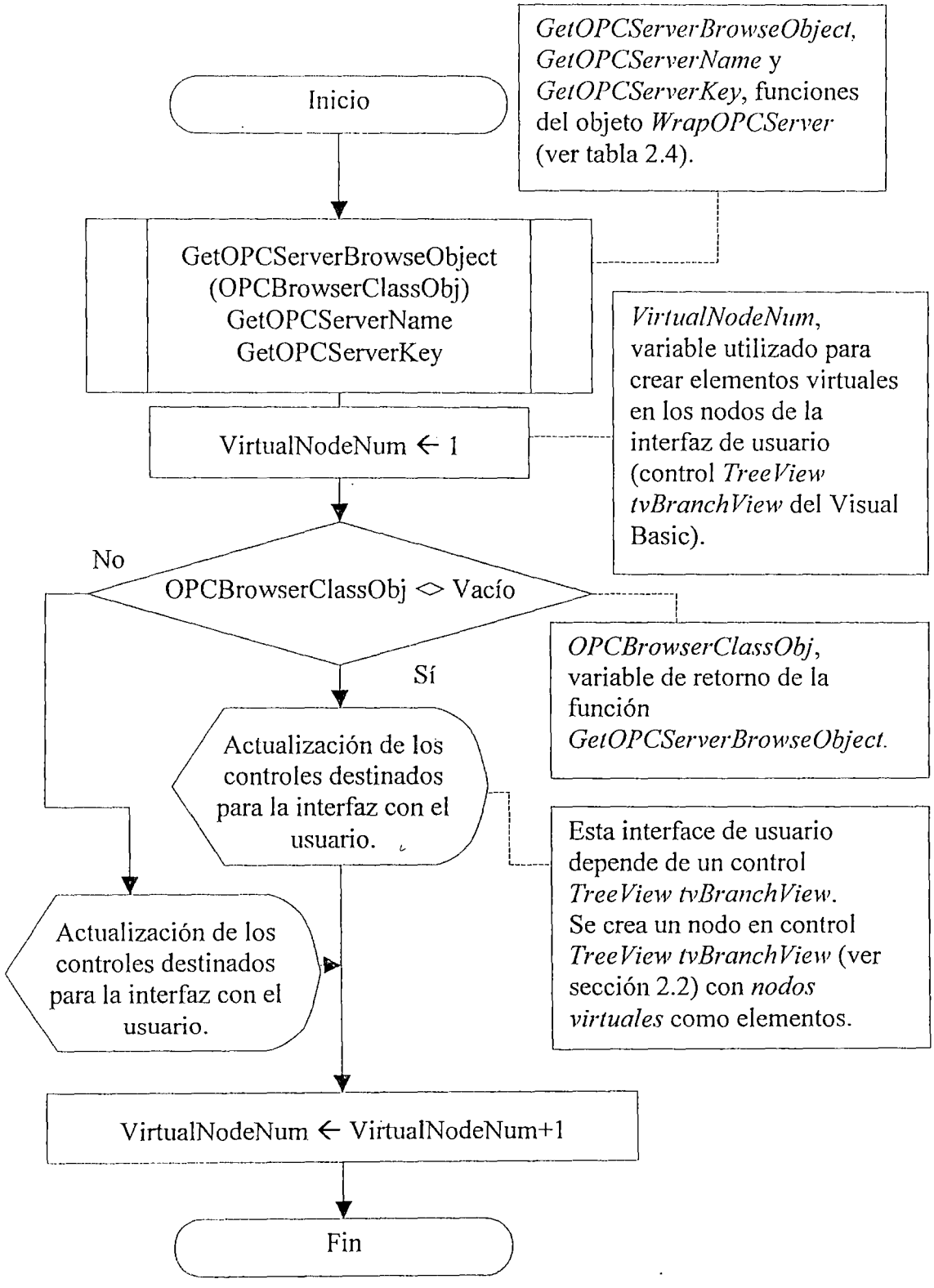


Figura 2.26 Algoritmo para el inicio de la búsqueda dentro del *namespace*

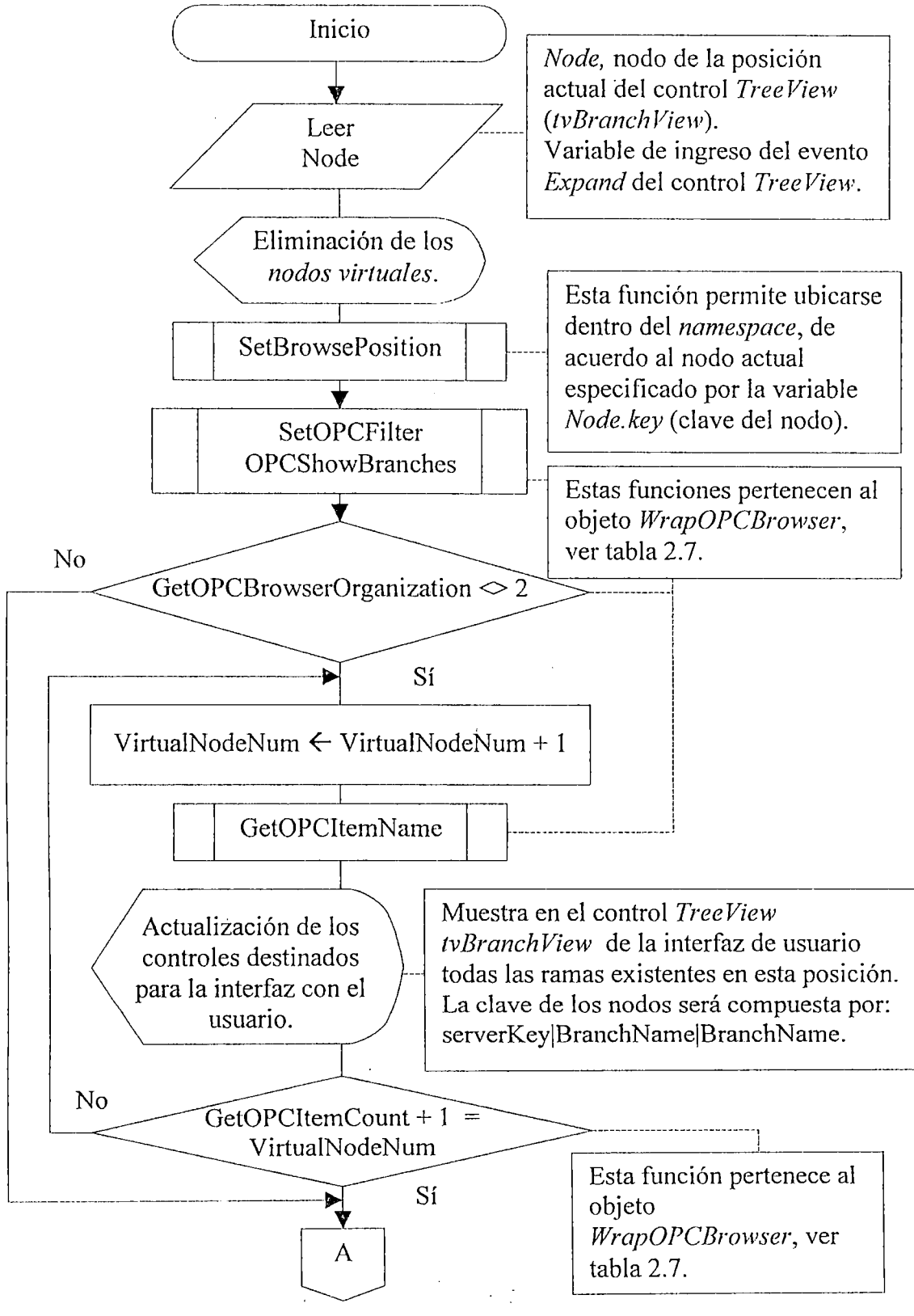


Figura 2.27 Algoritmo para la búsqueda dentro del *namespace* (*continúa...*)

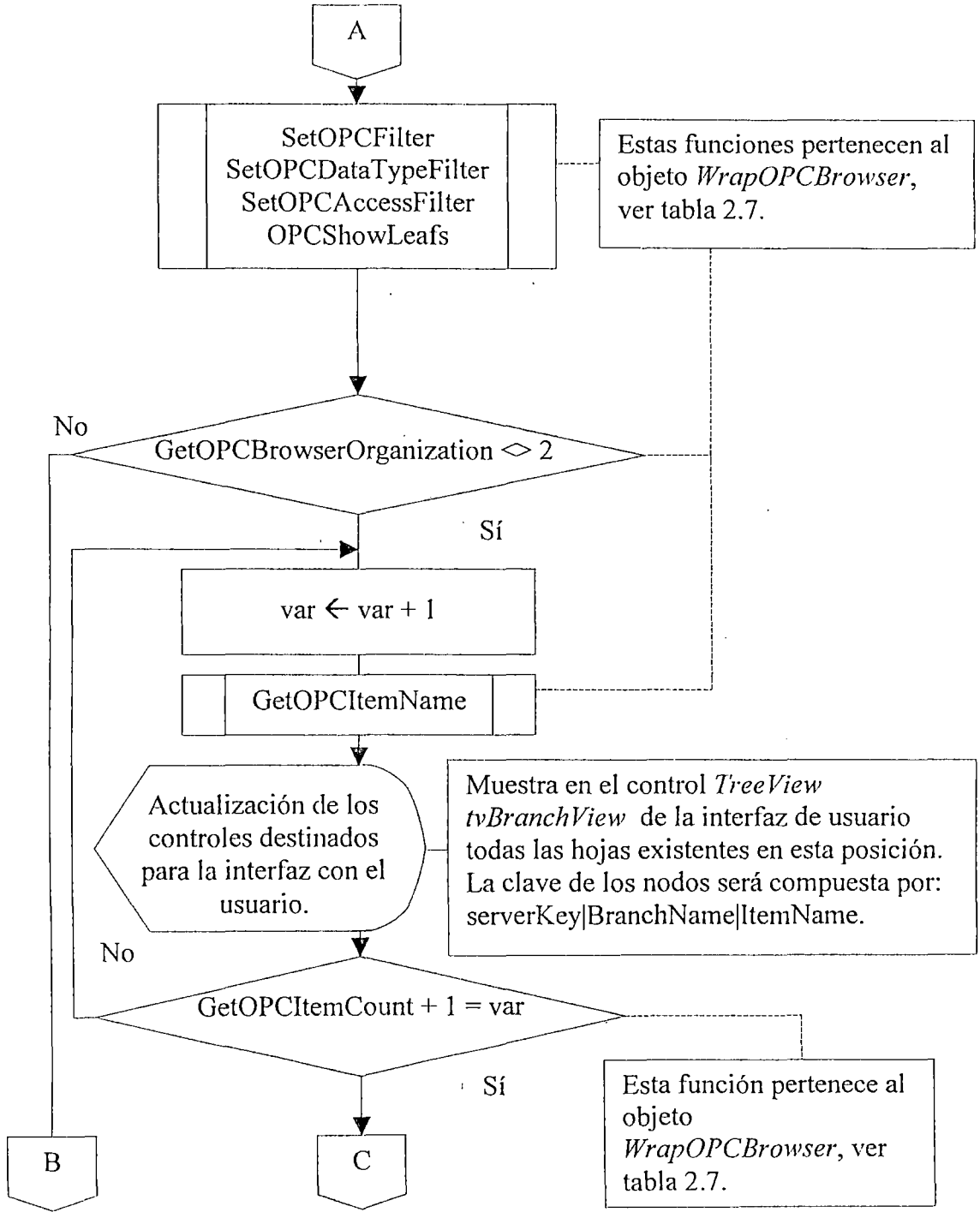


Figura 2.27 (*Continuación de la página 89.*) Algoritmo para la búsqueda dentro del *namespace* (*continúa...*)

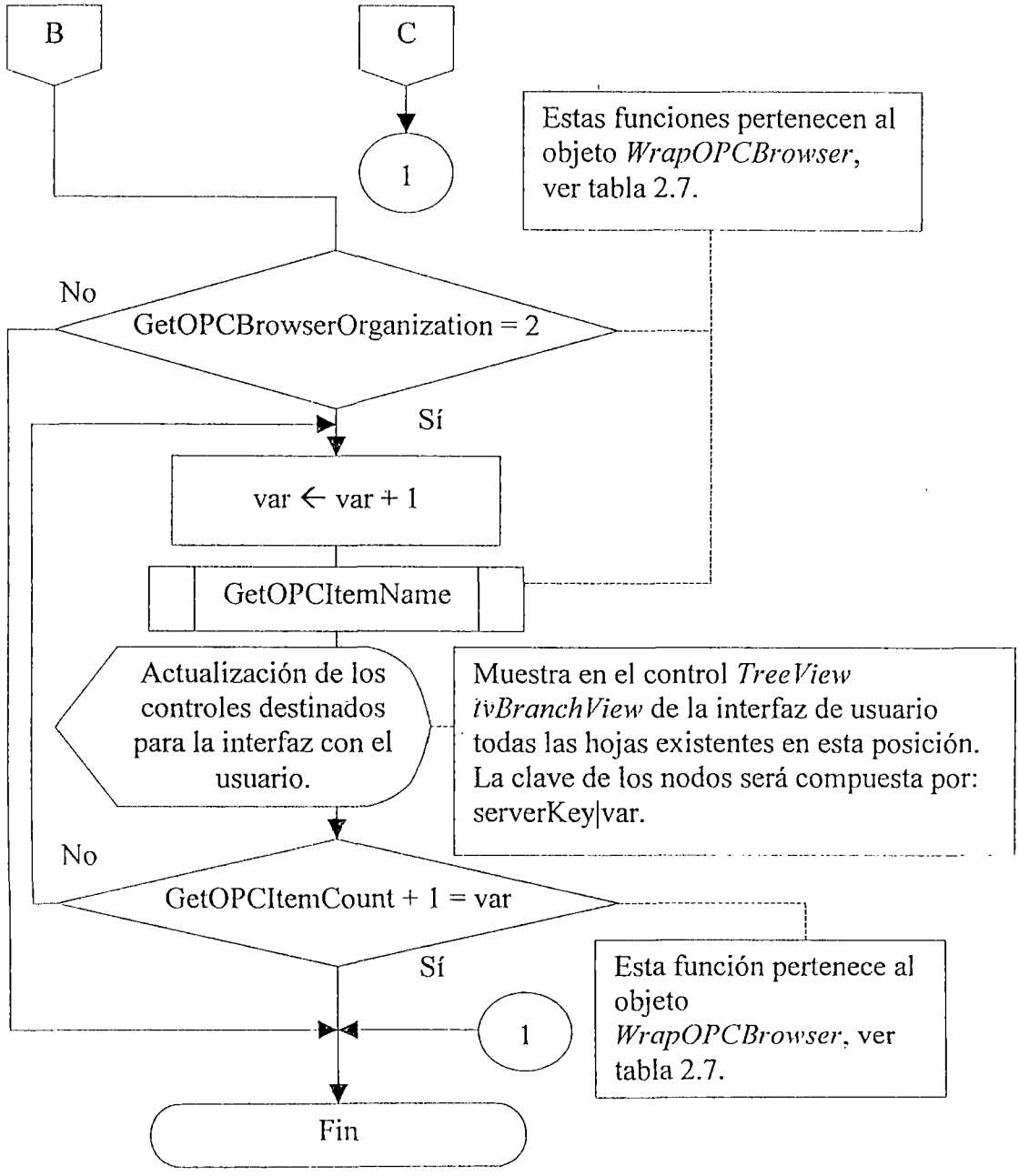


Figura 2.27 (*Continuación de la página 90.*) Algoritmo para la búsqueda dentro del *namespace*

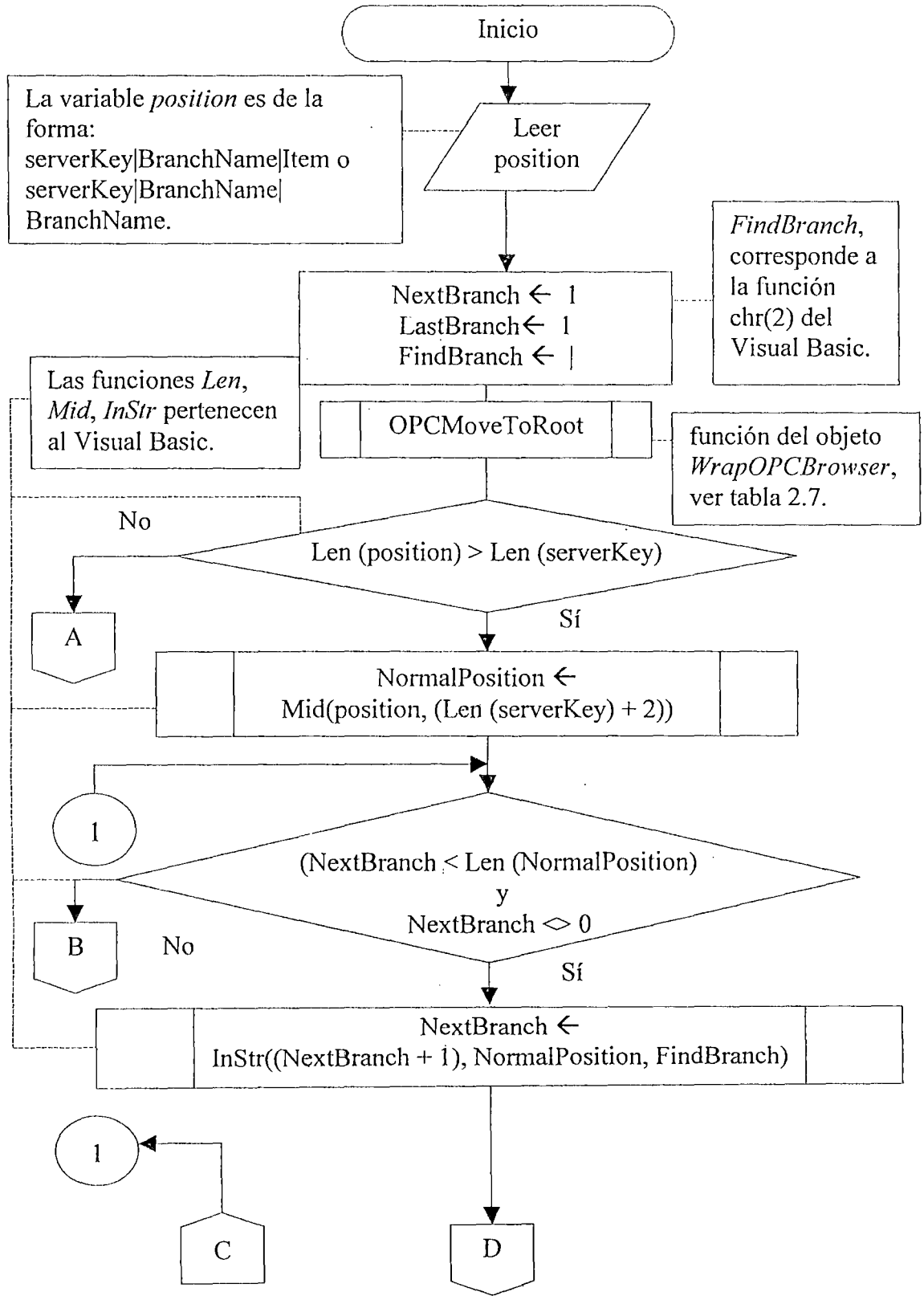


Figura 2.28 Algoritmo de la función *SetBrowsePosition,* para la ubicación específica dentro del *namespace* (*continúa...*)

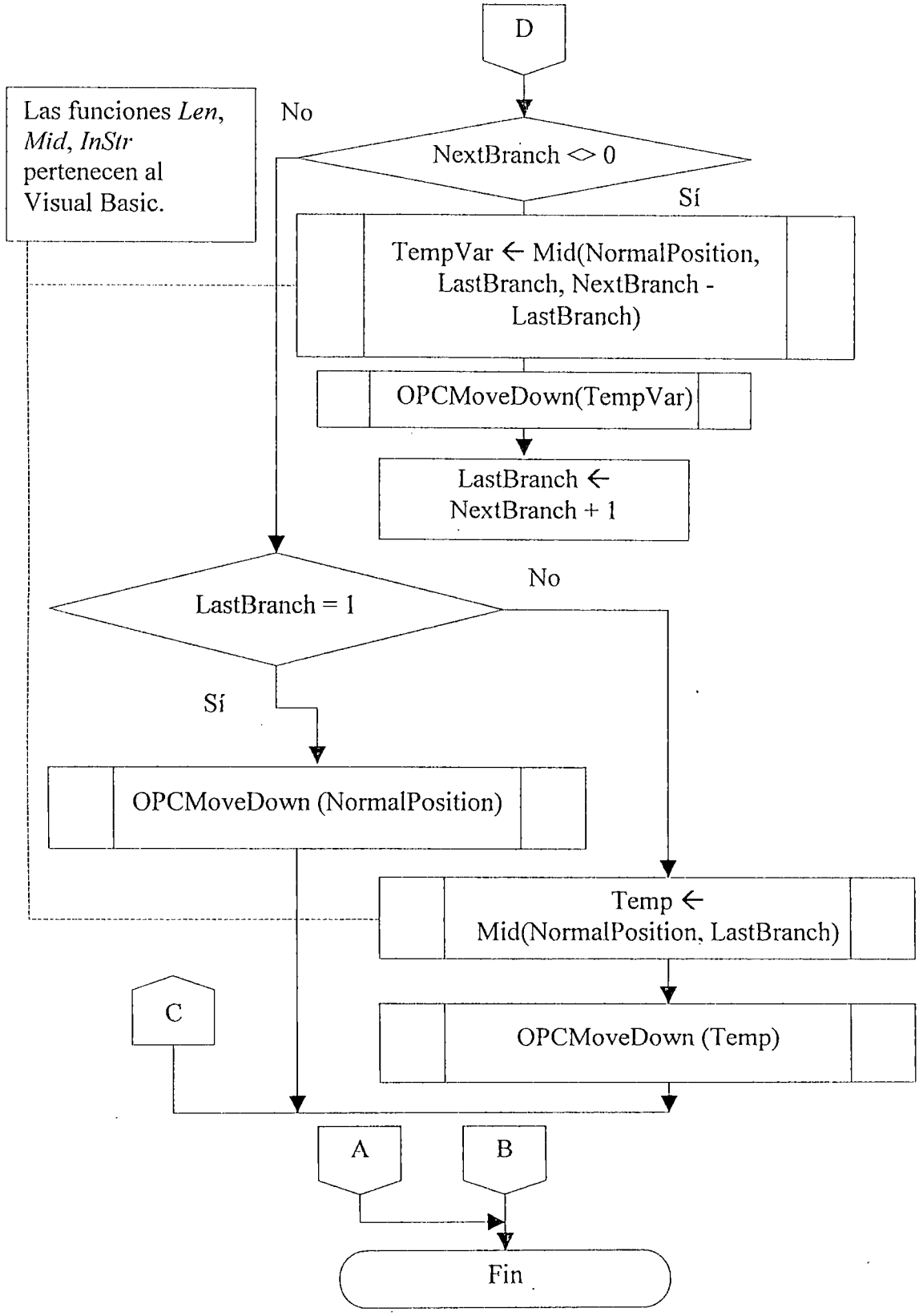


Figura 2.28 (*Continuación de la página 92.*) Algoritmo de la función *SetBrowsePosition*, para la ubicación específica dentro del *namespace*

2.1.7 Manejo de la Operación de Lectura Asíncrona

La lectura asíncrona se lleva a cabo por medio de dos etapas, los cuales son mostrados como algoritmos en las figuras 2.29 y 2.30

El algoritmo de la figura 2.29 corresponde a la función *AsyncReadOPCItem* del objeto *WrapOPCGroup*, esta función permite iniciar la operación de lectura asíncrona. Luego de iniciada esta operación el servidor de OPC notifica el término de esta transacción por medio del evento *AsyncReadComplete,* el algoritmo de este evento se encuentra en la figura 2.30. Estos dos procedimientos permiten el manejo de una operación de lectura asíncrona.

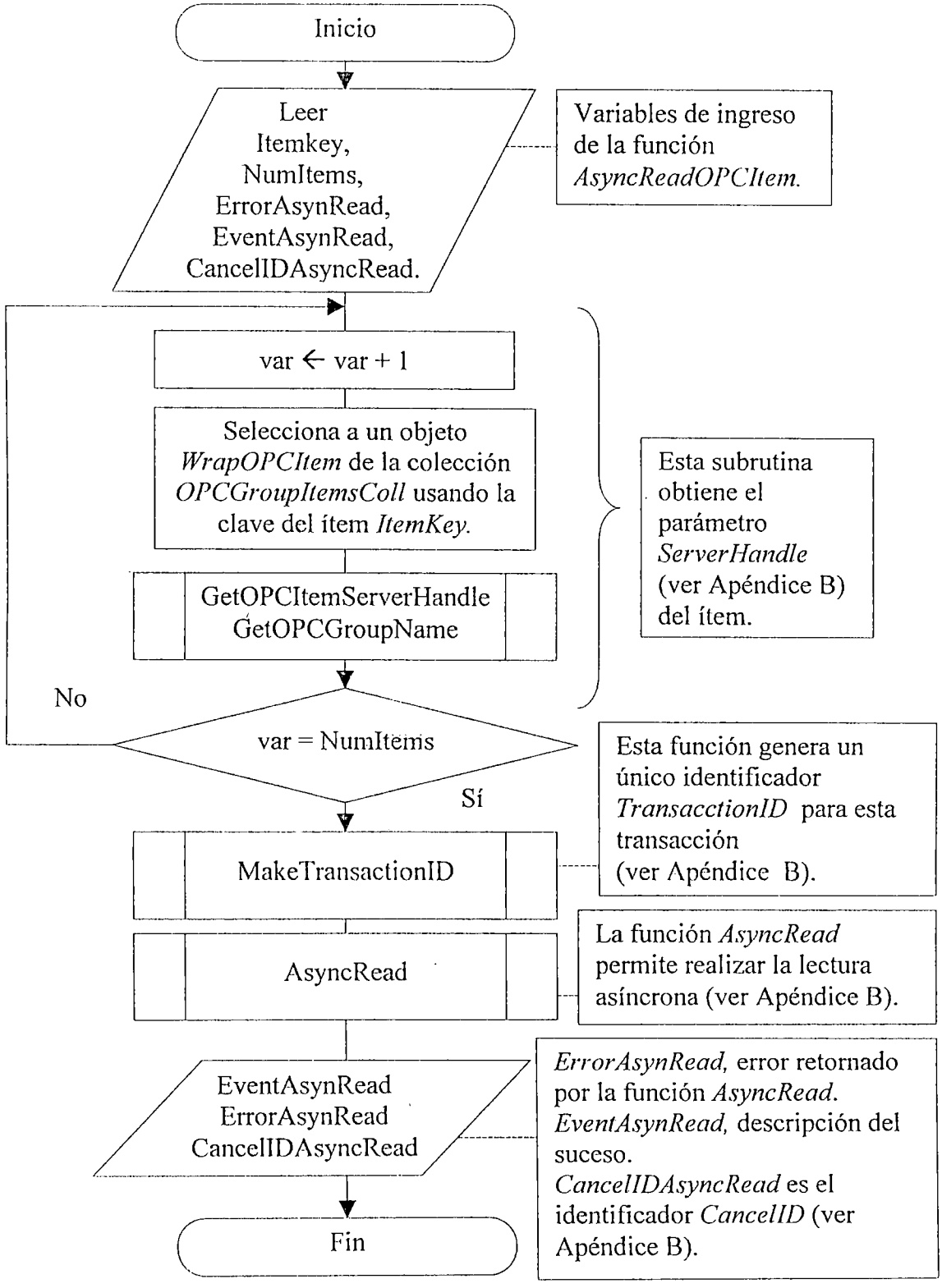


Figura 2.29 Algoritmo de la función *AsyncReadOPCItem,* para iniciar una operación de lectura asíncrona

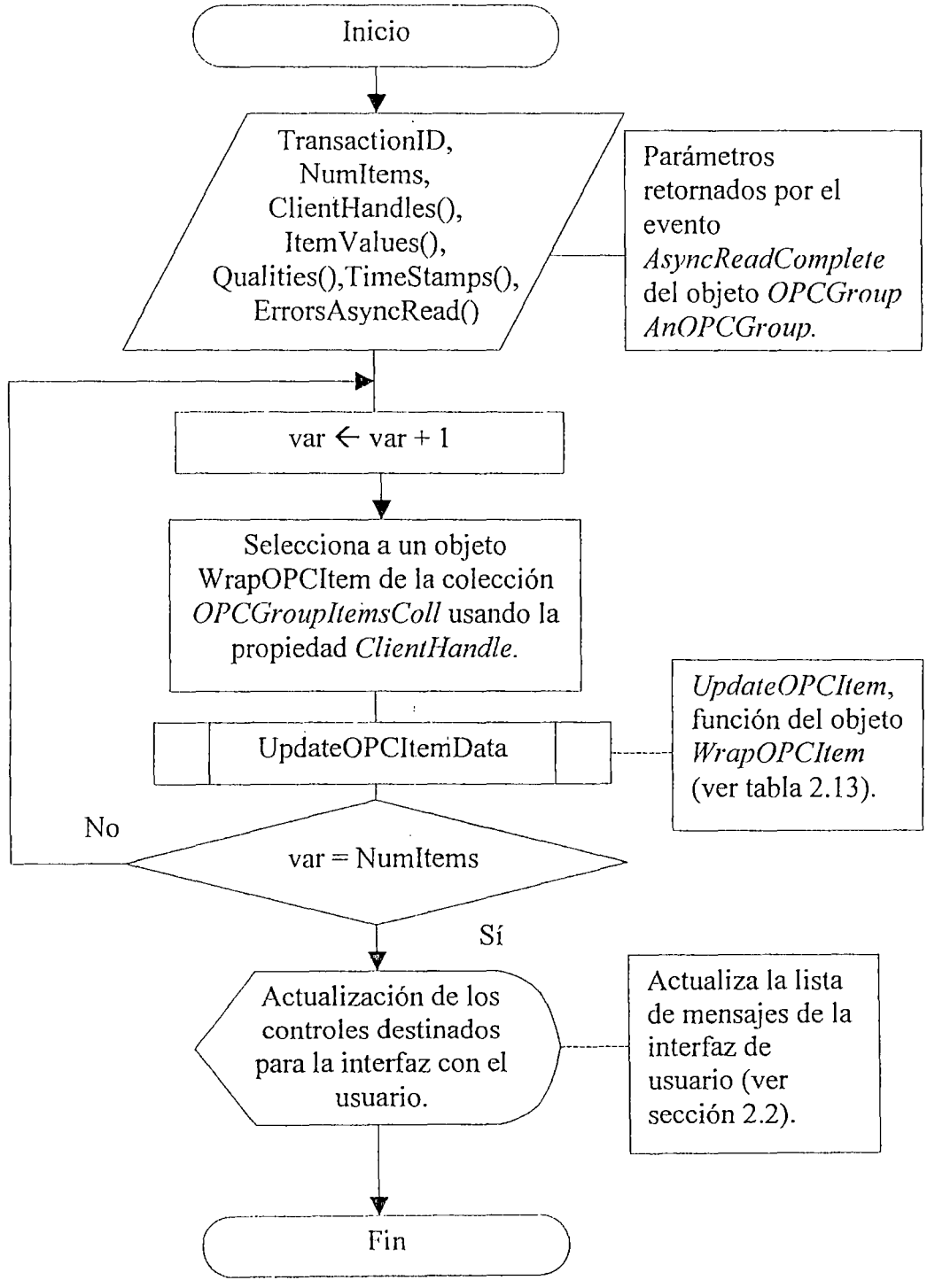


Figura 2.30 Algoritmo del evento *AsyncReadComplete,* para la notificación del término de una operación de lectura asíncrona

2.1.8 Manejo de la Operación *Refresh*

Esta operación consta de dos pasos, el primero es la función *OPCAsyncRefresh* del objeto *WrapOPCGroup*, mostrado en la figura 2.31, el cual inicializa la operación *Refresh.* Luego el servidor de OPC notifica el término de esta transacción por medio del evento *DataChange*. Este evento es mostrado en la figura 2.32.

El evento *DataChange* diferencia el tipo de operación por medio del parámetro *TransactionID* (ver Apéndice A), es decir para una notificación del cambio en el parámetro *Value* o *Quality* del ítem, el parámetro *TransactionID* es cero (ver figura 2.32). A diferencia del evento producido por la operación *Refresh* el cual depende del número del parámetro *TransactionID* generado por la función *MakeTransactionID* (ver figura 2.31).

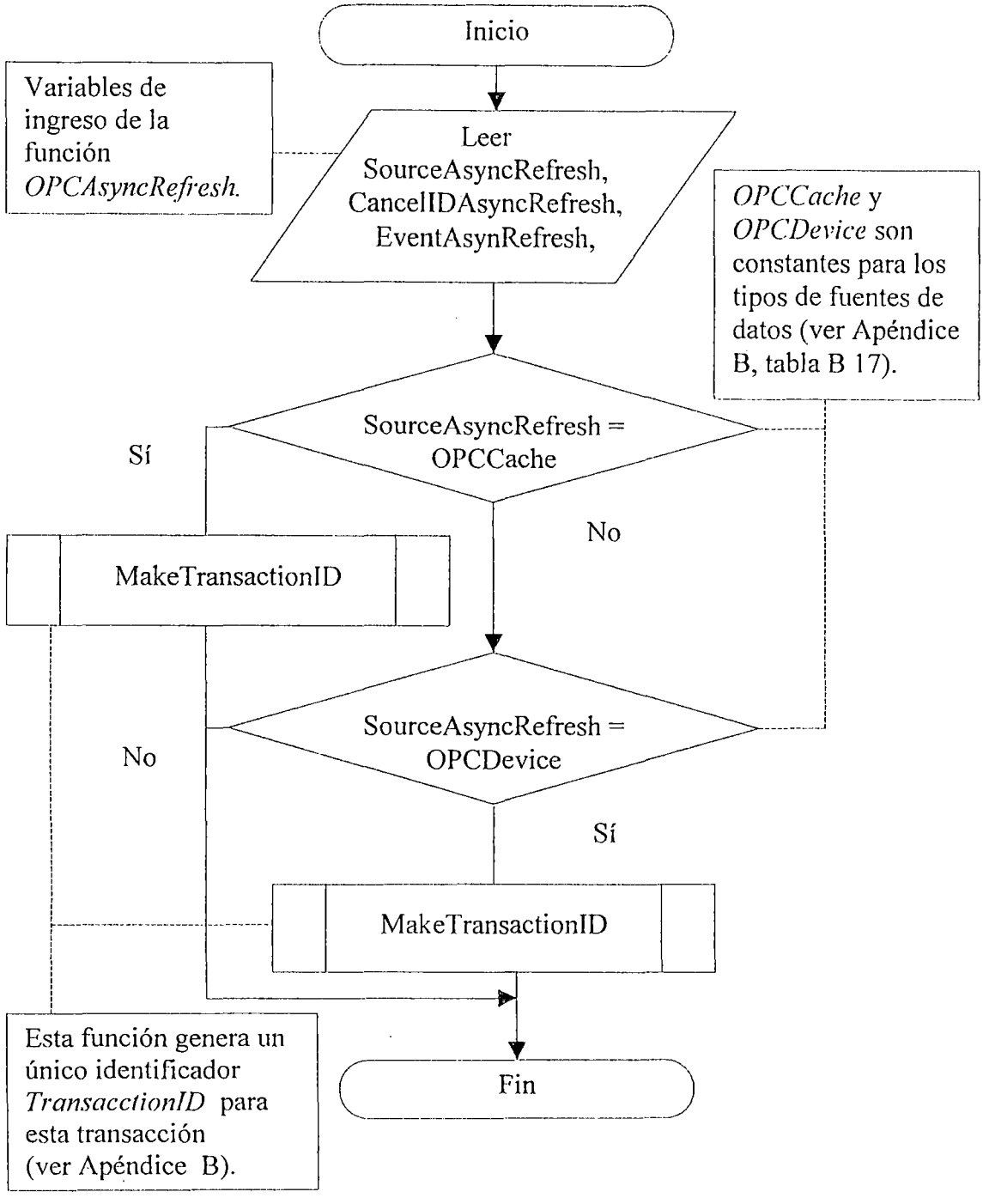


Figura 2.31 Algoritmo de la función *OPCAsyncRefresh,* para iniciar una operación de lectura asíncrona

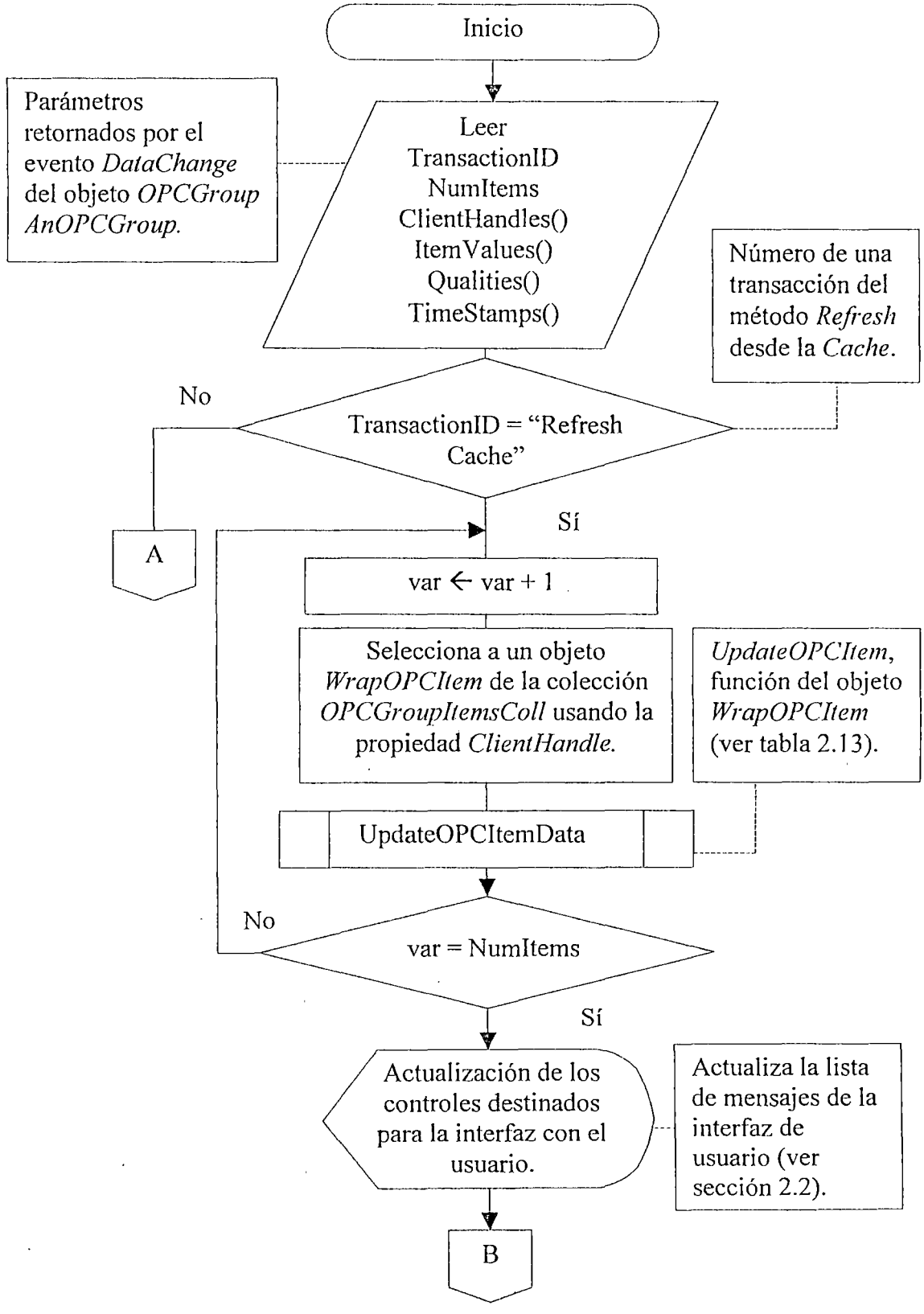


Figura 2.32 Algoritmo del evento *DataChange,* para las notificaciones desde el servidor de OPC (*continúa...*)

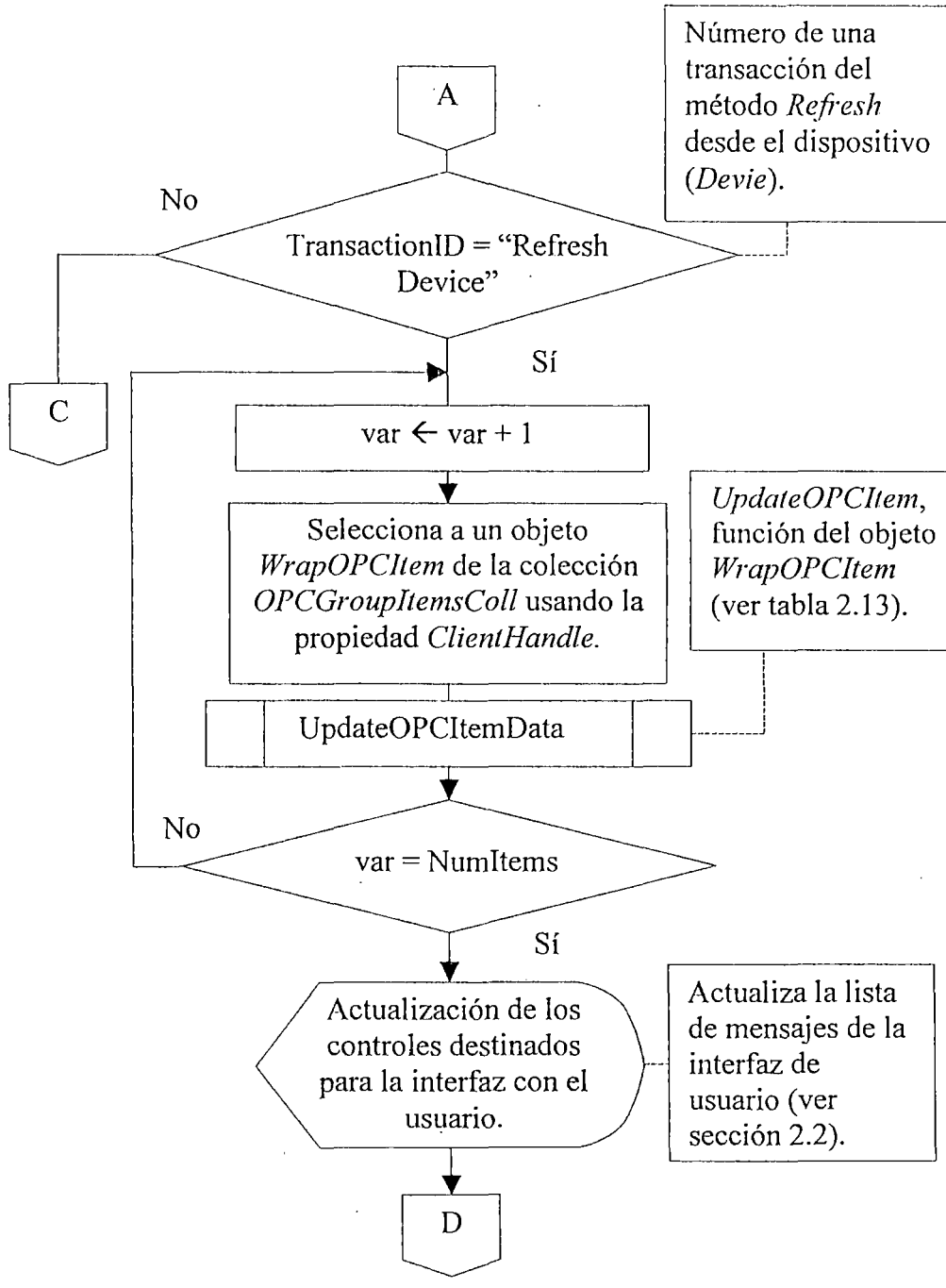


Figura 2.32 (*Continuación de la página 99.*) Algoritmo del evento *DataChange,* para las notificaciones desde el servidor de OPC (*continúa...*)

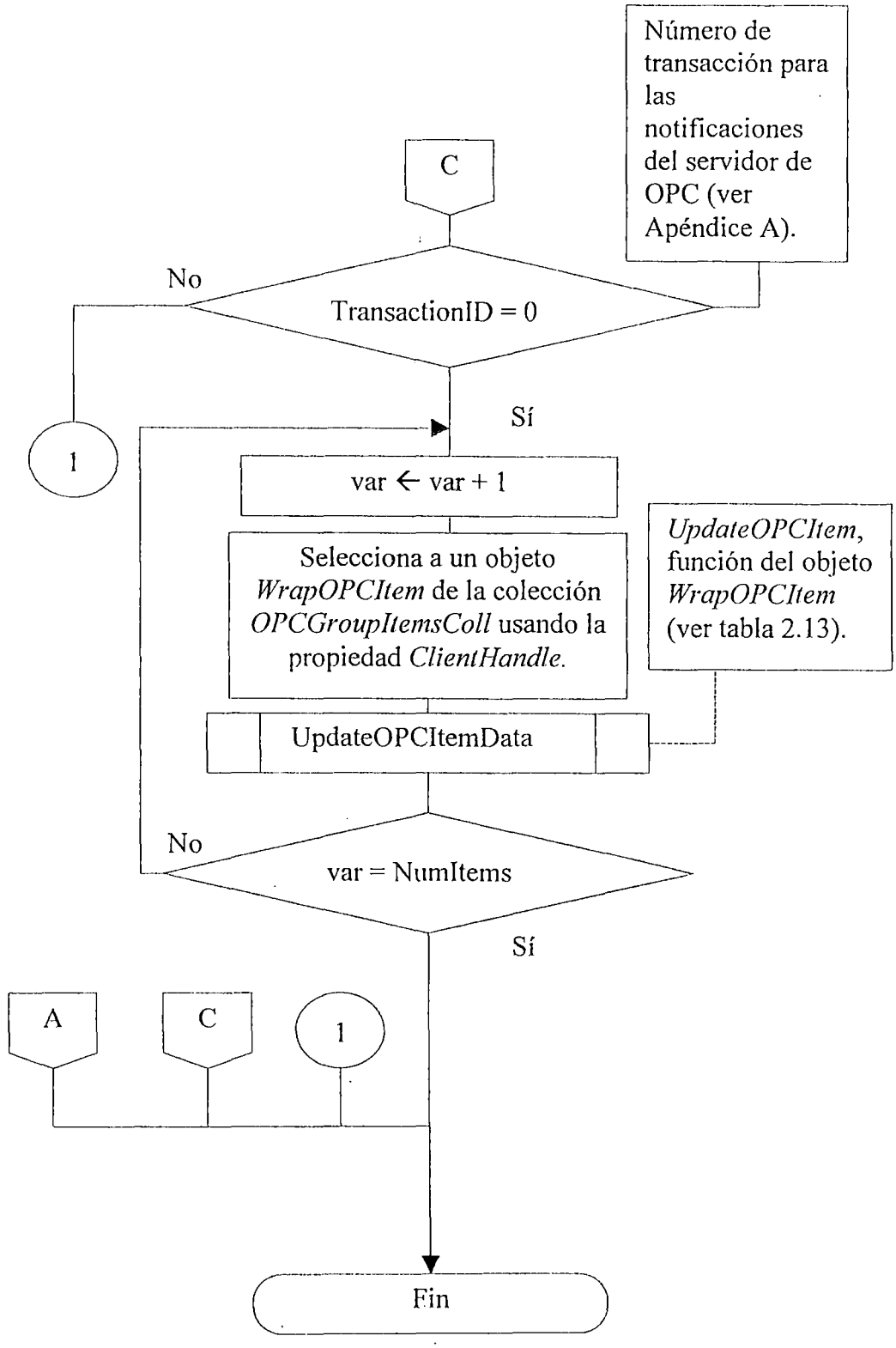


Figura 2.32 (*Continuación de la página 100.*) Algoritmo del evento *DataChange*, para las notificaciones desde el servidor de OPC

2.1.9 Manejo de una Operación de Escritura Asíncrona

El manejo de esta operación es llevado a cabo por medio de los algoritmos de las figuras 2.33 y 2.34. La función *OPCAsyncWrite* del objeto *WrapOPCGroup* es mostrada en la figura 2.33, esta función se encarga de inicializar la operación asíncrona. Luego el servidor de OPC se encarga de realizar la operación, para luego notificar por medio del evento *AsyncWriteComplete* del término de esta operación. El manejo de este evento es mostrado en la figura 2.34.

La utilización de estos dos procedimientos permiten realizar una operación de lectura asíncrona.

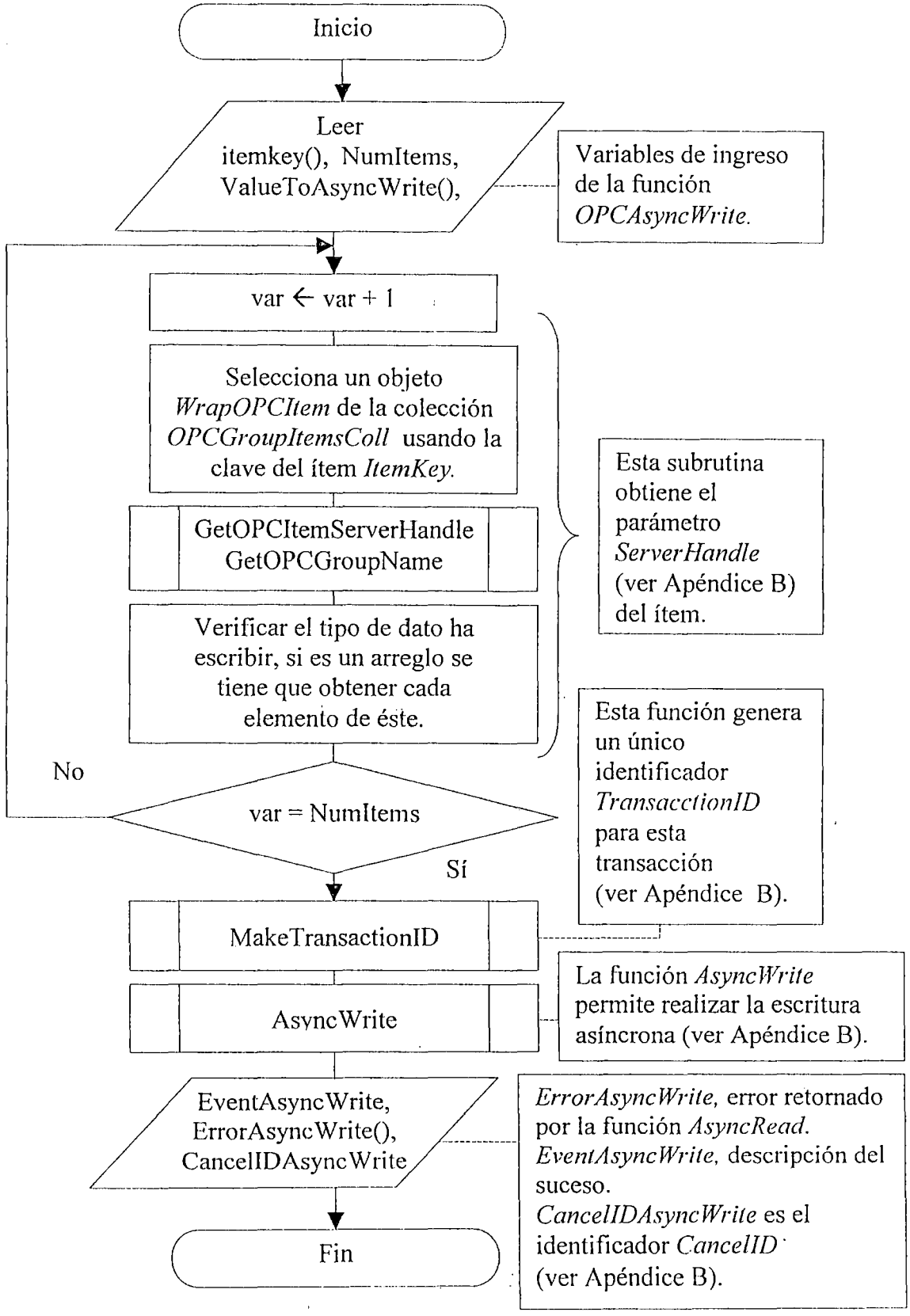


Figura 2.33 Algoritmo de la función *OPCAsyncWrite,* para iniciar una operación de escritura asíncrona

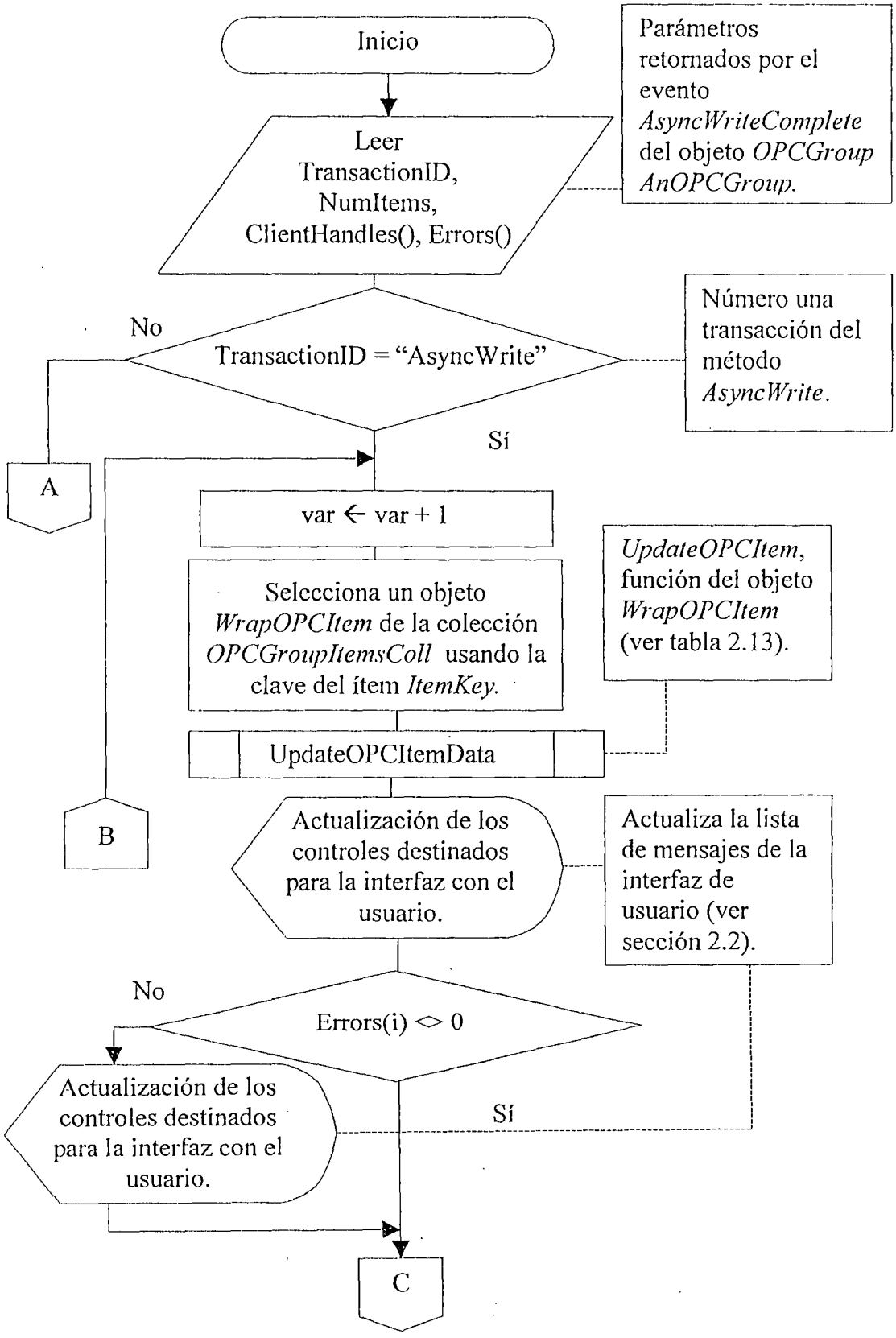


Figura 2.34 Algoritmo del evento *AsyncWriteComplete,* para la notificación del término de una operación de escritura asíncrona (*continúa...*)

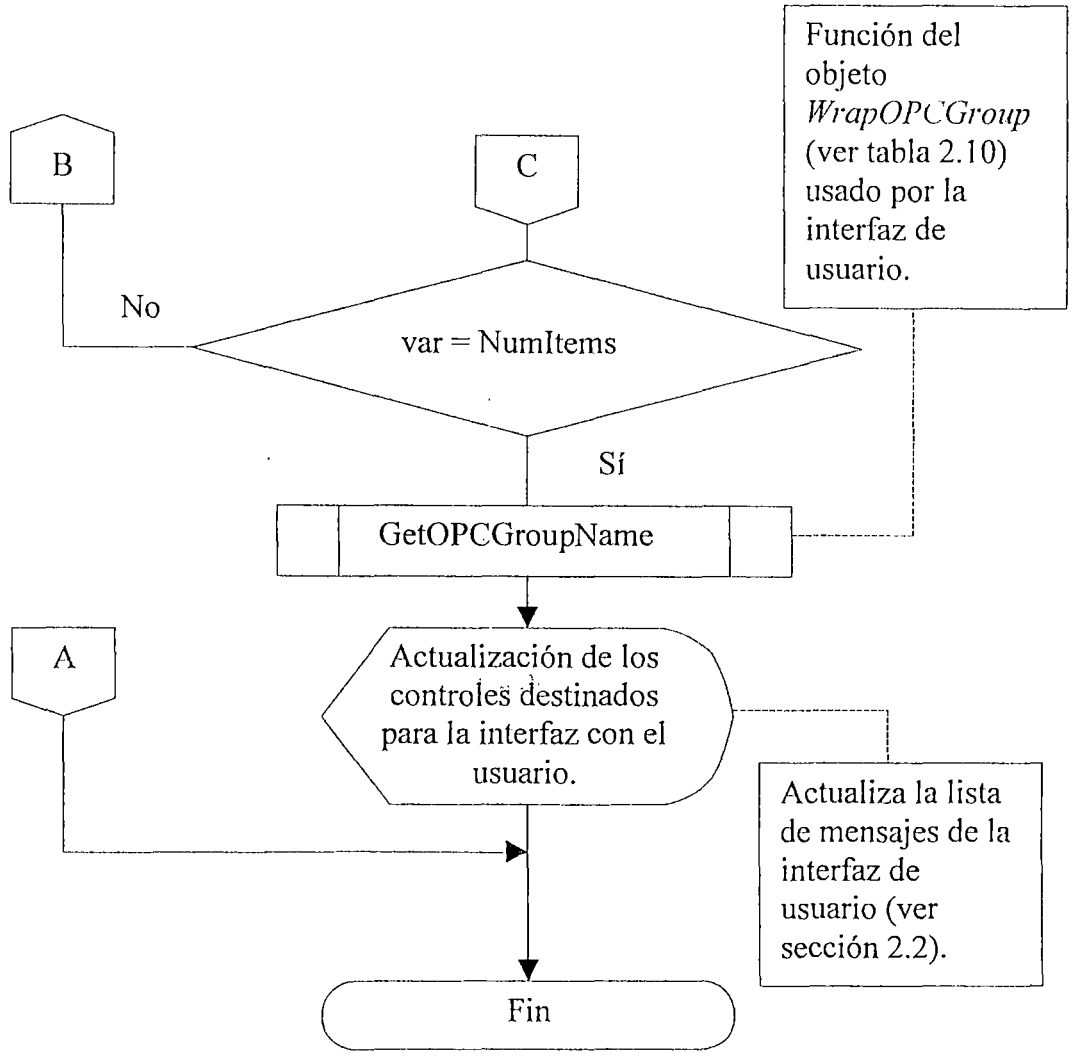


Figura 2.34 (*Continuación de la página 104.*) Algoritmo del evento *AsyncWriteComplete,* para la notificación del término de una operación de escritura asíncrona

2.1.10 Manejo de las Operaciones Síncronas

Las operaciones de lectura y escritura síncronas son mostradas en las figuras 2.35 y 2.36, respectivamente.

Estas operaciones tienen la particularidad de no retornar hasta que la operación haya finalizado, es decir los valores han sido aceptados o rechazados.

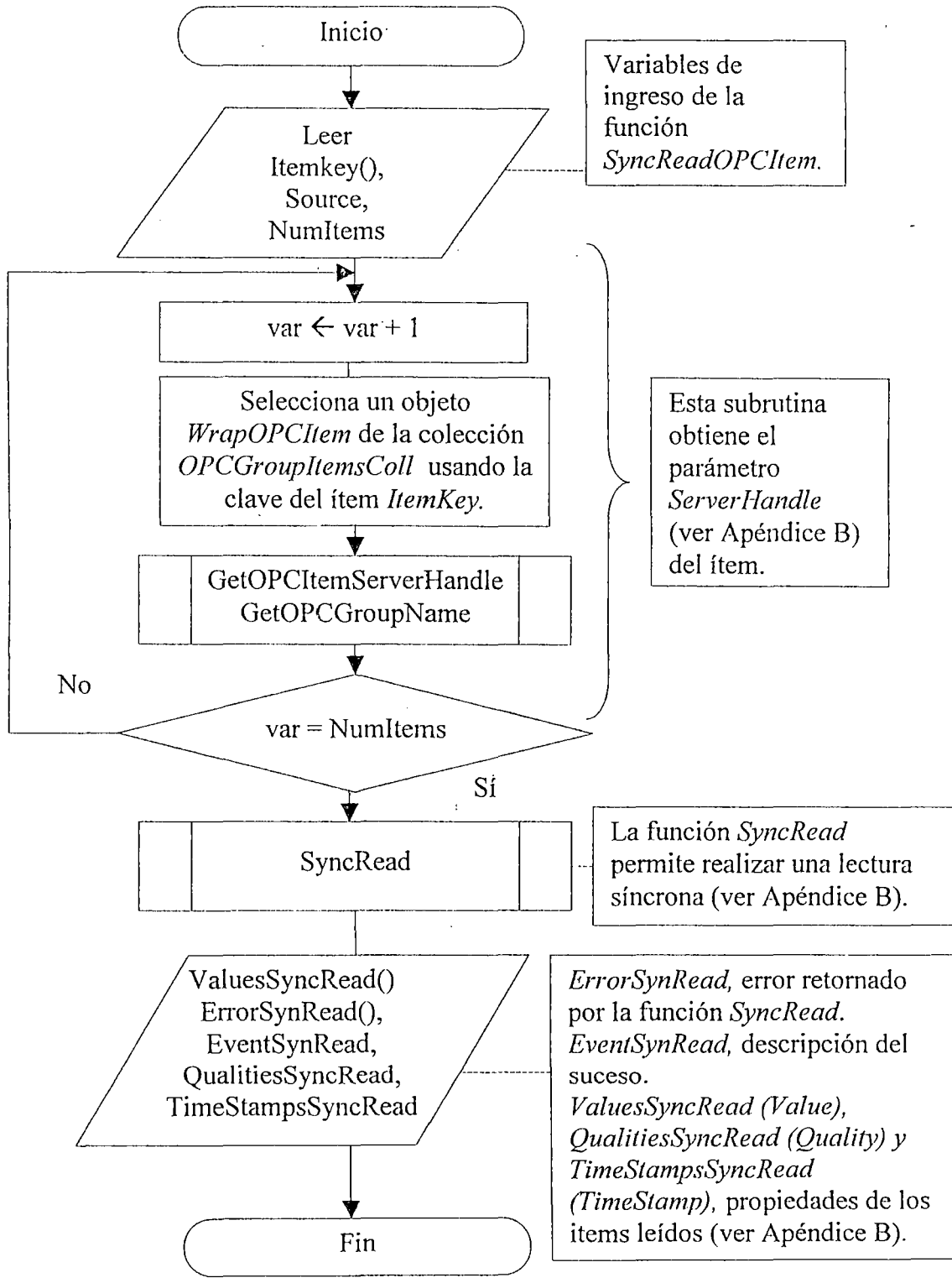


Figura 2.35 Algoritmo de la función *SyncReadOPCItem,* para una operación de lectura síncrona

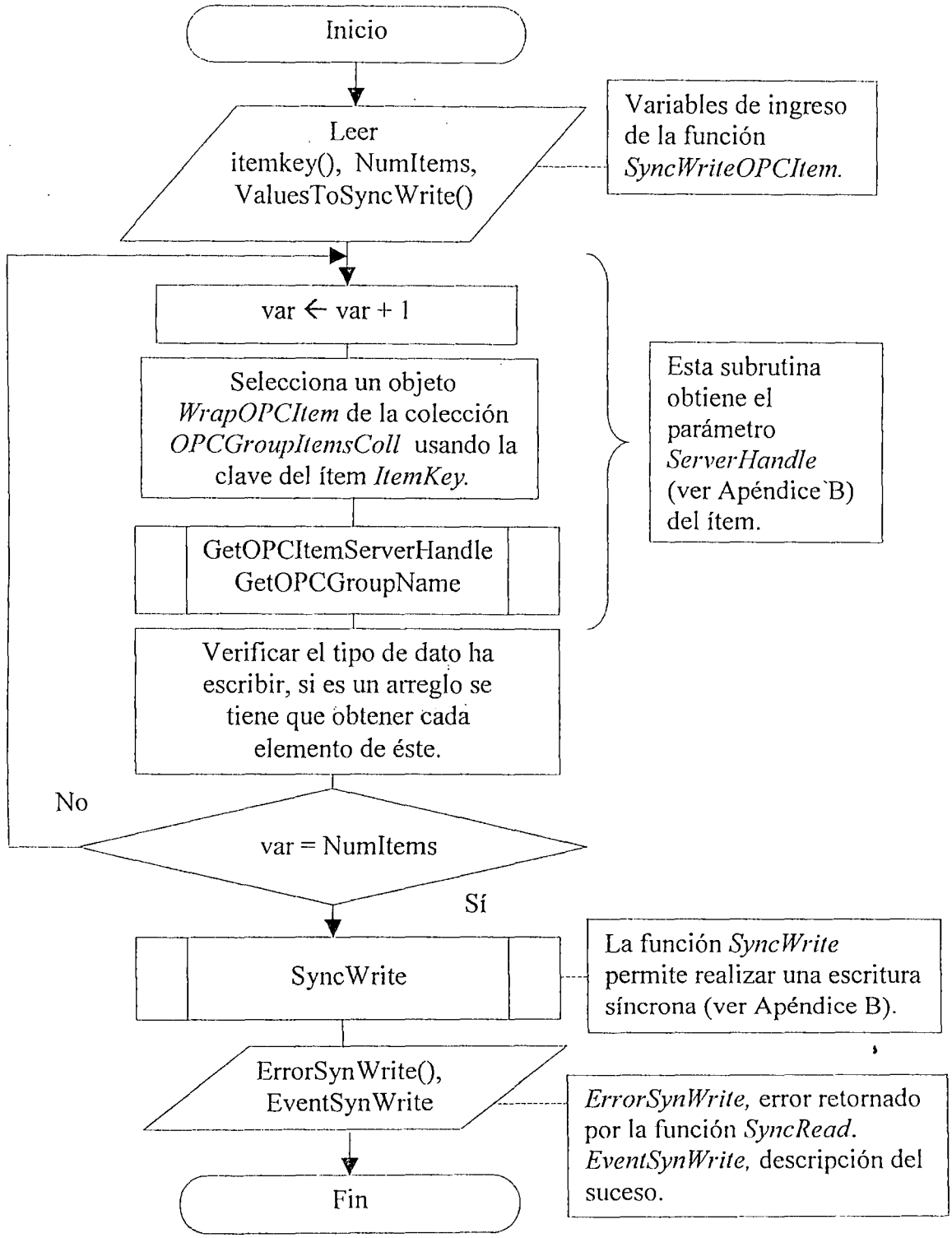


Figura 2.36 Algoritmo de la función *SyncWriteOPCItem,* para una operación de escritura síncrona

2.2 Diseño de la Interfaz de Usuario

El diseño de la interfaz de usuario fue realizado tomando en cuenta la facilidad para el acceso con el usuario, es decir éste debe ser muy simple de usar y muy flexible en el manejo. Por lo tanto se diseño un formulario principal compuesta por tres controles principales, los cuales se encargan de visualizar y controlar la funcionalidad de OPC, estos controles son:

- El control TreeView "tvTreeView"
- El control ListView "lvListViewH"
- El control TabStrip "tbTabStrip"
    - El control TreeView "OPCServerListTreeView"
    - El control TreeView "tvBranchView"
    - El control ListView "lvListView"

2.2.1 El Control *TreeView tvTreeView*

Este control es usado para interactuar de manera gráfica con todas las conexiones existentes, es decir con un servidor, un grupo o un ítem de OPC.

Para la creación de los nodos de cada conexión (servidor, grupo o ítem) de este control, se usa como clave la misma utilizada tanto para el objeto *WrapOPCServer, WrapOPCGroup* y *WrapOPCItem,* respectivamente.

2.2.2 El Control *ListView lvListViewH*

Este control es usado para mostrar la descripción de todos los eventos y errores producidos durante la aplicación de la funcionalidad de OPC.

2.2.3 El Control *TabStrip tbTabStrip*

Este control permite interactuar con tres controles diferente a la vez, ya que permite escogerlos por medio de una etiqueta. Estos controles son:

2.2.3.1 El Control *TreeView OPCServerListTreeView*

Este control permite poder escoger los servidores presente en la máquina local.

2.2.3.2 El Control *TreeView tvBranchView*

Este control permite tener toda la funcionalidad del objeto *OPCBrowser*, es decir se puede obtener la configuración del *namespace* del servidor de OPC en forma gráfica.

Para la creación de las claves de los nodos de este control se considera lo siguiente:

- Nodo Raíz: Clave del servidor
- Nodo de una Rama:

  Clave del servidor|Nombre de la rama padre|Nombre de la rama padre| ...

  |Nombre de la rama
- Nodo de la hoja o ítem:

  Clave del servidor|Nombre de la rama padre|Nombre de la rama padre| ...

  |Nombre de la hoja o ítem

El carácter "|" corresponde a la función chr(2) del Visual Basic.

2.2.3.3 El Control *ListView lvListView*

Con este control se obtiene la funcionalidad de los objetos *OPCGroup* y *OPCItem*, es decir permitirá ver las propiedades más importantes del ítem como: Valor, calidad y tiempo de actualización de los items de un grupo. Además permite tener todos los modos de comunicación habilitados por OPC.

Además del formulario principal se han creado varios formularios, cuyo objetivo es brindar una interfaz de usuario para cada funcionalidad de OPC. Las características de cada formulario creado son mostradas en la tabla 2.14

Formulario	Descripción
frmTagnameDictionary	Permite seleccionar items, para ser mostrados gráficamente.
frmSelectOPCServer	Permite seleccionar, ver y adicionar a la configuración principal los servidores presentes en la máquina local o remota.
frmRealTimeTrendConfiguration	Permite configurar los parámetros gráficos
frmRealTimeTrend	Permite ver en forma gráfica los items y parámetros configurados previamente por el formulario *frmRealTimeTrendConfiguration.*
frmOPCServerProperties	Permite ver las propiedades del servidor de OPC.
frmOPCGroupProperties	Permite ver las propiedades del grupo de OPC.
FrmMain	Permite controlar, ver y modificar la funcionalidad de OPC
frmItemUpdateInterval	Permite ver y modificar la frecuencia de actualización de los datos de los items en el formulario principal.
FrmItemFilter	Permite configurar los parámetros de los filtros que se aplicarán en la búsqueda dentro del *namespace.*
frmGetErrorString	Permite obtener los errores de OPC.
frmAsyncSyncWrite	Permite realizar las dos formas de escritura síncrona y asíncrona.
FrmAddItem	Permite adicionar un ítem a un grupo de OPC.
FrmAddGroup	Permite adicionar un grupo a un servidor de OPC.
FrmAbout	Muestra el nombre del desarrollador y la versión del software.

Tabla 2.14 Descripción de los formularios diseñados para el software OPC Client

Tomando en cuenta los objetos diseñados para el software OPC Client (ver figura 2.1), se ha desarrollado la funcionalidad de OPC para Excel. La única diferencia es la interfaz de usuario que va ha utilizar, pero la estructura será la misma.

CAPÍTULO III

INTERFAZ DE USUARIO DEL SOFTWARE OPC CLIENT

El software cliente diseñado e implementado en el capítulo anterior provee una interfaz de usuario muy amigable, el cual permitirá interactuar con un servidor de OPC de una manera fácil y rápida. Por tal razón es importante saber como funciona esta interfaz de usuario, este capítulo trata este tema y además muestra la funcionalidad que ofrece OPC para el acceso de datos implementado en el software OPC Client.

3.1 Ventana Principal

La figura 3.1 muestra la ventana principal del software, el cual permite tener un acceso fácil a toda la funcionalidad del acceso de datos de OPC.

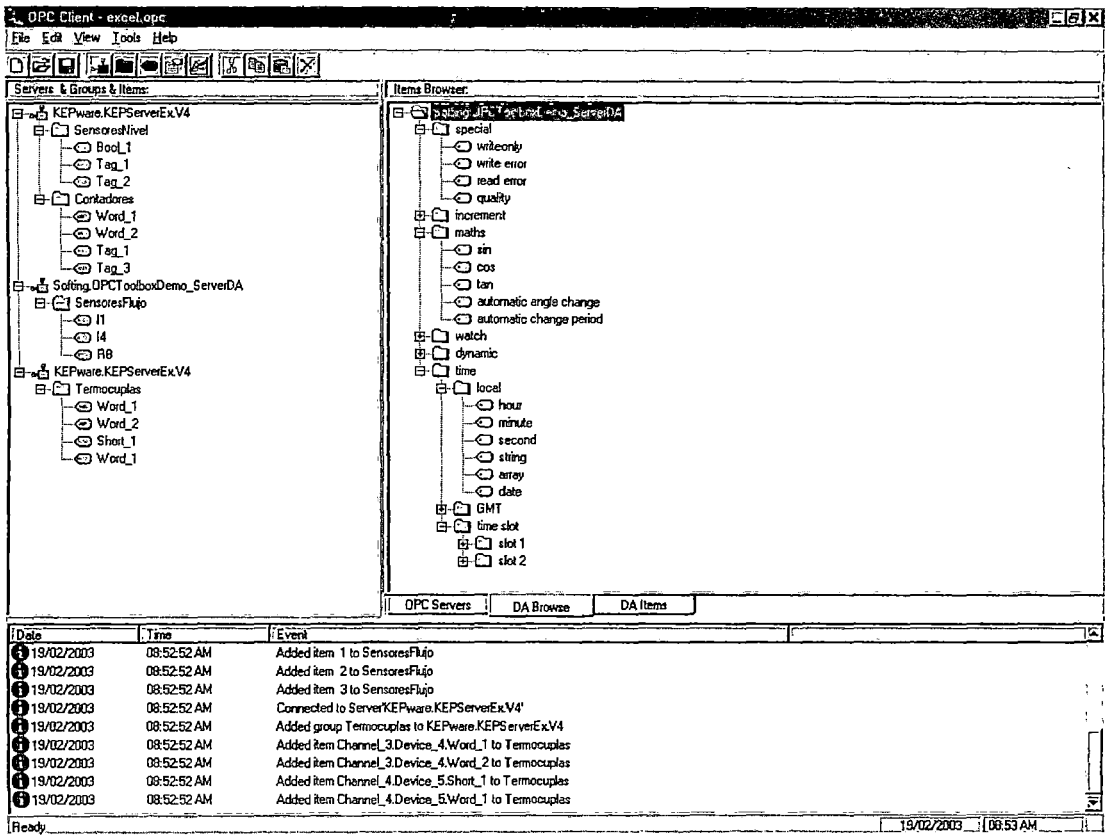


Figura 3.1.Ventana principal del software OPC Client

La ventana principal de la figura 3.1 presenta las siguientes partes:

- Barra de Título
- Menú Principal
- Barra de Herramientas

- Explorador de Conexiones
- Fichero
- Lista de Mensajes

3.1.1 Barra de Título

La aplicación mostrará una **Barra de Título** como se muestra en la figura 3.1, el cual consta del nombre del Software "OPC Client" seguido por el nombre del archivo del proyecto abierto.

3.1.2 Menú Principal

En este menú se puede encontrar los siguientes submenús:

- **File**

  El cual consta de:

  - **New** Crea un nuevo proyecto.
  - **Open** Abre un nuevo proyecto ya existente desde el disco.
  - **Save** Guarda el proyecto actual al disco.

- **Save As** Guarda el proyecto actual al disco.
- **Exit** Termina la aplicación.

- **Edit**

El cual consta de:

- **New Server Connection** Crea una nueva conexión con un Servidor de OPC (objeto *OPCServer*).
- **New Group** Crea un nuevo grupo (objeto *OPCGroup*).
- **New Item** Crea un nuevo ítem (objeto *OPCItem*).
- **Cut** Corta el elemento (servidor, grupo o ítem) actualmente seleccionado.
- **Copy** Copia el elemento (servidor, grupo o ítem) actualmente seleccionado.
- **Paste** Pega el elemento (servidor, grupo o ítem) actualmente seleccionado.
- **Delete** Elimina el elemento (servidor, grupo o ítem) actualmente seleccionado.
- **Properties** Muestra las propiedades del servidor o grupo seleccionado.

- **View**

El cual consta de:

- **Tag Dictionary** Adiciona el ítem seleccionado a un grupo de *Tags*, que luego se podrán mostrar gráficamente por medio de la ventana **Real Time Trend.**
- **Real Time Trend** Muestra en forma gráfica los items adicionados al **Tag Dictionary**.
- **Item Display Update Interval** Cambia el tiempo de actualización de los datos que se mostrarán en la pantalla.
- **Excel** Carga el Software Microsoft Excel con la funcionalidad de OPC.
- **Clear Messages** Borra todos los mensajes mostrados en la **Lista de Mensajes**.
- **Status Bar** Habilita o deshabilita la visualización de la barra de estado (**Status Bar**).

- **Tools**

- **Server**

- **Get Error String** Obtiene la respectiva descripción de un código de error, especificado por OPC o el servidor.

- **Item**

  - **Item Filter** Filtra los items que se mostrarán en el *namespace* (pestaña **DA Browser** del **fichero** de la ventana principal).
  - **Set Active** Activa el ítem seleccionado.
  - **Set Inactive** Desactiva el ítem seleccionado
  - **Synchronous Cache Read** Realiza la lectura en forma síncrona desde la *Cache* del ítem o items seleccionados.
  - **Synchronous Device Read** Realiza la lectura en forma síncrona desde el dispositivo (*device*) del ítem o items seleccionados.
  - **Synchronous Write** Realiza la escritura en forma síncrona del ítem o items seleccionados.
  - **Asynchronous Read** Realiza la lectura asíncrona del ítem o items seleccionados.
  - **Asynchronous Cahe Refresh** Actualiza los datos de todos los items pertenecientes a un grupo seleccionado desde la *cache*.

- **Asynchronous Device Refresh** Actualiza los datos de todos los items pertenecientes a un grupo seleccionado desde el dispositivo (*device*).
- **Asynchronous Write** Realiza la escritura asíncrona del ítem o items seleccionados.
- **Asynchronous Cancel** Cancela una transacción.
- **Delete** Elimina el ítem seleccionado.

- **Help**

- **About OPC Client** Muestra el nombre del desarrollador del software y la versión de la misma.

3.1.3 Barra de Herramientas

Esta barra ubicada en la parte superior de la ventana principal está compuesta por 10 botones, los cuales tienen las siguientes funciones:

- **New** Crea un nuevo proyecto.
- **Open** Abre un proyecto ya existente desde el disco.
- **Save** Guarda el proyecto actual al disco.

-  **New Server** Crea una nueva conexión con un servidor.
-  **New Group** Crea un nuevo grupo.
-  **New Item** Crea un nuevo ítem.
-  **Properties** Muestra las propiedades del servidor o grupo seleccionado.
-  **Trend** Muestra en forma gráfica los items adicionados al **Tag Dictionary**.
-  **Cut** Corta el elemento (servidor, grupo o ítem) actualmente seleccionado.
-  **Copy** Copia el elemento (servidor, grupo o ítem) actualmente seleccionado.
-  **Paste** Pega el elemento (servidor, grupo o ítem) actualmente seleccionado.
-  **Delete** Elimina el elemento (servidor, grupo o ítem) actualmente seleccionado.

3.1.4 Explorador de Conexiones

En el extremo superior izquierdo de la figura 3.1 y en la figura 3.2 se puede apreciar el explorador de conexiones. El cual permite tener una visión general de todas las conexiones existentes, ya sea con los servidores, grupos e items creados. Además, permite seleccionar de una menara fácil una conexión existente.

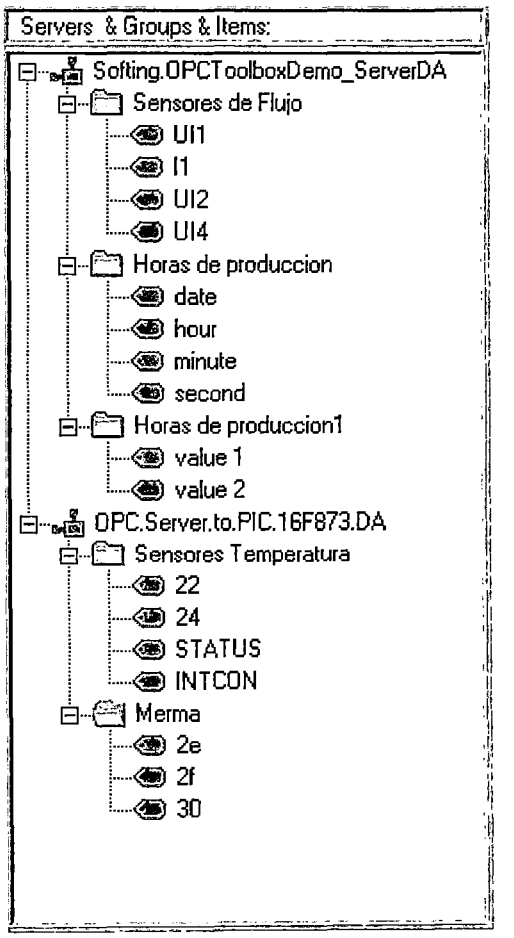


Figura 3.2 *Explorador de Conexiones*

3.1.5 Fichero

Este control se encuentra ubicada en el extremo superior derecho de la figura 3.1, en donde se pueden apreciar tres pestañas los cuales son:

- **OPC Servers**

  Este control ofrece una lista desplegable de todos los servidores de OPC que se encuentran registrados en la máquina local (ver figura 3.3).

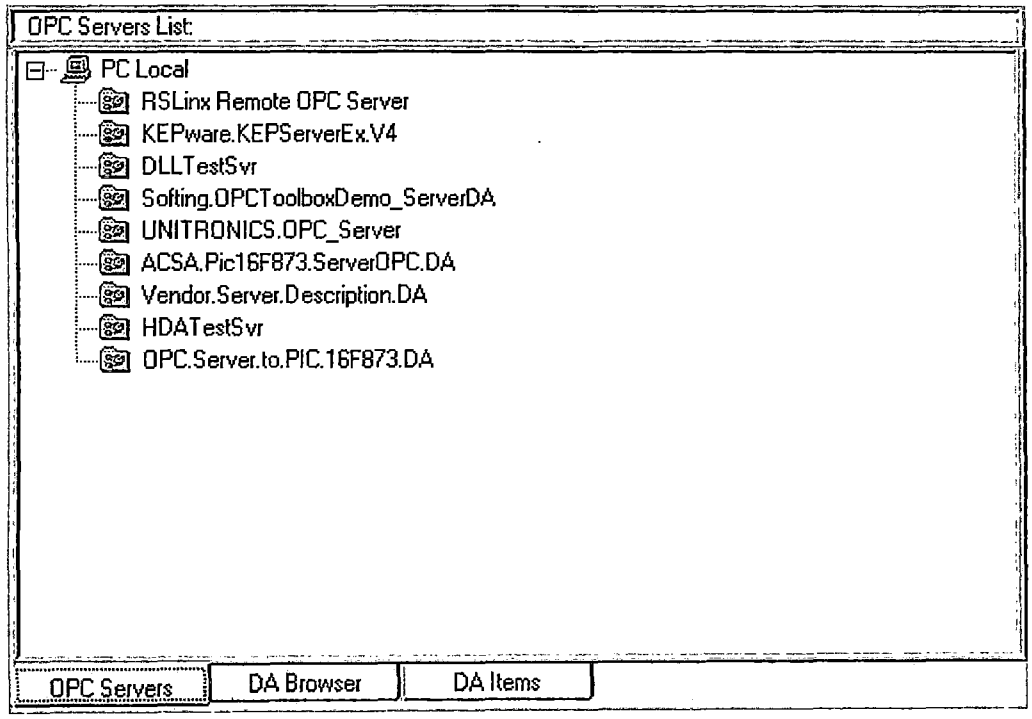


Figura 3.3 Pestaña *OPC Servers*

- **DA Browser**

  Este control ofrece toda la funcionalidad del objeto OPCBrowser de OPC, es decir permite visualizar todo el *namespace* del servidor seleccionado. Permitiendo encontrar el ítem deseado, que luego puede ser adicionado a un grupo (ver figura 3.4).

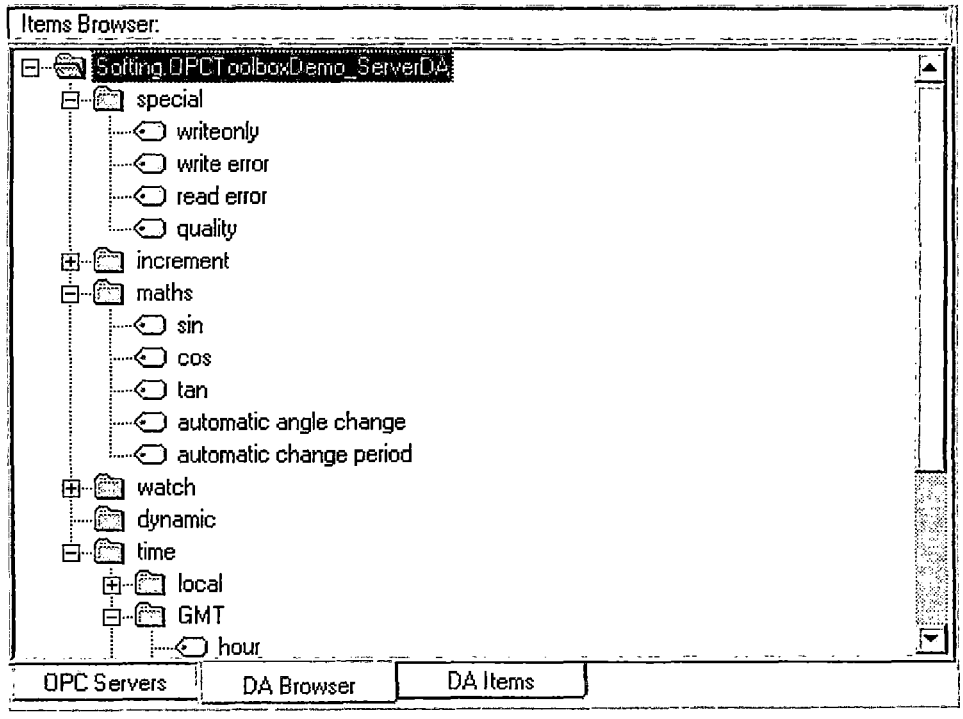


Figura 3.4 Pestaña *DA Browser*

- **DA Items**

  Tal como se aprecia en la figura 3.5 este control ofrece una lista de items pertenecientes a un grupo seleccionado, esta lista de items consta de las siguientes propiedades: identificador del ítem (*ItemID*), el valor del ítem (*Value*), tipo de dato (*DataType*), tiempo de actualización (*Timestamp*) y estado del dato (*Status*).

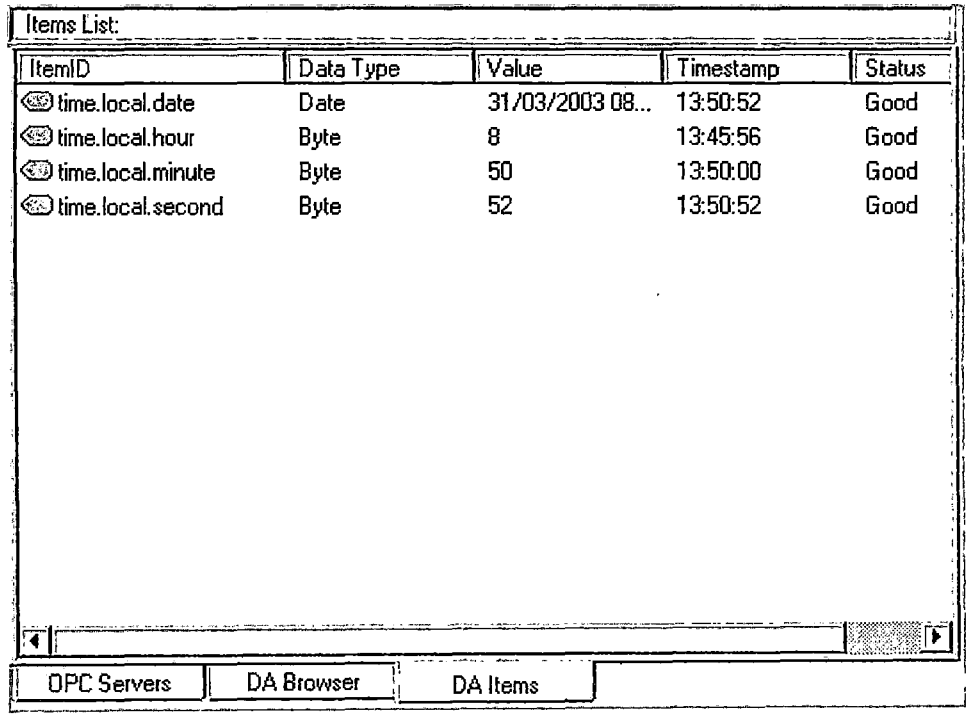


Figura 3.5 Pestaña *DA Items*

3.1.6 Lista de Mensajes

El control mostrado en la figura 3.6 muestra los eventos y errores producidos. Este control está dividido en tres columnas: la fecha (Date), hora (Time) del evento o error producido y finalmente la descripción mostrada en la columna evento (Event).

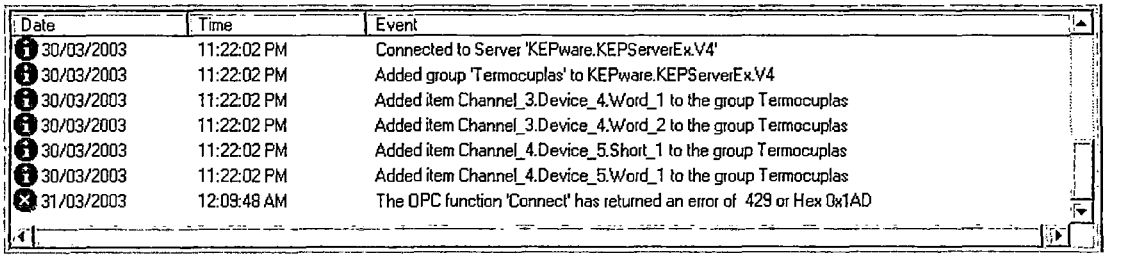


Figura 3.6 *Lista de Mensajes*

3.2 Creación de Objetos

Después de conocer toda la Interfaz de usuario que ofrece el software es necesario saber cómo realizar una conexión con el servidor (objeto *OPCServer*), cómo crear grupos (objetos *OPCGroup*) e items (objetos *OPCItem*). Estos objetos tienen una secuencia de creación el cual está especificada por OPC.

Primero se debe realizar la conexión con un servidor. Una vez realizado el primer paso, un nuevo grupo puede ser adicionado a la conexión, teniendo la facilidad de variar las propiedades: nombre del grupo (*Group Name*), *UpdateRate*, *DeadBand*, y *Status*. Finalmente, los items pueden ser adicionados a los grupos. Estos items tienen propiedades iniciales que se pueden variar, tales como: *ActiveState* y *DataType*.

3.2.1 Conexión con un Servidor de OPC

Esta conexión provee un lazo entre un servidor de OPC y el cliente de OPC, gracias a esta conexión, los grupos pueden ser adicionados al servidor.

Una forma directa de hacer una conexión con un servidor de OPC local, es desplegar la lista de servidores que se muestran al seleccionar la pestaña **OPCServers** de la ventana principal (ver figura 3.7). Luego se debe hacer doble click en un servidor de OPC deseado, para realizar la conexión con este.

Para poder visualizar la lista de servidores de OPC que están presentes en la máquina local o remota, tal como se aprecia en la figura 3.8, se debe seleccionar **New Server...** de menú **Edit**, se debe presionar el botón **New Server** de la **Barra de Herramientas** o se debe hacer click derecho sobre el servidor seleccionado en el **Explorador de Conexiones** y elegir **New Server Connection...** del menú desplegable.

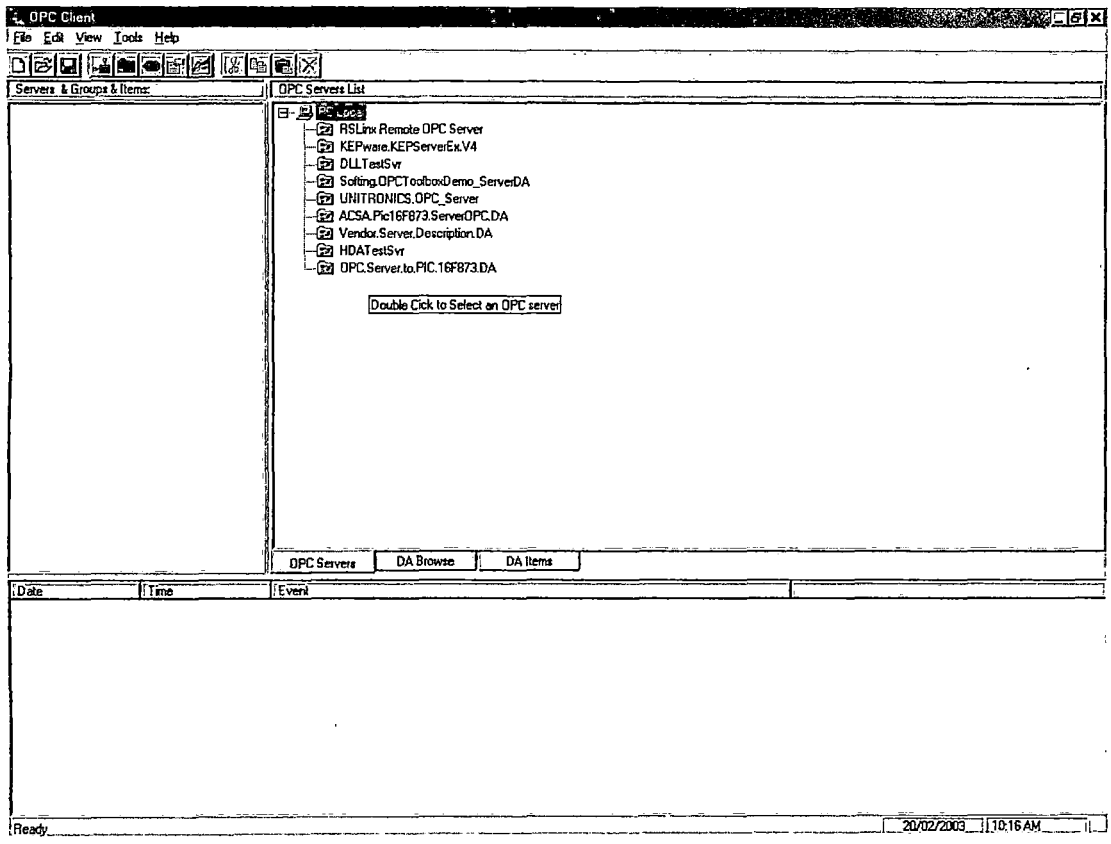


Figura 3.7 Pestaña *OPC Servers* que permite realizar una conexión con un servidor de OPC local

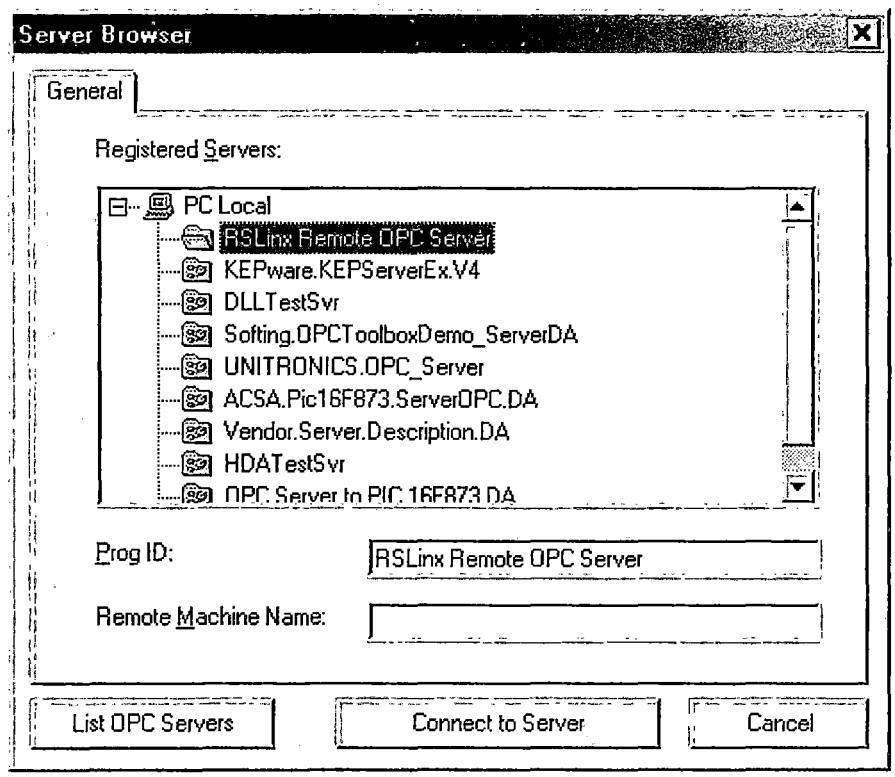


Figura 3.8 Ventana *Server Browser* que permite realizar una conexión con el servidor de OPC local o remoto

3.2.1.1 Campo *Prog ID*

Especifica el identificador del Programa del servidor de OPC, con el cual el cliente se va ha conectar. Se puede buscar los servidores registrados en la máquina local, expandiendo la rama *PC Local* (ver figura 3.8).

El campo **Prog ID** se actualizará con el nombre del servidor de OPC, Al hacer doble click en cualquier servidor de OPC de la lista desplegable de la figura 3.8. Luego se debe presionar el botón **Connecto to Server** para realizar el proceso de conexión; si se trata de un servidor de OPC remoto se debe de rellenar el campo **Remote Machine Name** con el nombre del nodo donde se encuentra este.

3.2.1.2 Campo *Remote Machine Name*

Especifica el nombre del nodo de la máquina donde reside el servidor de OPC remoto (especificado por **ProgID**). Si se desea saber la lista de servidores de OPC registrados en una máquina remota, es necesario rellenar el campo **Remote Machine Name** con el nodo de la máquina remota, para luego presionar el botón **List OPC Servers.** Si el servidor está localizado en una máquina local, se debe dejar en blanco este campo.

Una vez que el cliente realiza una conexión satisfactoria con un servidor, se puede invocar las propiedades del mismo (ver figura 3.9) al seleccionar **Properites...** del menú **Edit,** al presionar el botón **Properties** de la **Barra de Herramientas** o al elegir **Properties...** del menú desplegable que aparece al hacer click derecho sobre el servidor en el **Explorador de Conexiones.**

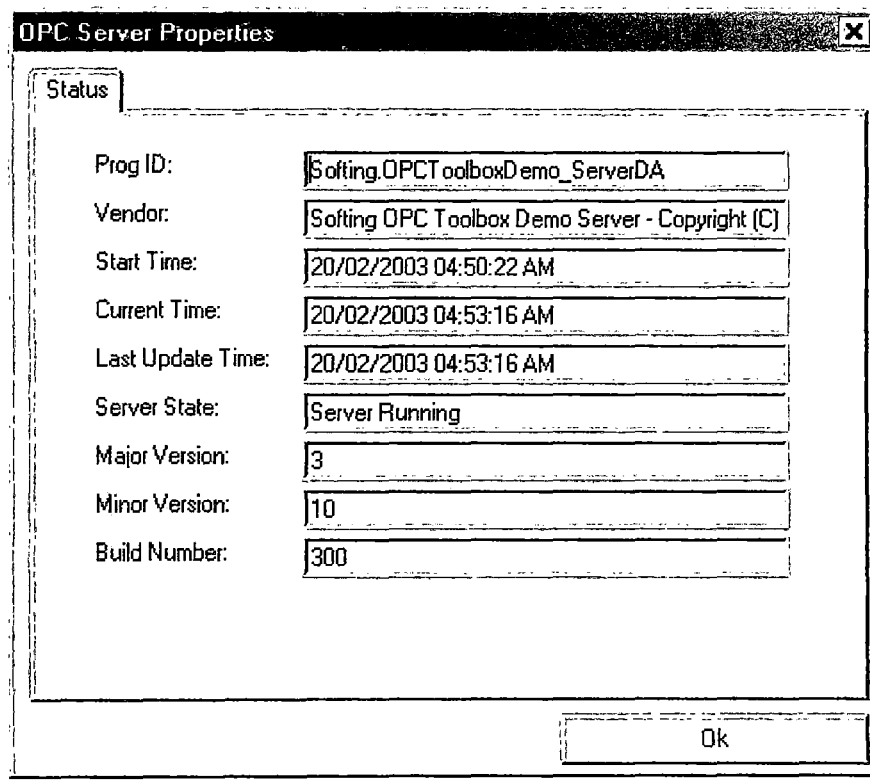


Figura 3.9 Ventana *OPC Server Properties* que muestra las propiedades de un servidor de OPC

La ventana de la figura 3.9 detalla las propiedades y el estado actual del servidor seleccionado. Esta información consta de lo siguiente:

- Prog ID — Cadena que muestra el identificador del programa del Servidor de OPC.

- Vendor — Una cadena que muestra la información específica de cada Servidor.
- Start Time — El tiempo en que la aplicación del servidor ha comenzado.
- Current Time — Tiempo actual del servidor.
- Last Update Time — Tiempo en el cual el servidor ha enviado el último valor al cliente.
- Server State — Estado actual del servidor (ver estados del servidor en el Apéndice B, cuadro B 19).
- Major Version — La versión mayor del Servidor, por ejemplo este valor podría ser el número "3" de la versión "3.12.33".
- Minor Version — La versión menor del Servidor, por ejemplo este valor podría ser el número "12" de la versión "3.12.33"
- Build Number — El número Build de la versión del Servidor, por ejemplo este valor podría ser el número "33" de la versión "3.12.33"

Todos estos parámetros están especificados por OPC.

3.2.2 Creación de un Nuevo Grupo

Un grupo es usado para organizar una colección de items. Para crear un nuevo grupo es necesario primero seleccionar un servidor en el **Explorador de Conexiones**. Luego se debe seleccionar **New Group...** del menú **Edit,** presionar el botón **New Group** de la **Barra de Herramienta** o seleccionar **New Group** del menú desplegable mostrado al hacer click derecho sobre el servidor seleccionado en el **Explorador de Conexiones**, para luego obtener una ventana similar al de la figura 3.10.

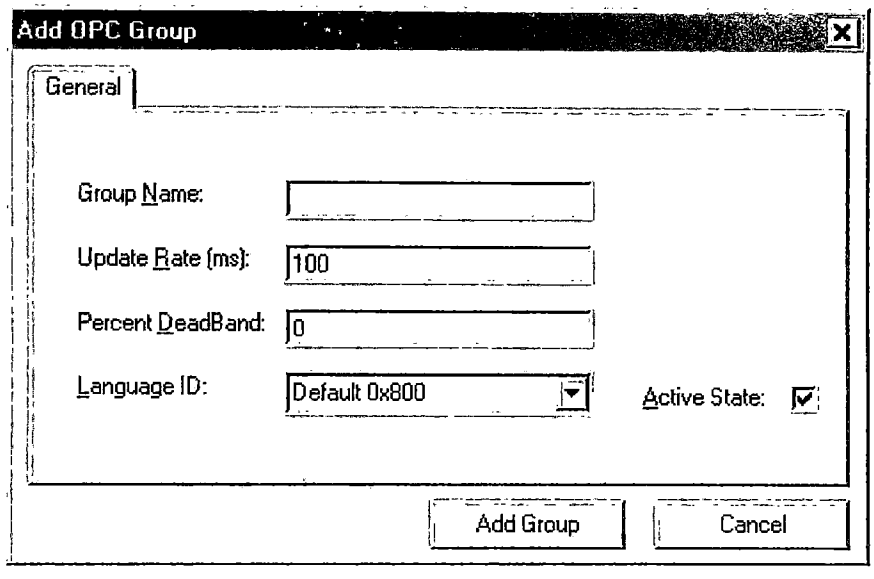


Figura 3.10 Ventana *Add OPC Group* que permite adicionar un nuevo grupo a un servidor de OPC

La ventana de la figura 3.10 permite preconfigurar las propiedades del grupo que va ha ser adicionado a un servidor, estas propiedades vienen a ser:

- Name
- Update Rate
- Percent DeadBand
- Language ID
- Active State

3.2.2.1 Campo *Name*

Especifica un nombre para identificar el grupo. Si éste no es ingresado, el Servidor de OPC generará un nombre único. Si se ingresa un nombre ya existente en el servidor de OPC seleccionado, el software OPC Client generará un nombre único adicionándole un índice al final.

3.2.2.2 Campo *Update Rate*

Especifica cuan a menudo, en milisegundos, el servidor de OPC va ha proveer la actualización de los cambios de las propiedades *Value* y *Quality* de los items pertenecientes al grupo.

3.2.2.3 Campo *Percent Deadband*

Especifica el porcentaje del valor del ítem que se tomará en cuenta, para notificar al Software cliente de OPC de un cambio producido en este.

3.2.2.4 Campo *Language ID*

Especifica el lenguaje que será usado por el servidor, para la descripción de un código de error. Esta propiedad es usada en con la ventana *Get Error* (ver figura 3.14).

3.2.2.5 Campo *Active State*

Especifica el estado del grupo que se va ha adicionar. Cuando el grupo es activado, el Software cliente de OPC recibirá la actualización producidos en los datos (*Value y Quality*) de los items activos, en caso contrario no recibirá ninguna notificación.

Cuando el grupo ha sido adicionado satisfactoriamente al proyecto, se puede invocar a las propiedades del grupo al seleccionar **Properties...** del menú **Edit,** al presionar el botón **Properites...** de la **Barra de Herramientas** o al hacer click derecho sobre el grupo seleccionado en el **Explorador de Conexiones** y escoger

Properties... del menú desplegable mostrado. Para luego modificar las propiedades del grupo en pleno trabajo (ver figura 3.11).

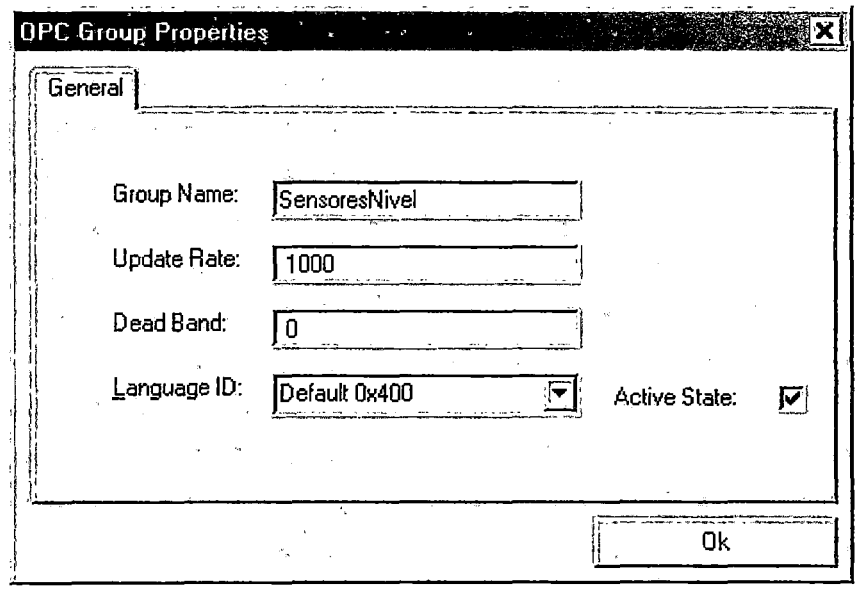


Figura 3.11 Ventana *OPC Group Properties* que permite modificar las propiedades del grupo en pleno trabajo

Los campos mostrados en la figura 3.11 son los mismos de la figura 3.10, por lo tanto tienen la misma función antes mencionada.

3.2.3 Creación de un Nuevo Ítem

Cada ítem representa el dato que puede ser accedido vía un servidor de OPC. Para adicionar un ítem es necesario primeramente seleccionar en el **Explorador de Conexiones** el grupo donde se desea adicionar el ítem, luego se debe presionar el botón **New Item** de la **Barra de Herramientas** o se debe hacer click derecho sobre el grupo seleccionado y escoger **New Item** en el menú desplegable mostrado, para observar el *namespace* del servidor (pestaña **DA Browser**, ver figura 3.12). Finalmente se debe hacer doble click en el ítem deseado para adicionarlo al grupo seleccionado.

Si el servidor de OPC no soporta el objeto OPC Browser (no tiene *namespace* configurado), se debe adicionar el ítem usando la opción **New Item...** del menú **Edit,** para obtener la ventana que se muestra en la figura 3.13, donde se muestra las propiedades que se pueden modificar antes de adicionar el ítem. Estas propiedades son:

- Item ID
- Data Type
- Active State
- Validate

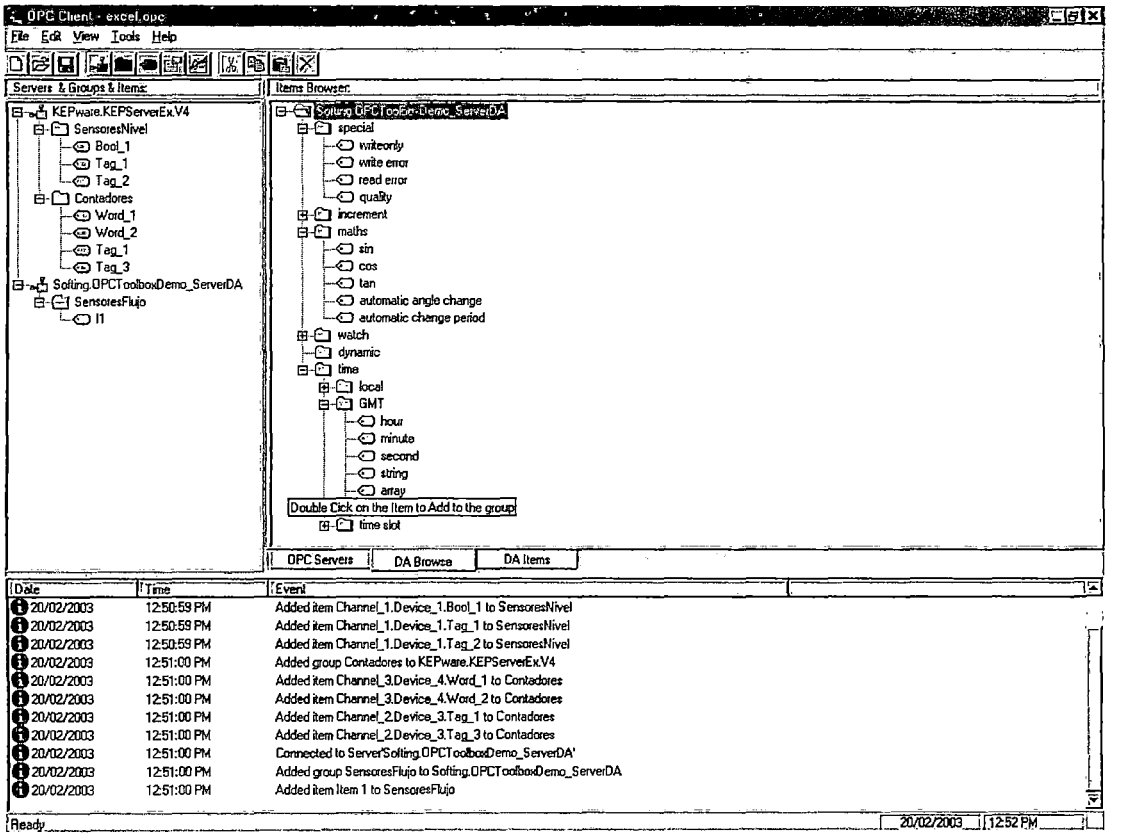


Figura 3.12 Pestaña *DA Browser* que permite mostrar el *namespace* del servidor de OPC

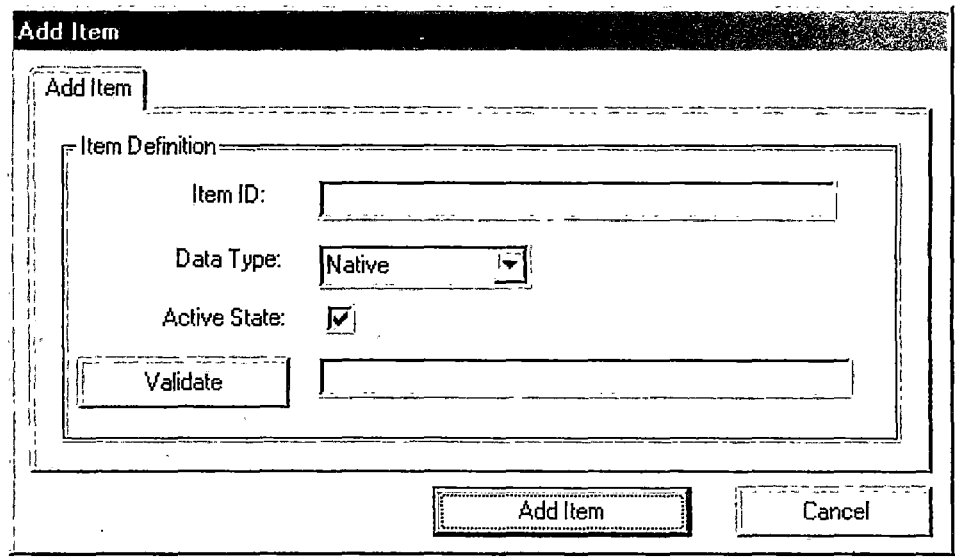


Figura 3.13 Ventana *Add Item* que permite adicionar un ítem y modificar sus propiedades antes de adicionarlo

3.2.3.1 Campo *Item ID*

El servidor de OPC usa esta propiedad para referenciar a un dato dentro del *namespace*.

3.2.3.2 Campo *Data Type*

Especifica el tipo de dato que va ha ser usado para la comunicación entre el servidor de OPC y el cliente OPC Client. Los tipos de datos que soporta el software OPC Client son las siguientes:

- **Native** Tipo de dato por defecto determinado por el servidor.
- **Boolean** Un simple bit.
- **BooleanArray** Arreglo de bits
- **Char** 8 bits con signo.
- **CharArray** Arreglo de variables tipo char.
- **Byte** 8 bits sin signo.
- **ByteArray** Arreglo de variables tipo Byte.
- **Short** 16 bits con signo.
- **ShortArray** Arreglo de variables tipo Short.
- **Word** 16 bits sin signo.
- **WordArray** Arreglo de variables tipo Word.
- **Long** 32 bits con signo.
- **LongArray** Arreglo de variables tipo Long.
- **Dword** 32 bits sin signo.
- **DwordArray** Arreglo de variables tipo Dword.
- **Float** 32 bits de simple precisión con punto flotante.
- **FloatArray** Arreglo de variables tipo Float.

- **Double** 64 bits de doble precisión con punto flotante.
- **DoubleArray** Arreglo de variables tipo Double.
- **Currency** 64 bits con signo para una variable tipo moneda.
- **CurrencyArray** Arreglo de variables tipo Currency.
- **Date** 64 bits para una variable tipo fecha.
- **DateArray** Arreglo de variables tipo Date.
- **String** Secuencia de Caracteres.

3.2.3.3 Campo *Active State*

Especifica el estado inicial que tendrá el ítem cuando sea adicionado, este valor puede ser activado o desactivado, si el ítem está desactivo no podrá recibir las notificaciones cuando las propiedades (*Value, Quality*) del ítem cambian de valor.

3.2.3.4 Botón *Validate*

Este botón verifica las propiedades del ítem, tales como: identificador del Ítem (*ItemID*), tipo de dato (*DataType*) y estado (*ActiveState*) antes de ser adicionado al grupo, los resultados se mostrarán en el campo situado al lado derecho de este botón.

3.2.3.5 Botón *AddItem*

Una vez verificadas las propiedades ingresadas para el ítem, se procede a adicionar el ítem a la configuración. Si no se ha verificado las propiedades del ítem previamente, el software OPC Client los verificará antes de adicionar el ítem respectivo.

3.2.3.6 Botón *Cancel*

Cancela la operación y cierra la ventana **Add Item**.

3.3 Operaciones de Intercambio de Datos con el Servidor de OPC

El software OPC Client permite realizar operaciones con los objetos *OPCServer*, *OPCGroup* y *OPCItem* especificados por OPC. Esto permite al usuario probar toda la funcionalidad del acceso de datos que ofrece la Interface de Automatización. Las siguientes operaciones se pueden llevar ha cabo:

3.3.1 Operaciones a Nivel del Servidor

3.3.1.1 Ventana *Get Error*

Al seleccionar **Get Error...** del menú **Tools|Server** se puede obtener la descripción de cualquier error válido, especificado por OPC (ver Apéndice B, tabla B 20) y el fabricante del servidor. (Ver figura 3.14.)

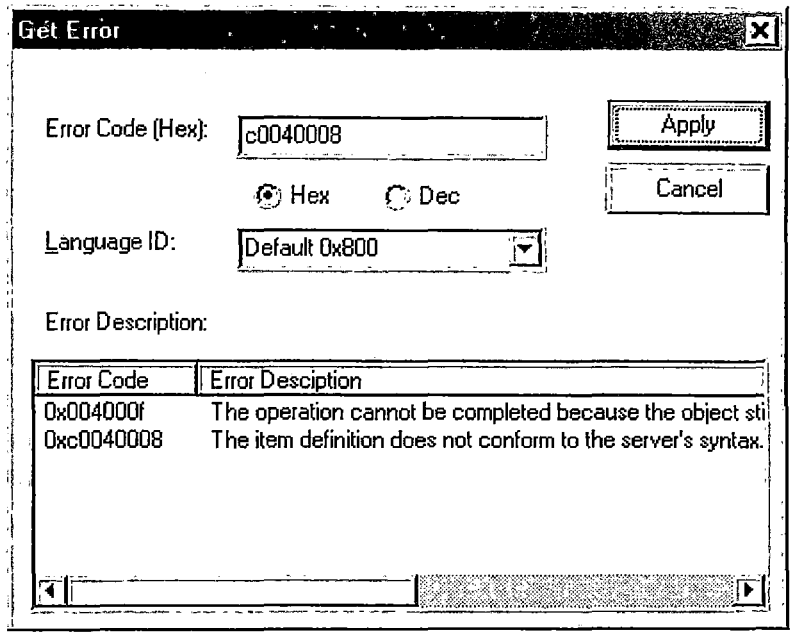


Figura 3.14. Ventana *Get Error* que permite obtener una cadena con la descripción de un código de error

Para pedir la descripción de un error al servidor, se debe ingresar el código del error en sistema hexadecimal o en sistema decimal (seleccionando la opción "Dec" de la figura 3.14), además se debe seleccionar el lenguaje que utilizará el servidor para notificar la descripción de este error. Finalmente se debe presionar el botón **Apply** de la figura 3.14.

Si el código del error es aceptado por el servidor, la descripción del error aparecerá como una lista, en caso contrario se mostrará un mensaje. Un servidor de OPC debería ser capaz de aceptar cualquier código de error de OPC, también los errores de Win32.

Esta operación prueba la funcionalidad de los siguientes métodos y propiedades del objeto *OPCServer* (ver Apéndice B):

Métodos

OPCServer.GetErrorString(), OPCServer.QueryAvailableLocaleIDs()

Propiedades

OPCServer.LocaleID

3.3.2 Operaciones a Nivel del *Namespace*

3.3.2.1 Ventana *Item Filter*

Al seleccionar **Item Filter** del menú **Tools|Item** se puede apreciar una ventana como se muestra en la figura 3.15, el cual permitirá realizar una búsqueda selectiva dentro del *namespace*, es decir permite filtrar los items que no se necesitan.

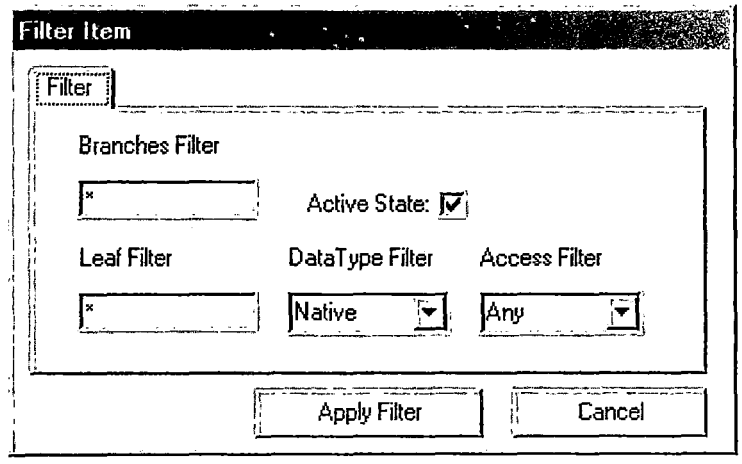


Figura 3.15 Ventana *Filter Item* que permite filtrar los items mostrados en el *namespace*

3.3.2.1.1 Campo *Branches Filter*

Aplica el filtro a los nombres de las ramas (nodos) del *namespace*.

3.3.2.1.2 Campo *Leaf Filter*

Aplica el filtro a los nombres de las hojas (*tags* o items) del *namespace*.

3.3.2.1.3 Campo *Active State*

Establece el estado (activado o desactivado) de los items que serán adicionados por medio de la pestaña **DA Browser** (ver figura 3.12).

3.3.2.1.4 Campo *DataType Filter*

Aplica el filtro sobre la base de la propiedad tipo de dato (*DataType*) del ítem dentro del *namespace*.

3.3.2.1.5 Campo *Access Filter*

Aplica el filtro utilizando la propiedad tipo de acceso (*AccessRight*) del ítem dentro del *namespace*.

Esta operación prueba la funcionalidad de los siguientes métodos y propiedades del objeto *OPCBrowser* (ver Apéndice B):

Métodos

OPCBrowser.ShowBranches, OPCBrowser.ShowLeafs

Propiedades

OPCBrowser.Filter, OPCBrowser.AccessRights, OPCBrowser.DataType

3.3.3 Operaciones a Nivel del Ítem y Grupo

3.3.3.1 Opción *Set Active/Inactive*

Al seleccionar **Set Active/Inactive** del menú **Tools|Item** se puede activar o desactivar el ítem seleccionado. Solamente un ítem activo puede recibir las actualizaciones cuando ocurre un cambio en la propiedad valor o calidad del ítem (el grupo al que pertenece también debe estar activo).

Esta operación prueba la funcionalidad de la siguiente propiedad del objeto *OPCItem* (ver Apéndice B):

Propiedades

OPCItem.IsActive

3.3.3.2 Opción *Synchronous Read (Cache/Device)*

Al seleccionar **Synchronous Cache Read** o **Synchronous Device Read** del menú **Tools|Item,** se puede llevar ha cabo una lectura síncrona del ítem o items seleccionados desde la *cache* o dispositivo respectivamente.

Esta operación prueba la funcionalidad del siguiente método y propiedad de los objetos *OPCItem* y *OPCGroup* (ver Apéndice B):

Métodos

OPCGroup SyncRead()

Propiedades

OPCItem.ServerHandle

3.3.3.3 Opción *Asynchronous Read (Cache/Device)*

Al seleccionar **Asynchronous Cache Read** o **Asynchronous Device Read** del menú **Tools|Item,** se puede llevar ha cabo un proceso de lectura asíncrona del ítem o items seleccionados desde la *cache* o dispositivo respectivamente.

Esta operación prueba la funcionalidad de los siguientes métodos y propiedades de los objetos *OPCItem* y *OPCGroup* (ver Apéndice B):

Métodos

OPCGroup AsyncRead(), OPCGroup_AsyncCancelComplete(), OPCGroup_AsyncReadComplete().

Propiedades

OPCItem.ServerHandle

3.3.3.4 Opción *Asynchronous Refresh* (*Cache/Device*)

Al seleccionar **Asynchronous Cache Refresh** o **Asynchronous Device Refresh** del menú **Tools|Item,** se puede llevar ha cabo una actualización asíncrona del ítem o items presentes en el grupo seleccionado desde la cache o dispositivo respectivamente.

Esta operación prueba la funcionalidad del siguiente método del objeto *OPCGroup* (ver Apéndice B):

Métodos

OPCGroup.AsyncRefresh()

3.3.3.5 Opción *Synchronous/Asynchronous Write*

Al seleccionar **Synchronous Write** o **Asynchronous Write** del menú **Tools|Item** (ver figura 3.16), se puede llevar ha cabo los métodos de escritura síncrona o asíncrona del ítem o items seleccionados.

Esta operación prueba la funcionalidad de los siguientes métodos y propiedades del objeto *OPCGroup* (ver Apéndice B):

Métodos

OPCGroup.AsyncWrite(), OPCGroup.SyncWrite ()

Propiedades

OPCItem.ServerHandle

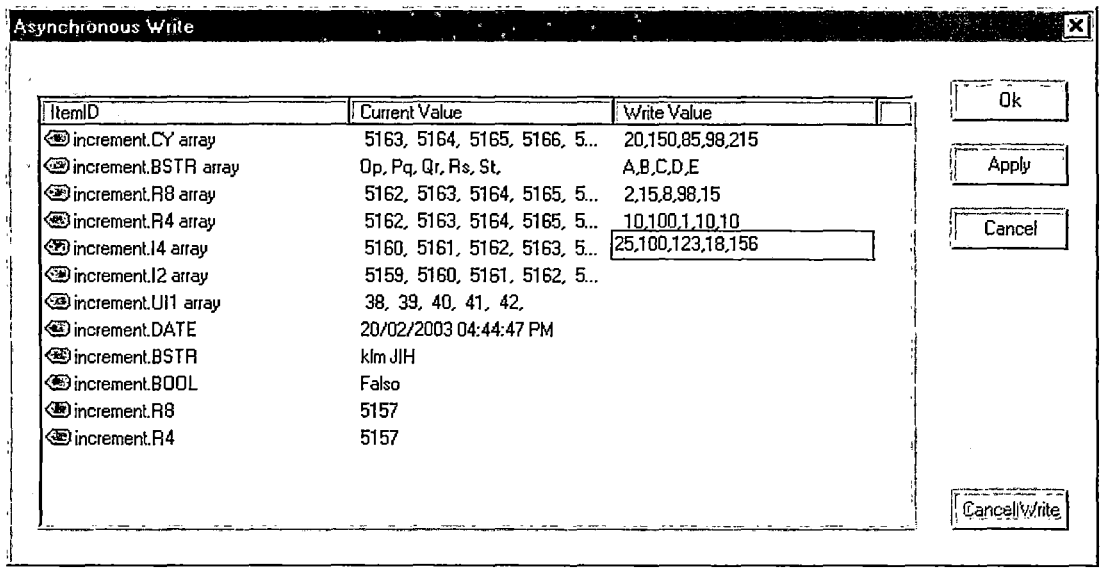


Figura 3.16 Ventana *Synchronous/Asynchronous Write* que permite realizar la operación de escritura

Los dos tipos de escritura utilizan la ventana de la figura 3.16, esta figura se divide en tres columnas: el identificador del ítem (*ItemID*), el valor del ítem, y una columna donde se ingresa los valores que serán escritos. Una vez ingresados estos valores se debe presionar el botón **Apply** para enviar el dato hacia el servidor de OPC. Al presionar el botón **OK**, se realiza la escritura de los datos y además cierra la ventana. El botón **Cancel** permite cerrar la ventana actual sin realizar ningún procedimiento.

El botón **Cancel Write** (se activa cuando se realiza el procedimiento Asynchronous Write) cancela esta transacción de escritura.

3.3.4 Operaciones Gráficas

El software OPC Client está desarrollado para proveer una interfaz gráfica, que permite mostrar en forma gráfica los valores de los items en tiempo real.

Para tener esta funcionalidad es necesario seguir el siguiente procedimiento:

- Configuración de la ventana **Tagname Dictionary**
- Configuración de la ventana **Real Time Trend Configuration**

3.3.4.1 Configuración de la Ventana *Tagname Dictionary*

Primeramente se debe seleccionar un ítem desde la lista de items mostrados en la pestaña **DA Items**, el cual se desea mostrar en forma gráfica. Luego se debe seleccionar **Tagname Dictionary** del menú **View**, para mostrar la ventana de la figura 3.17.

La ventana **Tagname Dictionary** posee varios campos que se debe rellenar, una vez configurados estos campos se debe presionar el botón **Save** para guardar la

configuración y si se desea eliminar esta configuración se debe presionar el botón **Cancel.** Al presionar el botón **Restore** se restablece todos los campos. Una vez terminada la configuración se debe presionar el botón **Close** para cerrar la ventana.

Es importante saber que los campos tales como Item, Group y Server, mostrados en la figura 3.17, son referidos al ítem seleccionado.

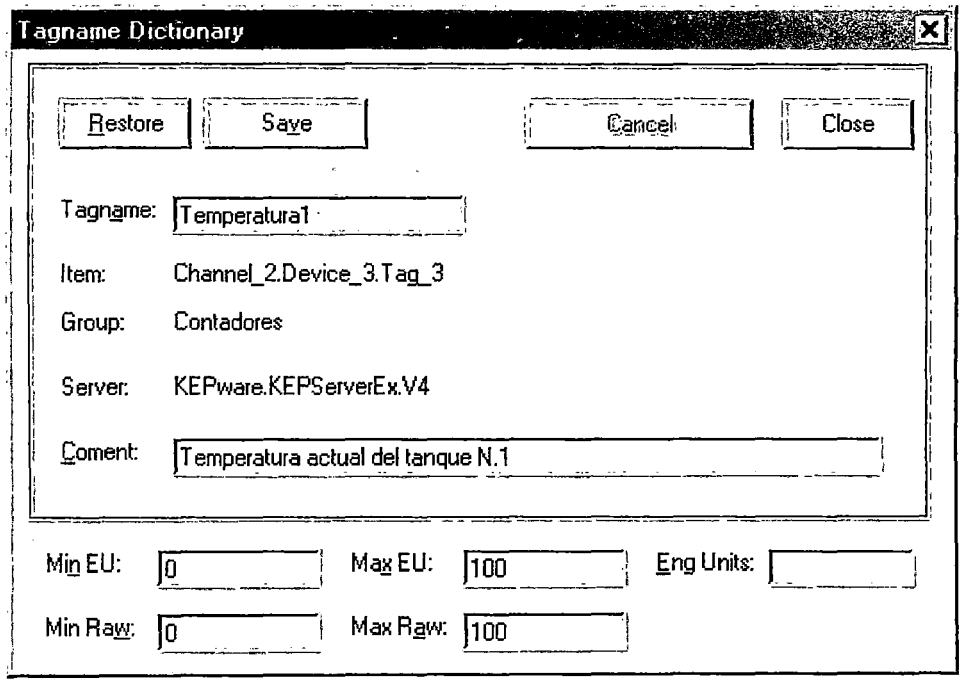


Figura 3.17 Ventana *Tagname Dictionary* que permite configurar el ítem que será graficado

3.3.4.2 Configuración de la Ventana *Real Time Trend Configuration*

Una vez que ya se tiene todos los items que se van ha graficar es necesario configurar las propiedades del gráfico. La ventana **Real Time Trend Configuration** es mostrada al seleccionar **Configuration** del menú **Arrange** de la ventana **Real Time Trend**. Para acceder a esta ventana se debe seleccionar **Real Time Trend...** del menú **View.**

En la figura 3.18 se muestra de la ventana **Real Time Trend Configuration**, el cual permite configurar los parámetros del gráfico (ventana **Real Time Trend)** y cuyos campos son:

- **Time** Especifica el rango del eje de la abscisa del gráfico (en segundos, minutos, horas).
- **Sample** Especifica el tiempo de muestreo (en milisegundo, segundos, minutos, horas).
- **Color** Especifica el color de relleno del gráfico (Fill Color) y el color de las líneas del gráfico (Line Color).
- **Time Divisions** Especifica las unidades que se mostrarán en el eje de las coordenadas.
- **Value Divisions** Especifican el rango máximo y mínimo del eje de las abscisas.

El campo **Active** mostrado en la parte inferior izquierda de la figura 3.18 permite activar el número de items que se mostrarán en el gráfico, 5 es el número máximo con el que se puede trabajar.

Además, se puede hacer un click en campo **Color** para cambiar el color del lápiz, con el que el ítem se va ha ver en el gráfico. El campo **Width** permite cambiar el ancho de dicho lápiz. El campo **Calidad** indica cual es el estado (propiedad Quality de OPC) del ítem, si el estado es "Good", entonces el ítem se mostrará gráficamente sin ningún problema.

Después de realizar la configuración se debe presionar el botón **OK** para ver la ventana **Real Time Trend** (ver figura 3.19) con los parámetros ya configurados.

Se puede acceder a los gráficos al seleccionar **Real Time Trend** del menú **View** o al presionar el botón **Trend** de la **Barra de Herramientas**, para obtener la ventana de la figura 3.19.

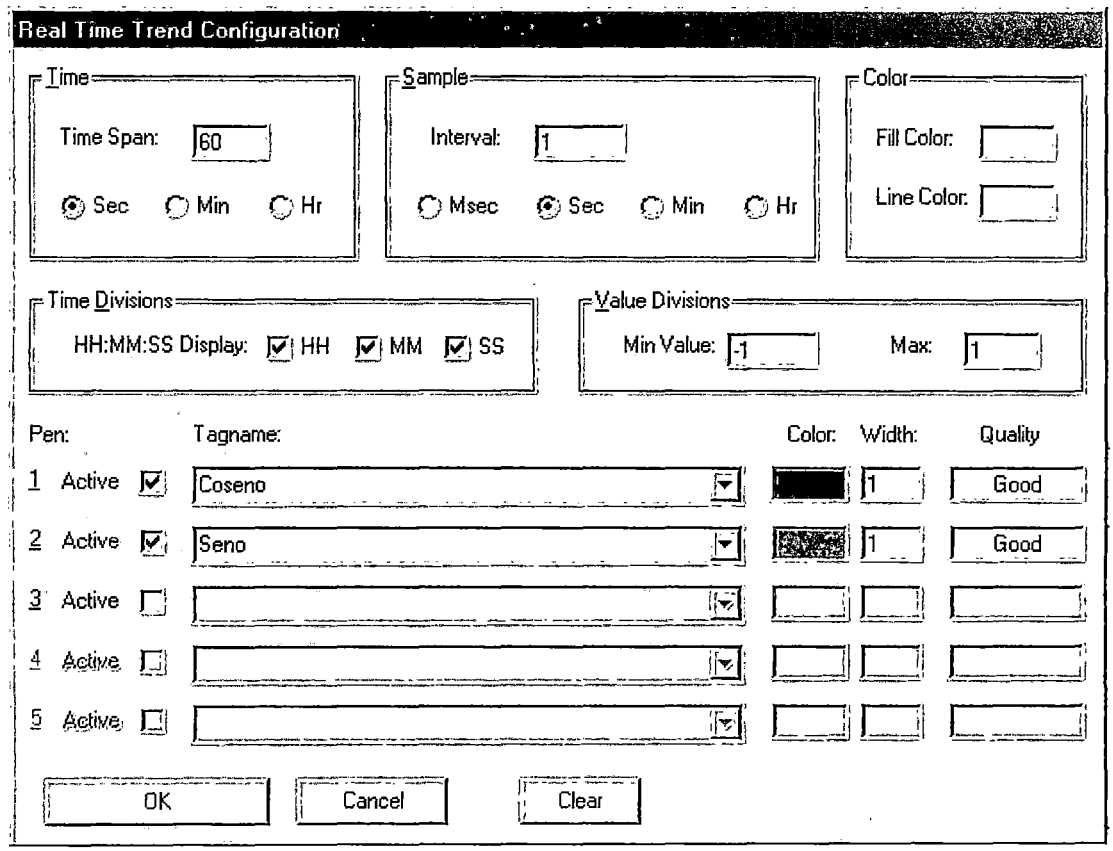


Figura 3.18 Ventana *Real Time Trend Configuration* que permite configurar las propiedades del gráfico *Real Time Trend*

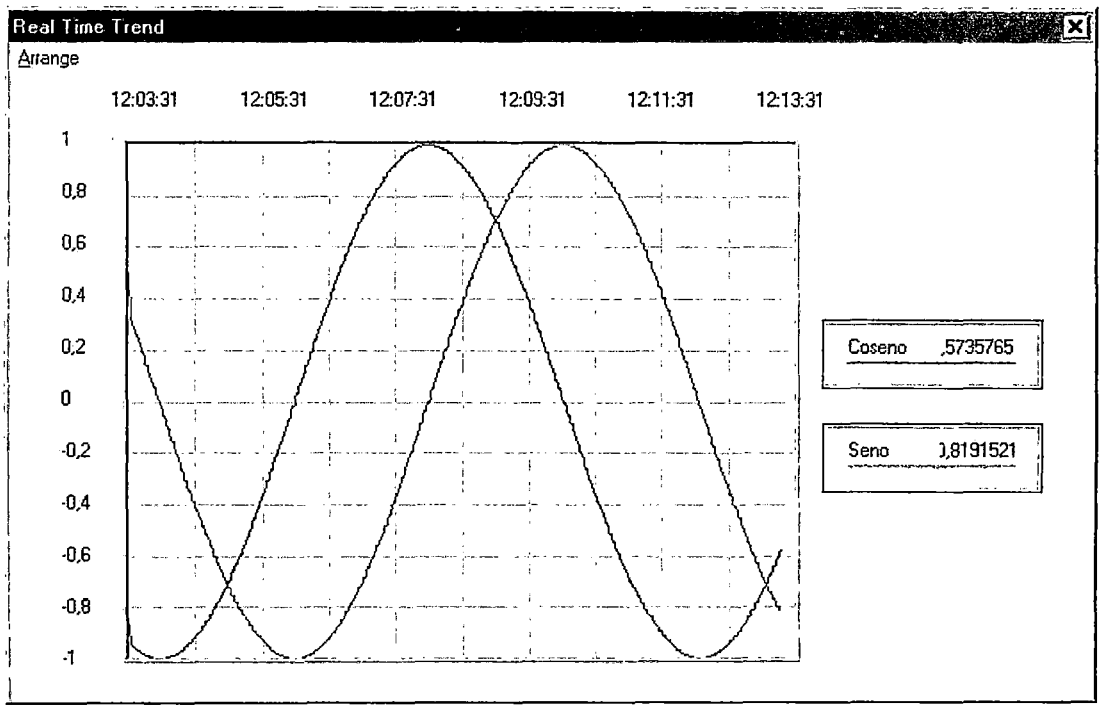


Figura 3.19 Ventana *Real Time Trend* que permite mostrar los items en forma gráfica

3.4 Interfaz de Usuario para Microsoft Excel

Al seleccionar **Excel...** del menú **View**, se podrá obtener la interfaz de usuario desarrollada para el software Microsoft Excel con la funcionalidad del acceso de datos de OPC.

Esta interfaz consta de un menú llamado "OPC Menu" (ver figura 3.20), el cual se obtiene habilitando los macros al abrir la aplicación de Excel.

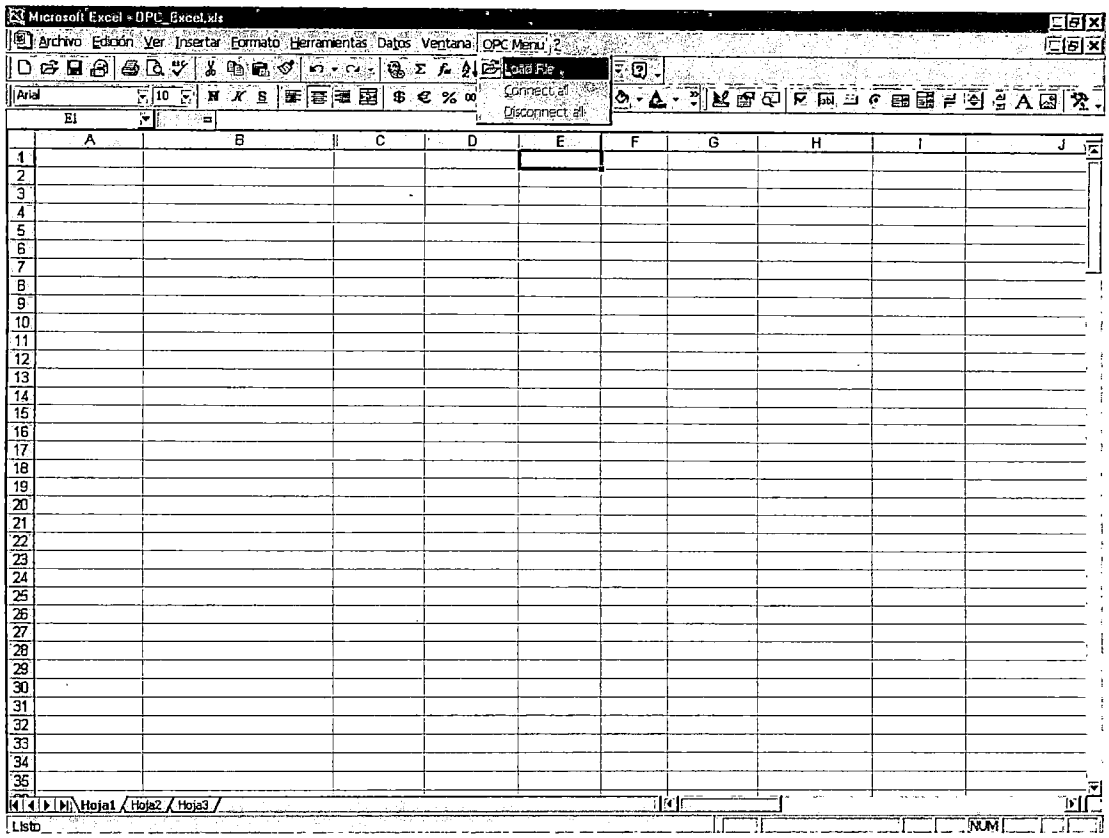


Figura 3.20 Menú *OPC Menu* para la Interfaz con Excel

El menú **OPC Menu** mostrado en la figura 3.20 consta de tres opciones:

- **Load File** Carga la configuración del archivo generado por el software OPC Client, archivo con extensión "*.opc".

- **Connect all** Realiza el proceso de conexión de la configuración del archivo cargado anteriormente. Los datos se mostrarán automáticamente utilizando el evento "DataChange" de la Interface de Automatización.

- **Disconnect all** Realiza el proceso de desconexión de la configuración del archivo conectado anteriormente.

Luego de cargar el archivo de configuración generado por el software OPC Client, se debe de conectarse con el servidor de OPC utilizando la opción **Connect all** del menú **OPC Menu**. Al finalizar la sección se debe seleccionar **Disconnect all,** para terminar con la conexión.

La gráfica 3.21 muestra como se actualizan en forma automática los datos en la hoja de cálculo. Con los datos en tiempo real se pueden generar gráficas y reportes muy fácilmente, ya que se puede utilizar toda la funcionalidad de Excel (ver figura 3.22).

El software OPC Client se encuentra en el CD adjunto con esta tesis.

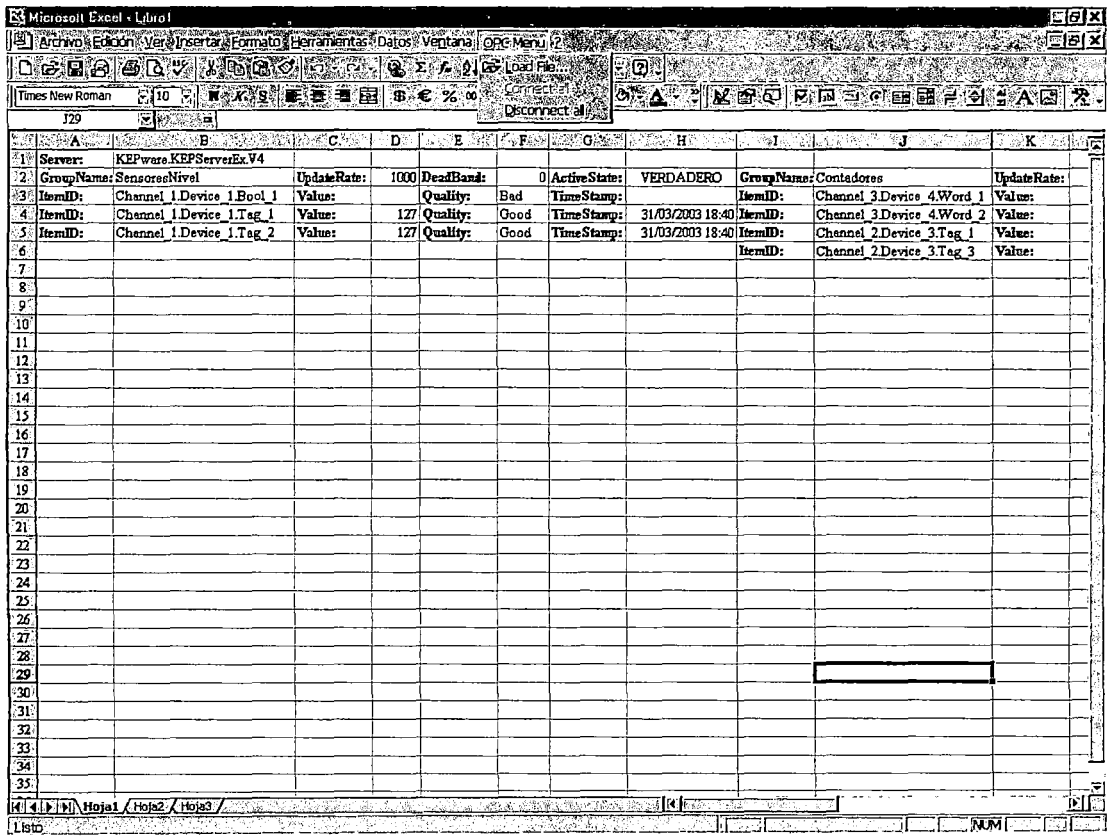
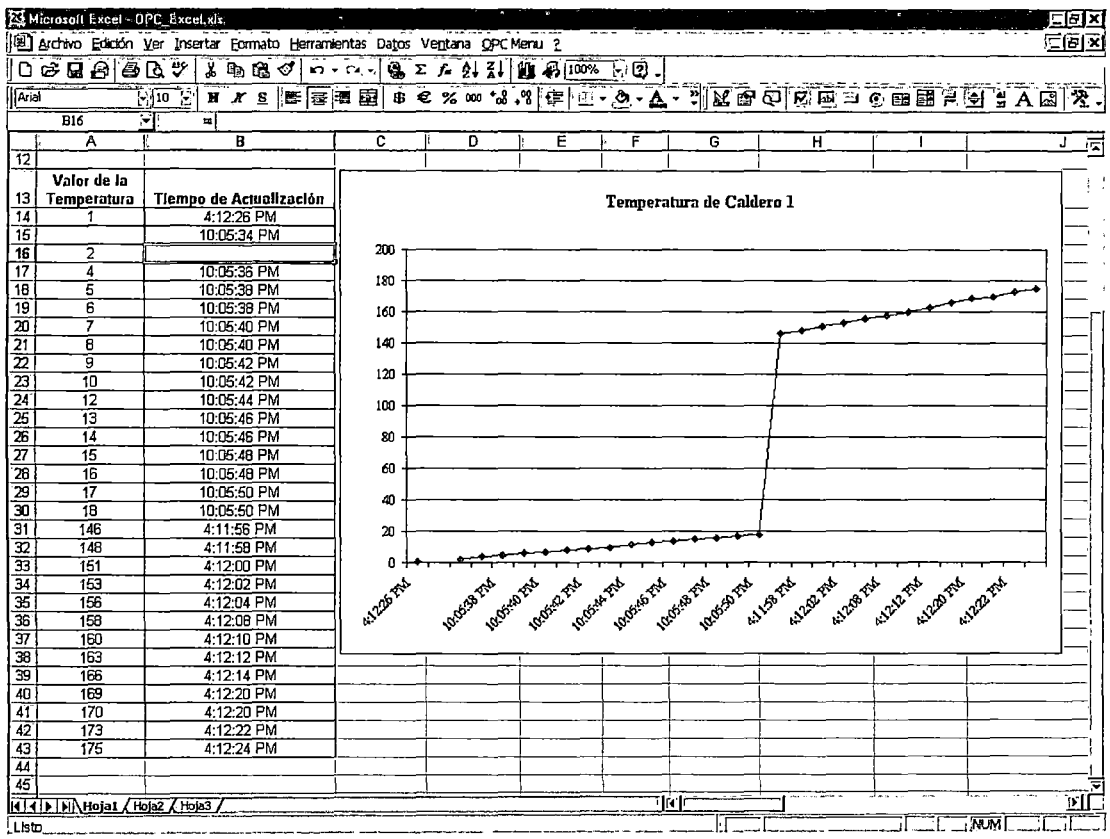


Figura 3.21 Actualización automática de las propiedades *Value, Quality* y *TimeStamp* de la configuración cargada inicialmente



Valor de la Temperatura	Tiempo de Actualización
1	4:12:26 PM
	10:05:34 PM
2	
4	10:05:36 PM
5	10:05:38 PM
6	10:05:38 PM
7	10:05:40 PM
8	10:05:40 PM
9	10:05:42 PM
10	10:05:42 PM
12	10:05:44 PM
13	10:05:46 PM
14	10:05:46 PM
15	10:05:48 PM
16	10:05:48 PM
17	10:05:50 PM
18	10:05:50 PM
146	4:11:56 PM
148	4:11:58 PM
151	4:12:00 PM
153	4:12:02 PM
156	4:12:04 PM
158	4:12:08 PM
160	4:12:10 PM
163	4:12:12 PM
166	4:12:14 PM
169	4:12:20 PM
170	4:12:20 PM
173	4:12:22 PM
175	4:12:24 PM

Figura 3.22 Gráfica en tiempo real de un ítem utilizando el acceso de datos de OPC y la funcionalidad de Excel

CAPÍTULO IV

DISEÑO E IMPLEMENTACIÓN DEL SERVIDOR DE OPC PARA EL MICROCONTROLADOR PIC 16F873

Este capítulo consta de todos los procedimientos realizados para crear el Servidor de OPC, el cual tendrá como dispositivo un microcontrolador PIC 16F873. La parte central de este capítulo es básicamente el desarrollo de la funcionalidad de OPC, ya que independientemente del dispositivo que se use la estructura será la misma

Tal como se muestra en el capítulo 1 existen diferentes especificaciones de OPC que están disponibles capa cada tipo de aplicación. En este caso se usa la especificación para el acceso de datos (ver tabla 1.1).

Para desarrollar el Servidor de OPC para el microcontrolador, se debe primeramente definir el hardware y el protocolo de comunicación que utilizará el

microcontrolador PIC para comunicarse con la PC. Para finalmente desarrollar la funcionalidad del acceso de datos de OPC.

4.1 Hardware para la Comunicación con la PC

El hardware depende del tipo de interface que se va a utilizar, para este caso se utiliza la interface serial RS-232.

El circuito de comunicación serial es mostrado en la figura 4.1, el cual consta básicamente del microcontrolador PIC y un circuito integrado (MAX232), el cual convierte la señal de 0 a 5 V. a un nivel de 3 a 15 V para el estado lógico 0 y de -3 a -15 V para el estado lógico 1.

La comunicación con la PC solamente se realiza usando dos pines, pines de recepción y de envío de datos, el resto de pines (RTS, CTS, DSR, etc) no fueron habilitados ya que no se encuentran disponibles en el microcontrolador.

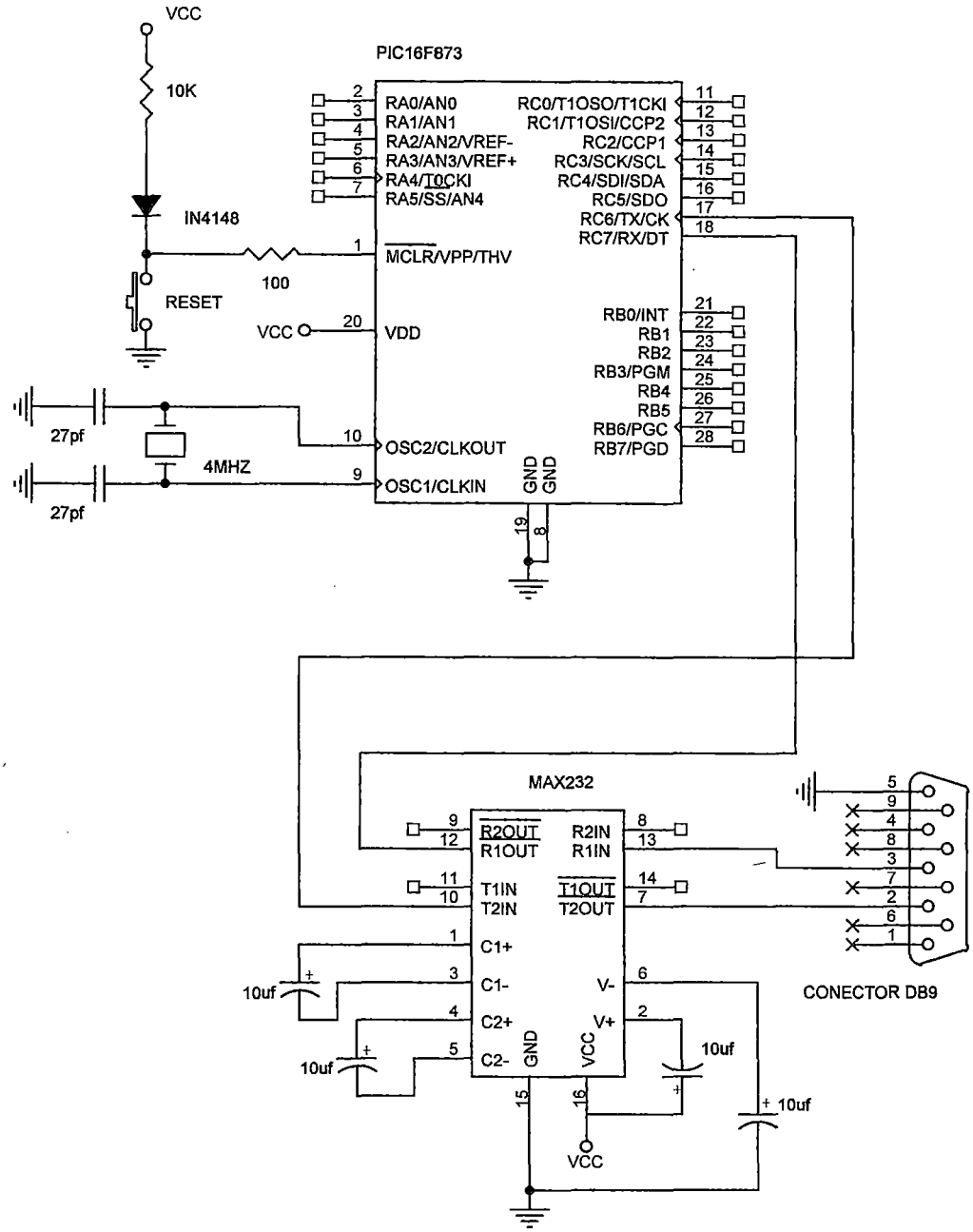


Figura 4.1 Esquema del hardware para la comunicación serial entre el microcontrolador con la PC

4.2 Programación del Microcontrolador para la Comunicación Serial

4.2.1 Definición del Protocolo de Comunicación

El protocolo de comunicación define un procedimiento o conjunto de reglas que permitan realizar el intercambio de información entre los dos dispositivos. La figura 4.2 establece claramente el protocolo definido para la comunicación serial. El cual tiene como objetivo realizar las operaciones de lectura o de escritura de todos los registros de memoria del microcontrolador.

El protocolo consta de un bloque de 2 bytes, de los cuales los dos primeros definen la dirección del dato y el tipo de operación a realizarse. El resto de bytes realiza la operación respectiva. La tabla 4.1 muestra el tiempo aproximado que tarda cada operación.

	Número de bytes Transferidos	**Número de bits transferidos (bits *start* y *stop*)**	**Tiempo para cada Operación (96100 bps)**
Operación de Lectura	4	40	4.16 ms
Operación de Escritura	5	50	5.21 ms

Tabla 4.1 Tiempo estimado para cada operación

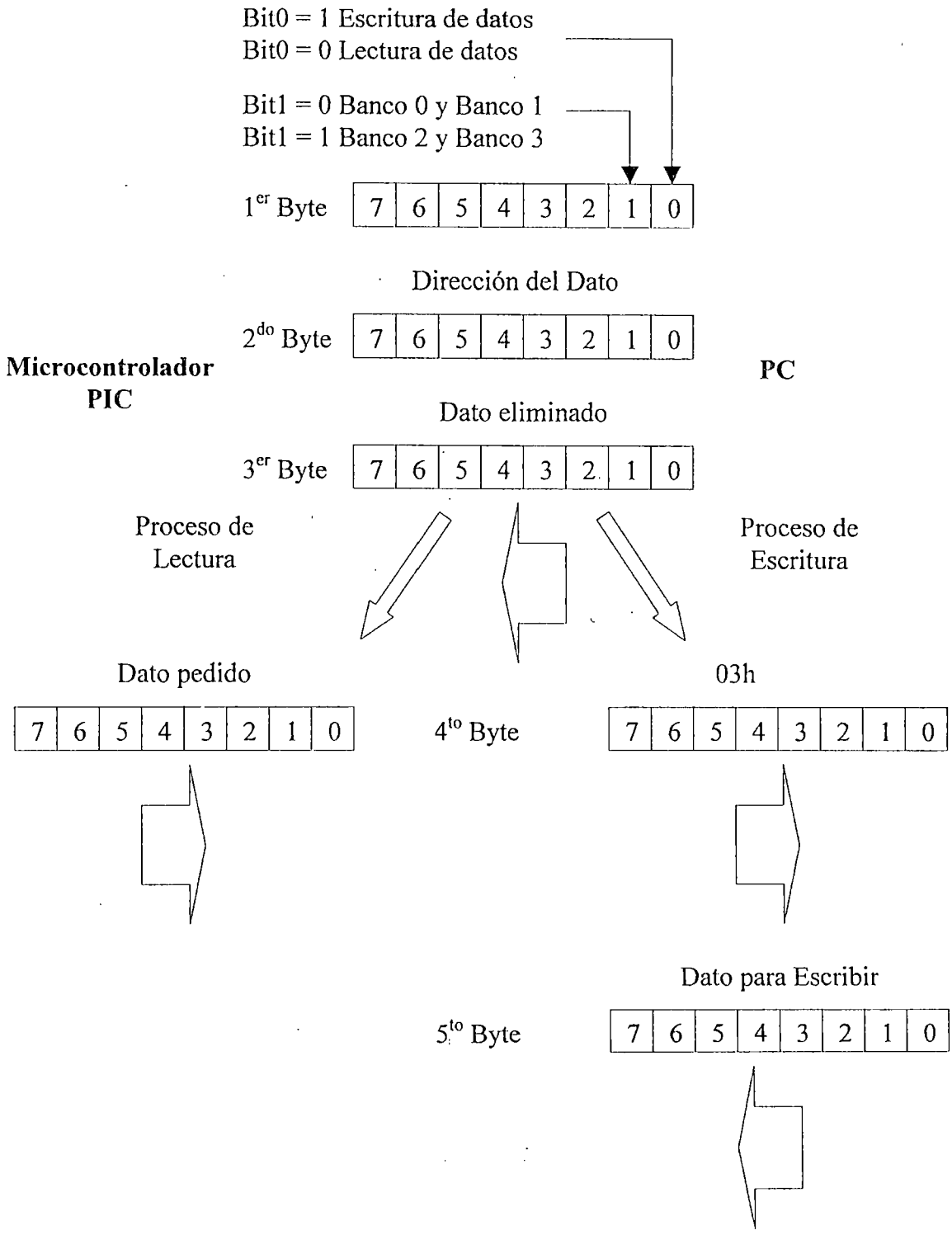


Figura 4.2 Protocolo de comunicación para el microcontrolador

4.2.2 Diseño del Algoritmo

Sobre la base del protocolo definido en la figura 4.2 se diseña el algoritmo que se utilizará para la programación del microcontrolador. Las figuras 4.3 y 4.4 muestran los algoritmos encargados de la comunicación serial.

Es importante tener presente que tanto la operación de lectura como la de escritura son síncronos, es decir estas operaciones tiene que terminar para poder seguir con su procedimiento habitual.

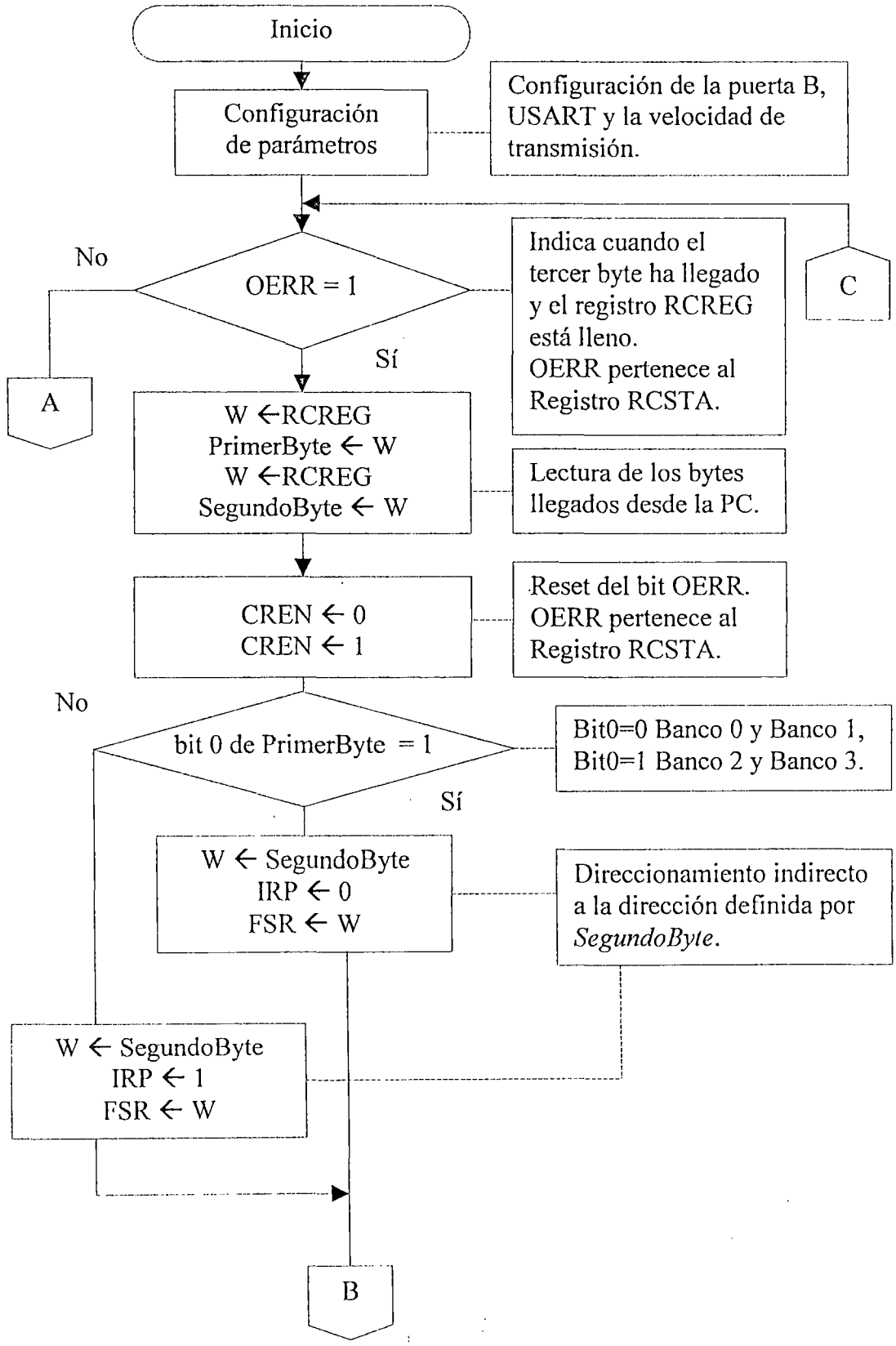


Figura 4.3 Algoritmo para el protocolo de Comunicación del microcontrolador (*continúa...*)

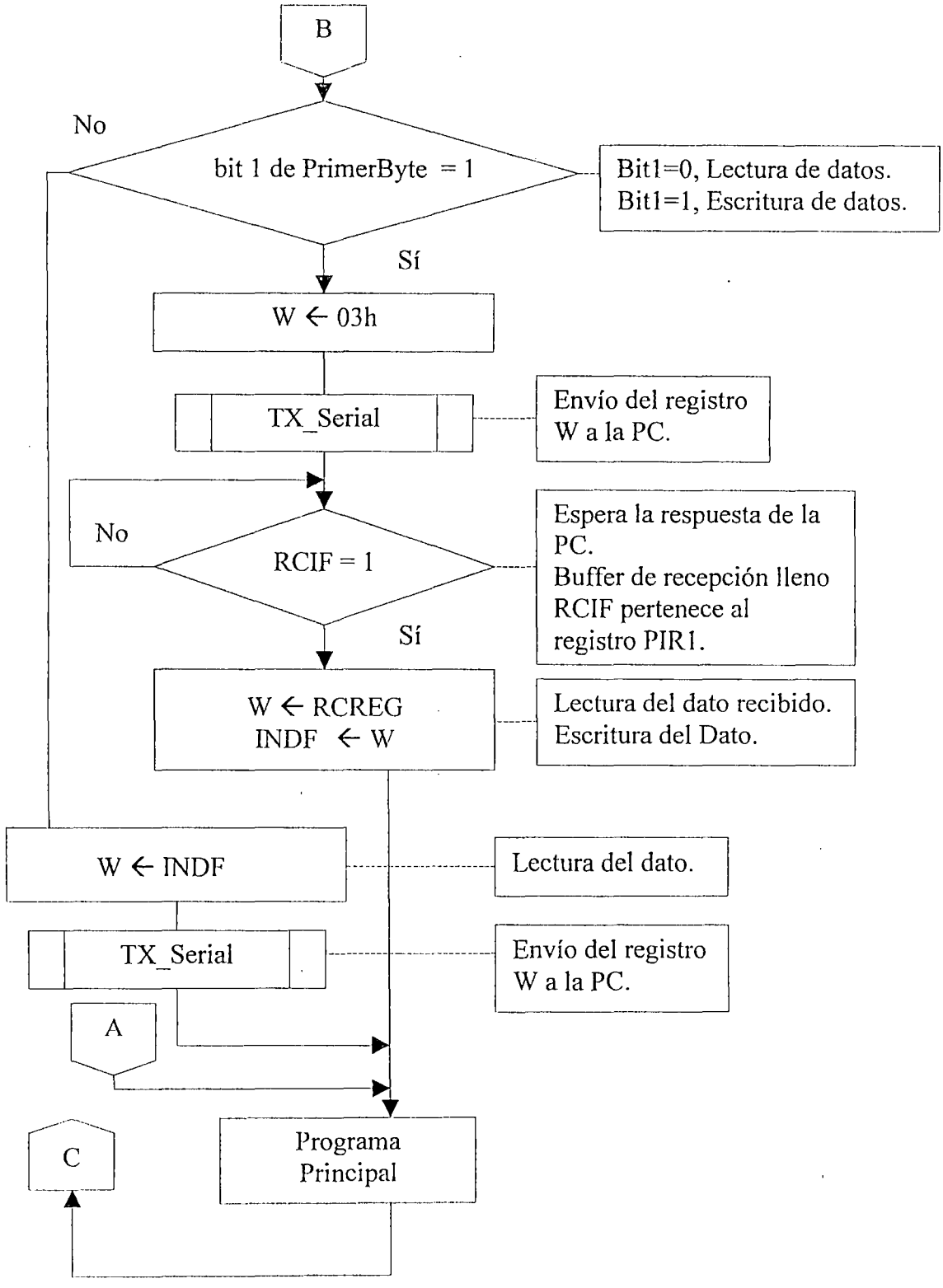


Figura 4.3 (*Continuación de la página 167.*) Algoritmo para el protocolo de comunicación del microcontrolador

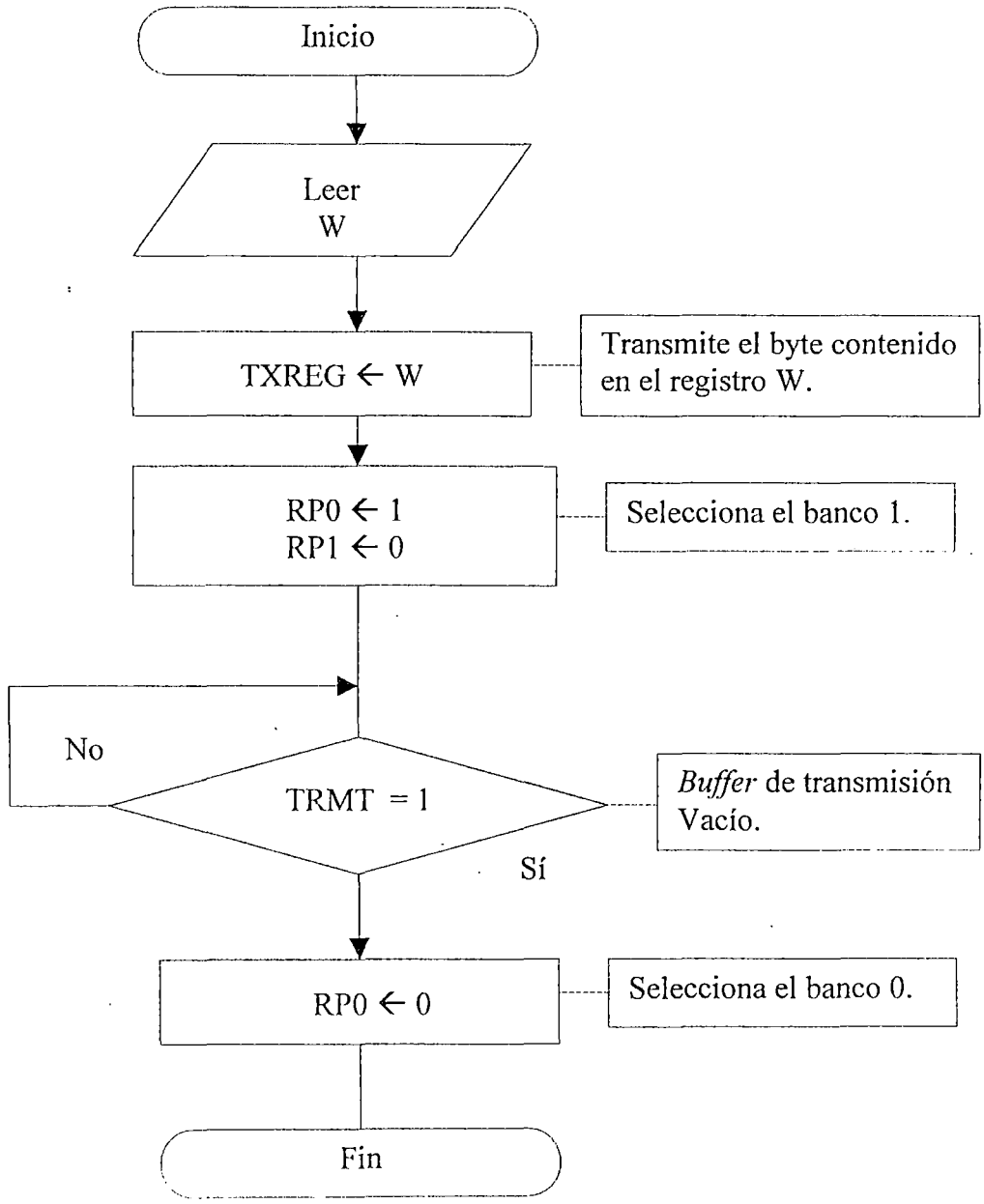


Figura 4.4 Algoritmo del procedimiento TX_Serial

4.2.3 Implementación del Algoritmo

El programa consta básicamente consta de las siguientes partes:

- Configuración de los parámetros de comunicación.
- Procedimiento de lectura y escritura de una dirección de memoria del microcontrolador.

Lista 4.1 muestra la configuración de los parámetros para la comunicación serial. En las líneas 5 y 4 de la lista 4.1 se definen las variables *PrimerByte* y *SegundoByte* los cuales almacenarán los datos leídos del Registro RCREG.

La configuración de los parámetros para la comunicación comienza en la etiqueta *config,* línea 12 de la lista 4.1, donde inicialmente se configuran los pines 17 (RC6/TX/CK) y 18 (RC7/RX/DT) del microcontrolador para que trabajen como salida y de entrada respectivamente. Las líneas 24 al 25 configuran la velocidad de transmisión (9600 Baudios), las líneas 26 al 27 configuran el registro TXTA con los siguientes valores: transmisión de 8 bits, Habilitación de la transmisión, modo Asíncrono y alta velocidad. El registro RCSTA es configurado en las líneas 29 al 30 de la siguiente manera: habilitación del puerto serial, 8 bits para la recepción y habilitación de la recepción.

La lista 4.2 viene hacer la implementación del algoritmo de la figura 4.3. El cual se inicia cuando la PC envía 3 bytes consecutivamente al microcontrolador, en

el primer byte (línea 8 de la lista 4.2) define el banco de memoria y el tipo de operación se va ha realizar, el segundo byte (línea 10 de la lista 4.2) contiene la dirección del registro de memoria. El tercer byte es eliminado, ya que solamente es usado para completar el bloque de tres bytes.

Las líneas 27 al 33 y 34 al 36 de la lista 4.2 vienen hacer los procedimientos para la operación de escritura y de lectura respectivamente. La implementación del algoritmo de la figura 4.4 es la lista 4.3. En la línea 37 de la lista 4.2 es posible adicionar un programa principal que será independiente de la comunicación serial.

```
1.  List        p=16f873
2.  ERRORLEVEL  -302
3.  #include    p16f873.inc

4.  PrimerByte      EQU  0x20
5.  SegundoByte     EQU  0x21

6.  __CONFIG _CP_OFF & _WDT_OFF & _XT_OSC & _LVP_OFF &
    _XT_OSC

7.          org     0
8.          goto    config
9.          org     4
10.         goto    inter
11.         org     5


12. config  clrf    PORTB
13.         clrf    PORTC
14.         clrf    PORTA
15.         bsf     STATUS,RP0 ;Banco1
16.         bcf     STATUS,RP1
17.         clrf    TRISB      ;Puerta B como salida
18.         movlw   b'10111111' ;RC7/Rx entrada
19.         movwf   TRISC      ;RC6/Tx salida
20.         movlw   b'00000110' ;Puerta A como E/S digitales
21.         movwf   ADCON1
22.         movlw   b'00000001' ;Puerta A, RA0 como entrada y el resto
                               ;como salidas
23.         movwf   TRISA
24.         movlw   d'25'
25.         movwf   SPBRG      ;9600 Baudios
26.         movlw   b'00100100' ;8 bits, habilita la transmisión
                               ;modo asíncrono y alta velocidad
27.         movwf   TXSTA
28.         bcf     STATUS,RP0 ;Banco 0
29.         movlw   b'10010000' ;Configuración del USART,
                               ;Habilita el Puerto Serial
30.         movwf   RCSTA      ;8 bits de recepción y
                               ;habilita la recepción continua
```

Lista 4.1 Configuración de los parámetros de comunicación

```
1.   bucle  bcf     STATUS,RP0      ;Banco 0
2.          bcf     STATUS,RP1

3.          btfss   PIR1,RCIF
4.          goto    proprin
5.          btfss   RCSTA,OERR      ; Buffer de recepción lleno
6.          goto    proprin

7.          movf    RCREG,W         ;Lectura del primer dato recibido
8.          movwf   PrimerBYTE
9.
10.         movf    RCREG,W         ;Lectura del segundo dato
                                    ;recibido
11.         movwf   SegundoBYTE

12.         bcf     RCSTA,CREN      ;Reinicia el bit CREN, OERR
13.         bsf     RCSTA,CREN

14.         btfss   PrimerByte,0    ;Bit0=1?
15.         goto    back7           ;No, Bit0=0 Banco 0 y Banco 1
16.         goto    back8           ;Sí,  Bit0=1 Banco 2 y Banco 3
17.  back7  movf     SegundoByte,w  ;Direccionamiento indirecto
18.         bcf     STATUS,IRP      ;No, Bit0=1 Banco 0 y Banco 1
19.         movwf   FSR
20.         goto    back9
21.  back8  movf    SegundoByte,w   ;Direccionamiento indirecto
22.         bsf     STATUS,IRP      ;No, Bit0=1 Banco 0 y Banco 1
23.         movwf   FSR
24.         goto    back9
25.  back9  btfss   PrimerByte,1    ;Bit1=1? Si es de lectura o
                                    ;escritura
26.         goto    read            ;No, Lectura de datos
27.         goto    write           ;Sí, Escritura de datos

28.  write  movlw   03h             ;Confirmación a la PC para el
                                    ;envío del dato hacia el PIC
29.         call    tx_ser1         ;Envío hacia la PC
```

Lista 4.2. Procedimiento para la lectura y escritura (*continúa...*)

```
30.  back6  btfss   PIR1,RCIF     ;Espera la respuesta de la PC,
                                  ;buffer de recepción lleno
31.         goto    back6
32.         movf    RCREG,W       ;Lectura del dato recibido
33.         movwf   INDF          ;Escritura del Dato
34.         goto    proprin

35.  read   movf    INDF,W        ;Lectura del dato recibido
36.         call    tx_ser1       ;Envío del dato correspondiente
37.         goto    proprin
37.  proprin goto   bucle         ;Adicionar el programa principal
                                  ;aquí
```

Lista 4.2 (*Continuación de la página 173.*) Procedimiento para la lectura y escritura

```
1.   tx_ser1 movwf  TXREG         ;Mueve el byte a transmitir
                                  ;al registro de transmisión
2.         bsf     STATUS,RP0    ;Banco 1
3.         bcf     STATUS,RP1
4.   back1 btfss   TXSTA,TRMT    ;Buffer vacío?, TRMT=1
5.         goto    back1         ;No

6.         bcf     STATUS,RP0    ;Banco 0
7.         return                ;Sí, Buffer vacío, dato
                                  ;transmitido
```

Lista 4.3. Procedimiento "tx_ser1" para la transmisión serial del registro W

4.3 Programación de la PC para la Comunicación Serial con el Microcontrolador

Una vez definido el protocolo de comunicación y la interface a usar con el microcontrolador es necesario desarrollar una librería encargada de la comunicación serial, para luego implementar las funciones que permitan seguir con las reglas definidas por el protocolo.

4.3.1 Diseño del objeto encargado de la Comunicación Serial

Este objeto es realizado utilizando el lenguaje de programación C++, ya que los servidores de OPC son implementados en el mismo lenguaje (ver la Especificación "OPC Common Definitions and Interfaces" versión 1.0). Para poder desarrollar el objeto encargado de la comunicación serial se utiliza las funciones del API Win 32 (ver el capítulo 3, del libro "Programación Avanzada en Win32").

El objeto encargado de la comunicación serial es implementado en la clase llamada "Serial232", el cual tiene la finalidad de brindar un mejor acceso y funcionalidad a la comunicación serial. Además, permite tener todas las propiedades funciones y procedimientos necesarios para una fácil comunicación. La figura 4.5 muestra el objeto con todas sus variables y funciones miembro implementadas, sus respectivas descripciones se puede apreciar en las tablas 4.2 y 4.3.

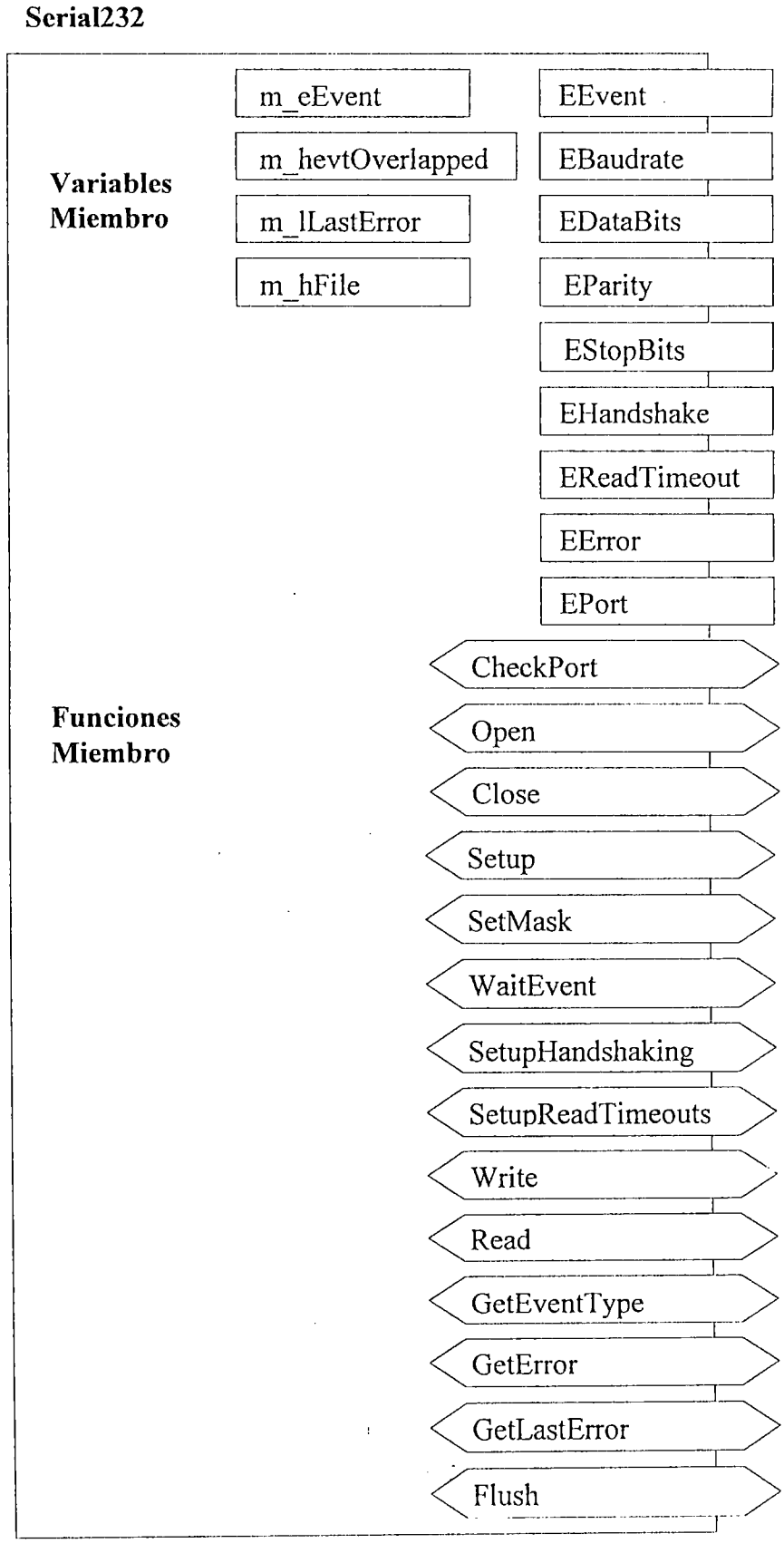


Figura 4.5 Diagrama del objeto *Serial232* (notación de Ege)

Nombre	Descripción
EEvent	Define las constantes para los eventos durante la comunicación.
EBaudrate	Define las constantes para la velocidad de transmisión (*Baudrate*).
EdataBits	Define las constantes para el número de Bits de transmisión.
Eparity	Define las constantes para la configuración de la Paridad.
EStopBits	Define las constantes para la configuración de Bits de parada.
Ehandshake	Define las constantes para la configuración del protocolo (RTS/CTS, XON/XOFF, etc.)
EreadTimeout	Define las constantes para la configuración del Timeout.
EError	Define las constantes para los errores de Comunicación.
EPort	Constantes para el estado del Puerto (puerto disponible, puerto en uso, etc.)
m_lLastError (variable privada)	Variable para el último error.
m_hFile (variable privada)	Manejador de objeto.
m_eEvent (variable privada)	Variable para el tipo de evento.
m_hevtOverlapped (variable privada)	Manejador para operaciones internas.

Tabla 4.2 Descripción de las variables públicas y privadas del objeto *Serial232*

Nombre	Descripción
CheckPort	Verifica si un puerto en particular está disponible.
Open	Abre un puerto de comunicación serial y especifica la medida de las colas de entrada y salida.
Close	Cierra el puerto de comunicación serial.
Setup	Establece los parámetros de comunicación tales como: *baudrate*, *databits*, *parity* y *stopbits*.
SetMask	Habilita los eventos que deben ser supervisados por el método *WaitEvent.*
WaitEvent	Espera uno de los eventos que han sido habilitados previamente por la función *SetMask.*
SetupHandshaking	Establece el tipo de protocolo (RTS/CTS, XON/XOFF, etc.)
SetupReadTimeouts	Establece el tipo de operación de lectura, el cual puede ser: Bloqueada, el cual causará que el método *Read* no retorna hasta que el número de bytes pedidos haya sido leído. No Bloqueada, el cual lee todos los bytes existentes en el buffer y retorna inmediatamente.
Write	Escribe los datos por el puerto serial.
Read	Lee los datos del puerto serial.
GetEventType	Obtiene la causa del evento.
GetError	Obtiene el error.
GetLastError	Obtiene el estado del ultimo error.
Flush	Elimina todos los *buffers* de transmisión y recepción.

Tabla 4.3 Descripción de las funciones públicas del objeto *Serial232*

4.3.2 Implementación del objeto encargado de la Comunicación Serial

La lista 4.4 muestra la declaración de las funciones y constantes de la clase *Serial232*. En la línea 1 de la lista 4.4 se define la clase Serial232, en las líneas 4 al 22 se definen las constantes descritas en la tabla 4.2. La definición de las funciones públicas descritas en la tabla 4.3 es mostrada en las líneas 25 al 35 en la lista 4.4.

```
1.  class CSerial232
2.  {

3.  public:
4.      typedef enum
        {
          EEventNone  = -1,              // Evento sin causa
          EEventBreak = EV_BREAK, // Señal de interrupción recibida
          EEventCTS   = EV_CTS,      // La señal CTS cambio de estado
          EEventDSR   = EV_DSR,      // La señal DSR cambio de estado
          EventError = EV_ERR,       // Un error line-status ha ocurrido
          EventRing   = EV_RING,     // Detección de llamada
          EEventRLSD  = EV_RLSD,  // La señal RLSD cambio de estado
          EEventRecv  = EV_RXCHAR,  // El dato es recibido en la entrada
          EEventRcvEv = EV_RXFLAG, //Evento carácter fue recibido en
                                    //la entrada
          EEventSend  = EV_TXEMPTY, //El último carácter de la salida
                                    //fue enviado
        }
5.      EEvent;

6.      typedef enum
        {
          EBaudUnknown = -1,            // No definido
          EBaud110    = CBR_110,        // 110 bits/sec
          EBaud300    = CBR_300,        // 300 bits/sec
          EBaud600    = CBR_600,        // 600 bits/sec
          EBaud1200   = CBR_1200,       // 1200 bits/sec
          EBaud2400   = CBR_2400,       // 2400 bits/sec
          EBaud4800   = CBR_4800,       // 4800 bits/sec
          EBaud9600   = CBR_9600,       // 9600 bits/sec
          EBaud14400  = CBR_14400,      // 14400 bits/sec
          EBaud19200  = CBR_19200,      // 19200 bits/sec
          EBaud38400  = CBR_38400,      // 38400 bits/sec
          EBaud56000  = CBR_56000,      // 56000 bits/sec
          EBaud57600  = CBR_57600,      // 57600 bits/sec
          EBaud115200 = CBR_115200,     // 115200 bits/sec
          EBaud128000 = CBR_128000,     // 128000 bits/sec
          EBaud256000 = CBR_256000,     // 256000 bits/sec
        }
7.      EBaudrate;
```

Lista 4.4 Definición de las funciones y constantes de la clase *Serial232*

(*continúa...*)

```
8.      typedef enum
        {
          EDataUnknown = -1,          // No definido
          EData5     = 5,             // 5 bits por byte
          EData6     = 6,             // 6 bits por byte
          EData7     = 7,             // 7 bits por byte
          EData8     = 8              // 8 bits por byte
        }
9.      EDataBits;

10.     typedef enum
        {
          EParUnknown = -1,           // No definido
          EParNone  =  NOPARITY,      // Sin paridad
          EParOdd   = ODDPARITY,      // Impar
          EParEven  = EVENPARITY,     // Par
          EParMark  = MARKPARITY,     // Marca
          EParSpace = SPACEPARITY     // Espacio
        }
11.     EParity;

12.     typedef enum
        {
          EStopUnknown = -1,          // No definido
          EStop1     = ONESTOPBIT,    // 1 bit de parada
          EStop1_5   = ONE5STOPBITS,  // 1.5 bit de parada
          EStop2     = TWOSTOPBITS    // 2 bits de parada
        }
13.     EStopBits;

14.     typedef enum
        {
          EHandshakeUnknown  = -1,    // No definido
          EHandshakeOff      = 0,     // Sin Protocolo handshaking
          EhandshakeHardware = 1,     // Protocolo Hardware
                                      // handshaking (RTS/CTS)
          EhandshakeSoftware = 2,     // Protocolo Software
                                      // handshaking (XON/XOFF)
          EhandshakeHardSoftware = 3  // Ambos Protocolos
                                      // (RTS y XON/XOFF)
        }
15.     EHandshake;
```

Lista 4.4 (*Continuación de la página 180.*) Definición de las funciones y constantes de la clase *Serial232* (*continúa...*)

```
16.     typedef enum
        {
          EReadTimeoutUnknown    = -1,      // No definido
          EreadTimeoutNonblocking =  0,     // Retornar
                                            // inmediatamente
          EreadTimeoutBlocking    = 1       // Espera o bloquea hasta
                                            // que todo es recuperado
        }
17.     EReadTimeout;

18.     typedef enum
        {
          EErrorUnknown = 0,                // No definido
          EErrorBreak     = CE_BREAK,       // Señal de interrupción
                                            //recibida
          EErrorFrame     = CE_FRAME,       // Error de trama (frame)
          EErrorIOE       = CE_IOE,         // Error del dispositivo
          EErrorMode      = CE_MODE,        // Modo no soportado
          EErrorOverrun   = CE_OVERRUN,// Pérdida de información en el
                                            //puerto
          EErrorRxOver    = CE_RXOVER,      // Desbordamiento del buffer
                                            // de recepción
          EErrorParity    = CE_RXPARITY,// Error de paridad
          EErrorTxFull    = CE_TXFULL       // Buffer de transmisión lleno.
        }
19.     EError;

20.     typedef enum
          {
          EPortUnknownError = -1,       // No definido
          EPortAvailable    =  0,       // El puerto es disponible
          EPortNotAvailable =  1,       // El puerto no está presente
          EPortInUse        =  2        // El puerto está en uso
          }
21.     EPort;


22.   public:
23.       CSerial232();
24.       virtual ~CSerial232();
```

Lista 4.4 (*Continuación de la página 181.*) Definición de las funciones y constantes de la clase *Serial232* (*continúa...*)

```
25.  public:

26.      EPort CheckPort (LPCTSTR lpszDevice);

27.      virtual LONG Open (LPCTSTR lpszDevice,
                            DWORD dwInQueue = 2048,
                            DWORD dwOutQueue = 2048);

28.      virtual LONG Close (void);

29.      virtual LONG Setup (EBaudrate eBaudrate = EBaud9600,
                            EDataBits eDataBits = EData8,
                            EParity   eParity   = EParNone,
                            EStopBits eStopBits = EStop1);

30.      virtual LONG SetMask (DWORD dwMask =
                                EEventBreak|EEventError|EEventRecv);

31.      virtual LONG WaitEvent (LPOVERLAPPED lpOverlapped = 0,
                                 DWORD dwTimeout = INFINITE);

32.      virtual LONG SetupHandshaking (EHandshake eHandshake);

33.      virtual LONG SetupReadTimeouts (EReadTimeout eReadTimeout);

34.      virtual LONG Write (const void* pData, size_t iLen,
                            DWORD* pdwWritten = 0,
                            LPOVERLAPPED lpOverlapped = 0,
                            DWORD dwTimeout = INFINITE);

35.      virtual LONG Read (void* pData, size_t iLen,
                            DWORD* pdwRead = 0,
                            LPOVERLAPPED lpOverlapped = 0,
                            DWORD dwTimeout = INFINITE);
```

Lista 4.4 (*Continuación de la página 182.*) Definición de las funciones y constantes de la clase *Serial232* (*continúa...*)

```
36.         EEvent GetEventType (void);
37.         EError GetError (void);
38.         LONG GetLastError (void) const    { return m_lLastError; }
39.         LONG Flush (void);
40.   protected:
41.         class CDCB : public DCB
            {
42.         public:
43.         CDCB() { DCBlength = sizeof(DCB); }
            };
44.   protected:
45.         LONG        m_lLastError;
46.         HANDLE      m_hFile;
47.         EEvent      m_eEvent;
48.         HANDLE      m_hevtOverlapped;
      };
49.   #endif
```

Lista 4.4 (*Continuación de la página 183.*) Definición de las funciones y constantes de la clase *Serial232*

4.3.3 Implementación del Protocolo de Comunicación con el Microcontrolador

Sobre la base del protocolo definido en la figura 4.2 y las funciones encargadas de la comunicación serial de la tabla 4.3, se implementa las funciones que permitan las operaciones de lectura y escritura con el microcontrolador. Dichas operaciones son mostradas como algoritmos en las figuras 4.6 y 4.7.

El algoritmo de la figura 4.6 corresponde a la siguiente función:

```
bool ReadPic (int Bank, int AdressData, int *pReadData)        (4.1)
```

La función (4.1) permite realizar el procedimiento de lectura de los registros de memoria del microcontrolador y sus parámetros ingresados son:

Bank	El Banco de memoria del registro a leer.
AdressData	La dirección de memoria del registro a leer.
pReadData	Un puntero que devuelve el dato leído

El algoritmo de la figura 4.7 corresponde a la función siguiente:

bool WritePic (int Bank, int AdressData, int WriteData) (4.2)

La función (4.2) permite una operación de escritura a un registro de memoria del microcontrolador y sus parámetros ingresados son:

Bank	El Banco de memoria del registro a escribir.
AdressData	Dirección de memoria del registro a escribir
WriteData	Nuevo valor que se va ha escribir en el registro de memoria

La función (4.4) establece la comunicación con los siguientes parámetros:

- Puerto: COM1
- *BaudRate* :9600
- Número de Bits: 8
- Paridad: None
- Bit de parada:1.

bool SetConnection() (4.3)

Esta función (4.4) termina la comunicación y libera la memoria creada al establecer la conexión por medio de la función *SetConnection* (4.3).

bool CutConnection() (4.4)

Si las funciones (4.1), (4.2), (4.3) y (4.4) terminan satisfactoriamente retornan verdadero, en caso contrario retorna falso.

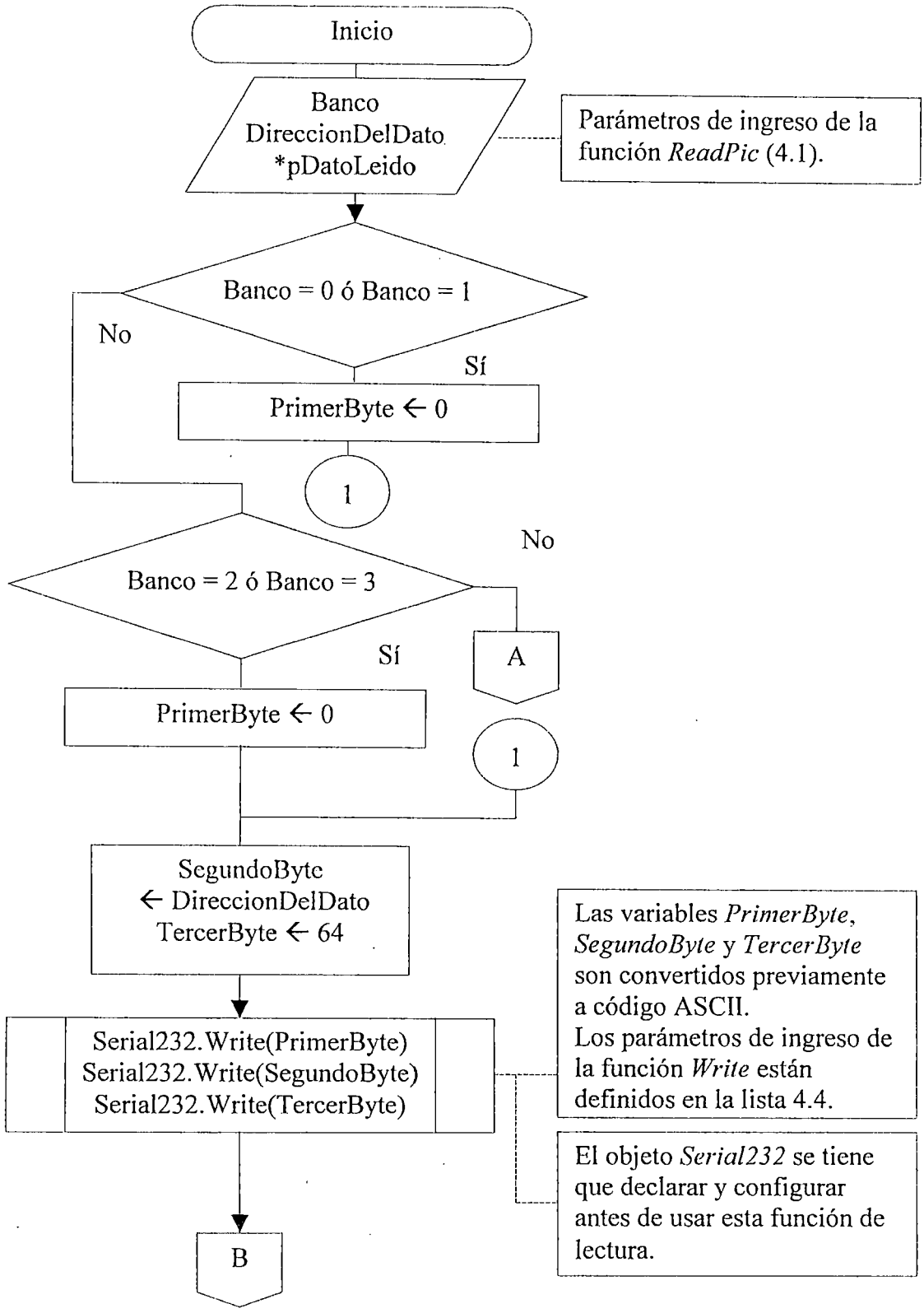


Figura 4.6 Algoritmo para la operación de lectura con el microcontrolador

(*continúa...*)

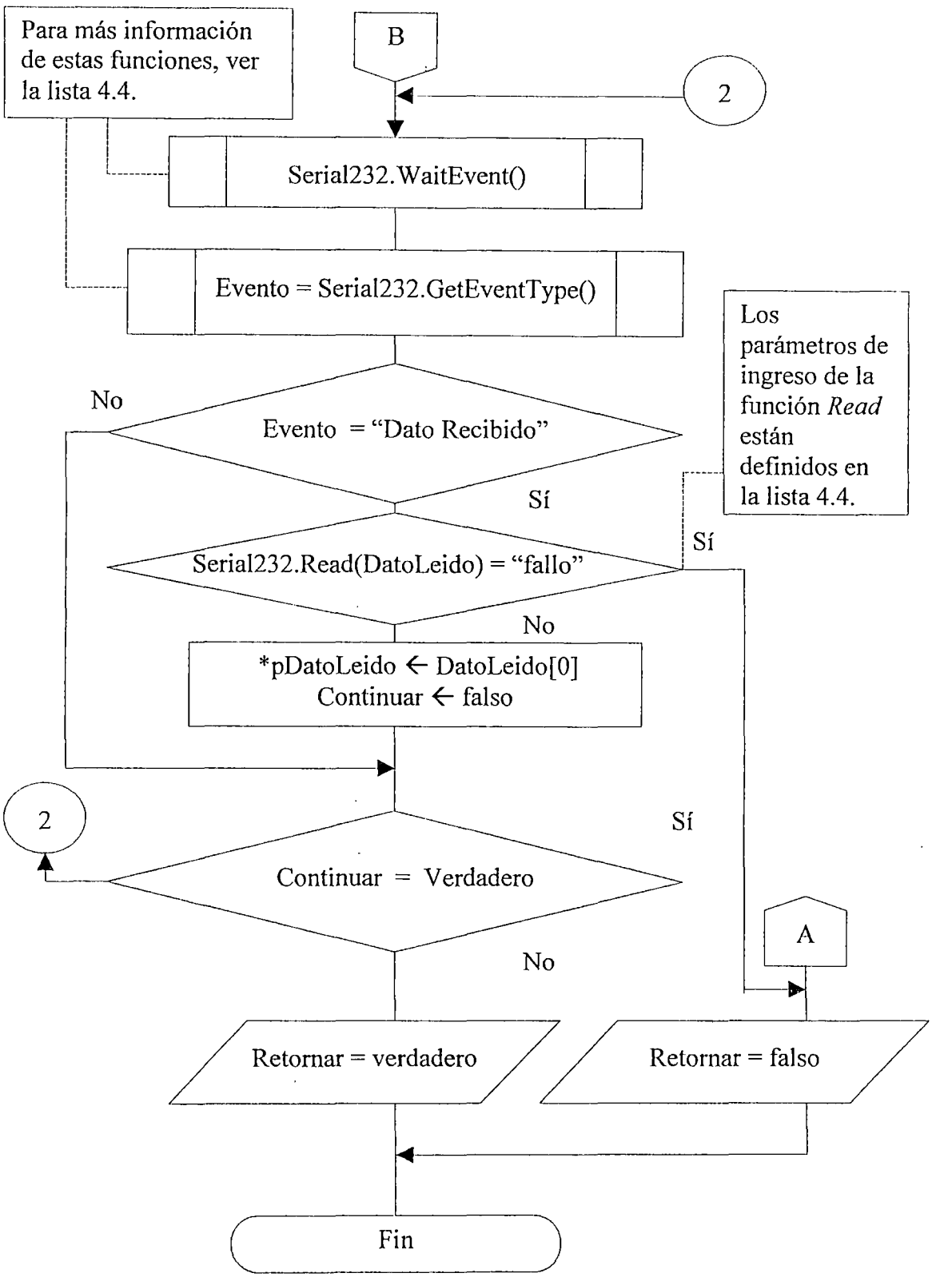


Figura 4.6 (*Continuación de la página 188.*) Algoritmo para la operación de lectura con el microcontrolador

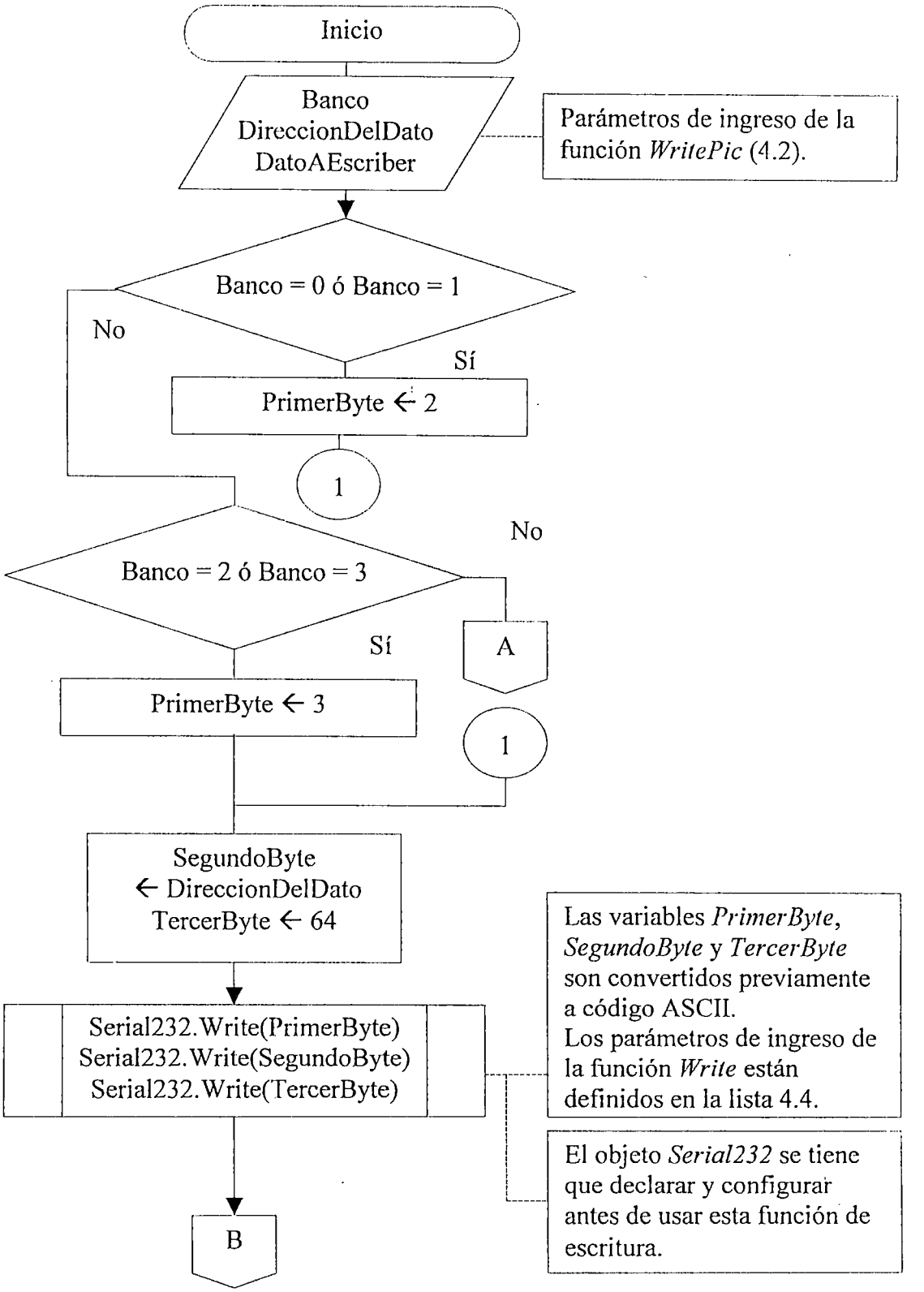


Figura 4.7 Algoritmo para la operación de escritura con el microcontrolador

(*continúa...*)

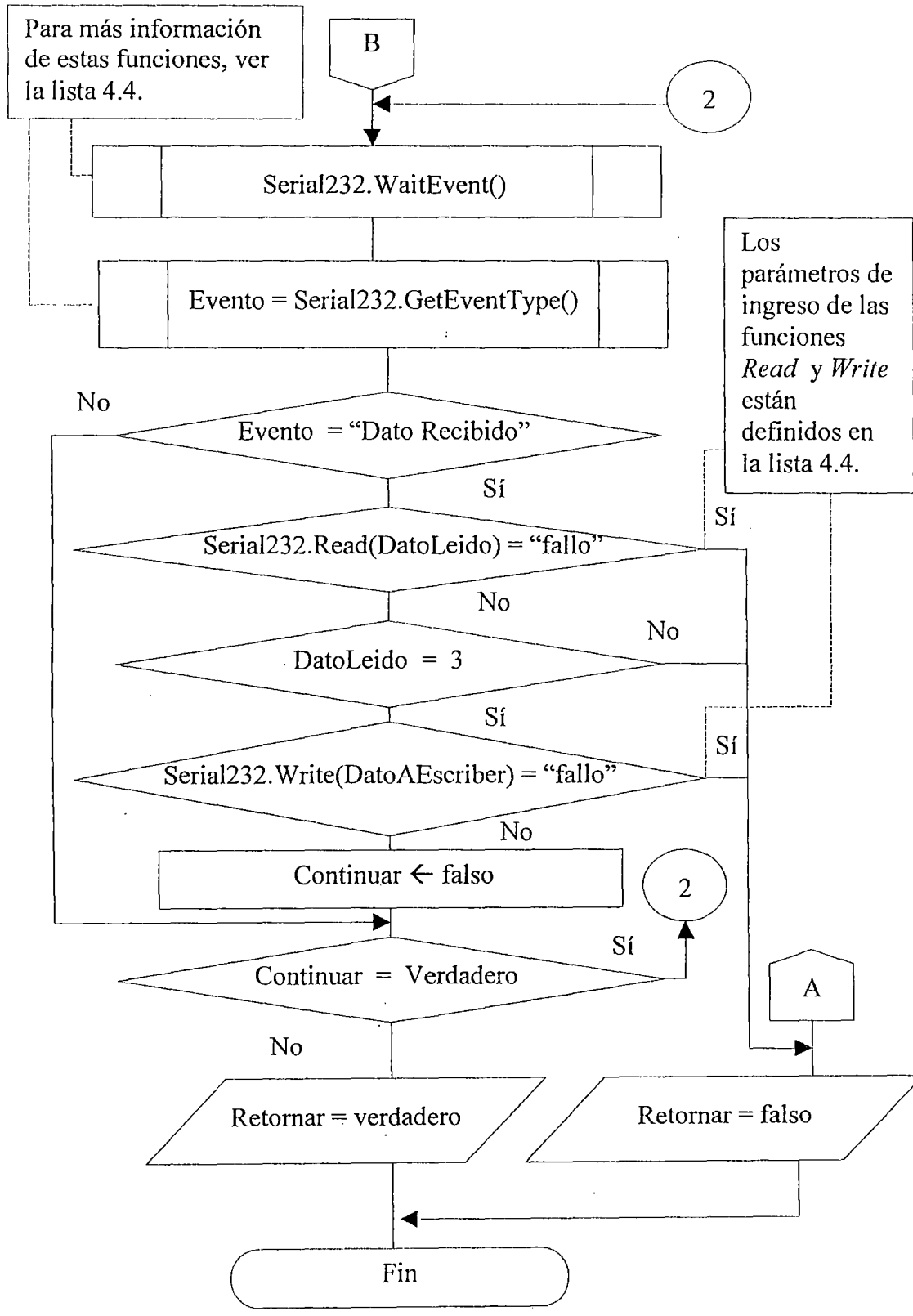


Figura 4.7 (*Continuación de la página 190.*) Algoritmo para la operación de escritura con el microcontrolador

4.4 Implementación del Servidor de OPC para el Microcontrolador

La implementación del Servidor de OPC para el Microcontrolador se realizó utilizando la herramienta (*toolkit*) *Xpress OPC Server* de la marca *Softing*, el cual encapsula la funcionalidad de OPC y DCOM.

4.4.1 La Herramienta (*Toolkit*) *Softing XPress OPC Server*

Esta herramienta provee la funcionalidad de un Servidor de OPC con la especificación Acceso de Datos de OPC versión 2.05 de la Fundación OPC. Esta funcionalidad incluye las partes que son comunes para todos los Servidores de OPC, además de la implementación de objetos de OPC y sus interfaces (*OPCServer*, *OPCGroup*, *OPCBrowser*, etc.) La definición de las funciones y estructuras es encuentra en el Apéndice C.

4.4.1.1 Arquitectura

La figura 4.8 muestra la arquitectura de la herramienta *XPress OPC Server*, el cual necesita externamente lo siguiente:

- Hardware Para este caso será el microcontrolador PIC 16F873 que suministra los dato.
- Librería DLL Para este caso será la Librería llamada "Serial232"

Las notificaciones a los clientes y llamado de las funciones son controladas por la herramienta. Este a su vez entrega la funcionalidad del acceso de datos a los clientes de OPC.

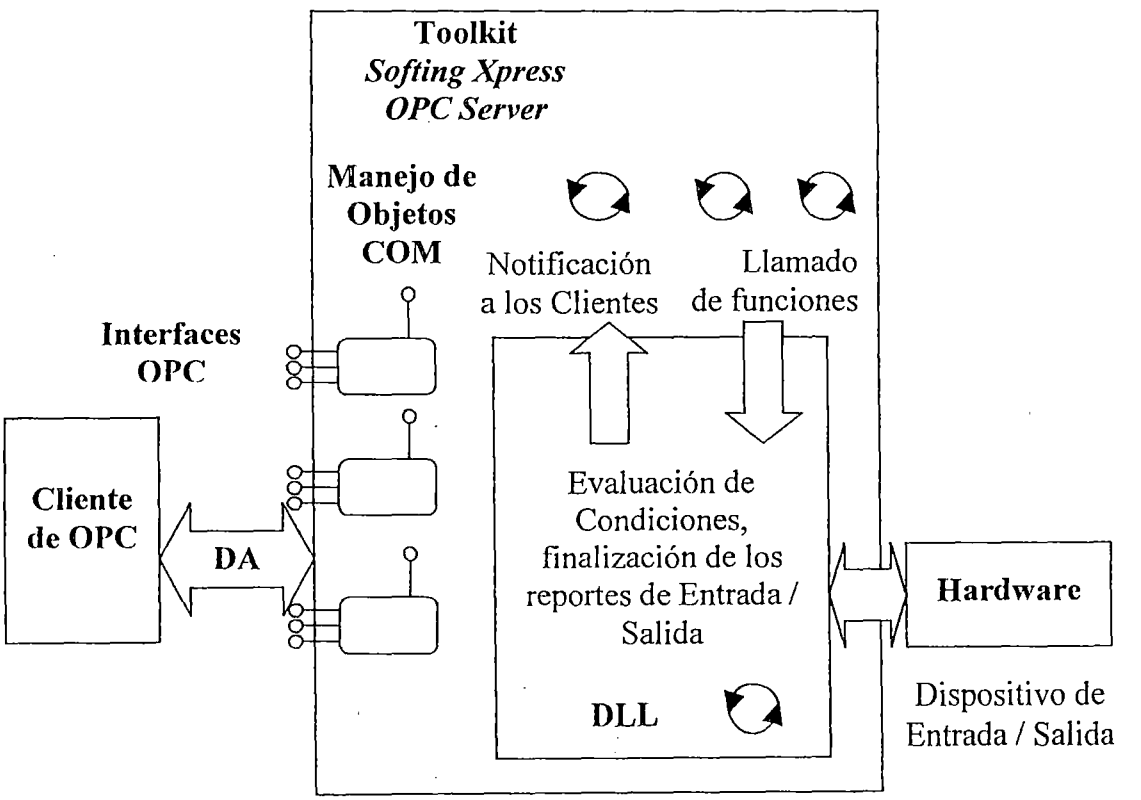


Figura 4.8 Arquitectura general de la herramienta *Softing XPress OPC Server*

En la tabla 4.4 se podrán apreciar todas las interfaces que soporta esta herramienta, y que ayudará a tener presente el alcance que tendrá el servidor de OPC desarrollado con esta herramienta. Para más información de las interfaces y objetos de OPC ver la Especificación "Data Access Custom Interface Standard" versión 2.05".

Objetos	**Interfaces**
OPCServer	IOPCCommon IOPCServer IOPCItemProperties IOPCServerPublicGroups IOPCServerBrowseServerAddressSpace IConnectionPointContainer IOPCShutdown
OPCGroup	IOPCItemMgt IOPCGroupStateMgt IOPCPublicGroupStateMgt IOPCSyncIO IOPCAsyncIO IOPCAsyncIO2 IDataObject IEnumOPCItemAttributes IConnectionPointContainer IOPCDataCallback

Tabla 4.4 Objetos e interfaces que soporta la herramienta *Softing XPress OPC Server*

4.4.1.2 Ambientes de Desarrollo

La herramienta permite el uso de los siguientes ambientes de desarrollo:

- Microsoft Visual C++ V6.0
- Borland Delphi V5.0
- Borland C++ Builder V4.0

El Servidor de OPC para el PIC 16F873 fue desarrollado usando el ambiente de desarrollo Microsoft Visual C++ V6.0.

4.4.2 Procedimiento para la Implementando de la funcionalidad del Acceso de Datos de OPC

El procedimiento para esta implementación consta de las siguientes tareas:

- El *namespace* debe de ser configurado. Esto incluye la creación de los nodos y items o *tags* y también la definición de sus propiedades.
- Se debe de proveer los datos provenientes desde el dispositivo (Hardware o *driver* de un dispositivo) al servidor.

4.4.2.1 Configuración del *Namespace*

La configuración del *namespace* es llevada a cabo por la función *SOXpSBuildDANameSpace* (ver función C.4 del Apéndice C). Esto permite la creación de los nodos y los *tags* así como sus propiedades. La función *SOXpSCreateTag* (ver función C.7 del Apéndice C) es llamada para crear un objeto *tag* en el *namespace* y la función *SOXpSCreateNode* (ver función C.6 del Apéndice C) es llamada para crear un objeto nodo en el *namespace*.

4.4.2.2 Manejo de los Pedidos para el Acceso de Datos

La herramienta *Softing XPress OPC Server* tiene un mecanismo para la implementación del intercambio de datos entre la funcionalidad de OPC y una librería dinámica el cual es implementado para el dispositivo de entrada y salida. Este intercambio de datos es realizado por medio del llamado pedido de datos. Un pedido de datos es entregado por la aplicación servidor de OPC a la librería dinámica cuando el cliente de OPC necesita leer o escribir un dato de un ítem.

El pedido de datos es entregado a la función *SOXpSHandleDARequests* (ver función C.5 del Apéndice C) en paquetes. Cada pedido debe de ser procesado por la librería dinámico o DLL. Este puede ser manejado inmediatamente sobrescribiendo el último valor con el nuevo y poniendo el estado del pedido en "handled".

Si el pedido no pudo ser procesado por algún motivo, se debería de poner su estado a "handled" y retornar un código de error E_FAIL en el variable miembro m_result (ver tabla C.1 del Apéndice C). También se puede usar la variable miembro m_quality (ver tabla C.1 del Apéndice C) para especificar la información del error, en las tablas C.4, C.5 y C.6 del Apéndice C se puede apreciar los valores que puede tomar la variable miembro m_quality con su respectivo significado.

La herramienta *Softing XPress OPC Server* ofrece tres tipos de operación para el acceso de datos, los cuales son:

- Modo de Comunicación Síncrona
- Modo de Comunicación Asíncrona
- Modo de Comunicación por Evento (*Event-driven I/O*)

Para más información de cada tipo de operación se debe consultar con el manual de usuario de la herramienta.

El servidor de OPC para el microcontrolador usa el modo de comunicación Síncrona por las siguientes razones:

- El tiempo de acceso al dispositivo es considerablemente pequeño (ver tabla 4.1).
- El dispositivo es pasivo, es decir los datos se deben adquirir desde él explícitamente.
- Las operaciones de lectura y escritura son síncronas, es decir la operación termina cuando la llamada de las funciones retornan.

4.4.3 Procedimiento Seguido en la Implementación del Servidor de OPC para el Microcontrolador PIC

La implementación del Servidor de OPC para el microcontrolador se llevó a cabo de la siguiente manera:

- Iniciación del servidor de OPC
- Implementación del *namespace*
- Implementación para el Pedido de Escritura y Lectura
- Finalización del Servidor
- Implementación de seguridad

Todos los procedimientos siguientes fueron implementados en el archivo "OPCServerPic.cpp".

4.4.3.1 Iniciación del Servidor

Antes de inicializar el servidor, se debe declarar la librería que se va ha utilizar para la comunicación serial (Serial232.h). Luego se debe declarar las variables y funciones globales que va ha utilizar, tal como se muestra en la lista 4.5.

```
1.  ....
2.  //-------------------------------------------------------------------------
3.  // Globals
4.  //-------------------------------------------------------------------------
5.  //Comunicación Serial
6.  //Definición de las variables para la comunicación con el PICF16F873
7.  CSerial232 serial;

8.  int ShowError (LONG lError, LPCTSTR lptszMessage);
9.  bool  SetConnection();
10. bool CutConnection();
11. bool  ReadPic (int Bank, int AdressData, int *pReadData);
12. bool WritePic (int Bank, int AdressData, int WriteData);
13.  ...
```

Lista 4.5. Declaración de las variables y funciones necesarias para la comunicación serial con el microcontrolador PIC

En la línea 7 del listado 4.5 se muestra la declaración del objeto *serial* el cual será usado por la funciones *SetConnection* (ver función 4.3), *CutConnection* (ver función 4.4), *ReadPic* (ver función 4.1), *WritePic* (ver función 4.2). Estas funciones son declaradas en las líneas 8 al 12 de la lista 4.5.

La función *SetConnection* es inicializado dentro de la función *SOXpSStart* (ver función C.2 del Apéndice C), ya que esta función se activará cuando el Servidor ha sido puesto en marcha.

4.4.3.2 Implementación del *Namespace*

La organización del *namesapce* del servidor de OPC para el microcontrolador es del tipo árbol o jerárquico, tal como se puede apreciar en la figura 4.9. El algoritmo para implementar este *namespace* es mostrado en la figura 4.10.

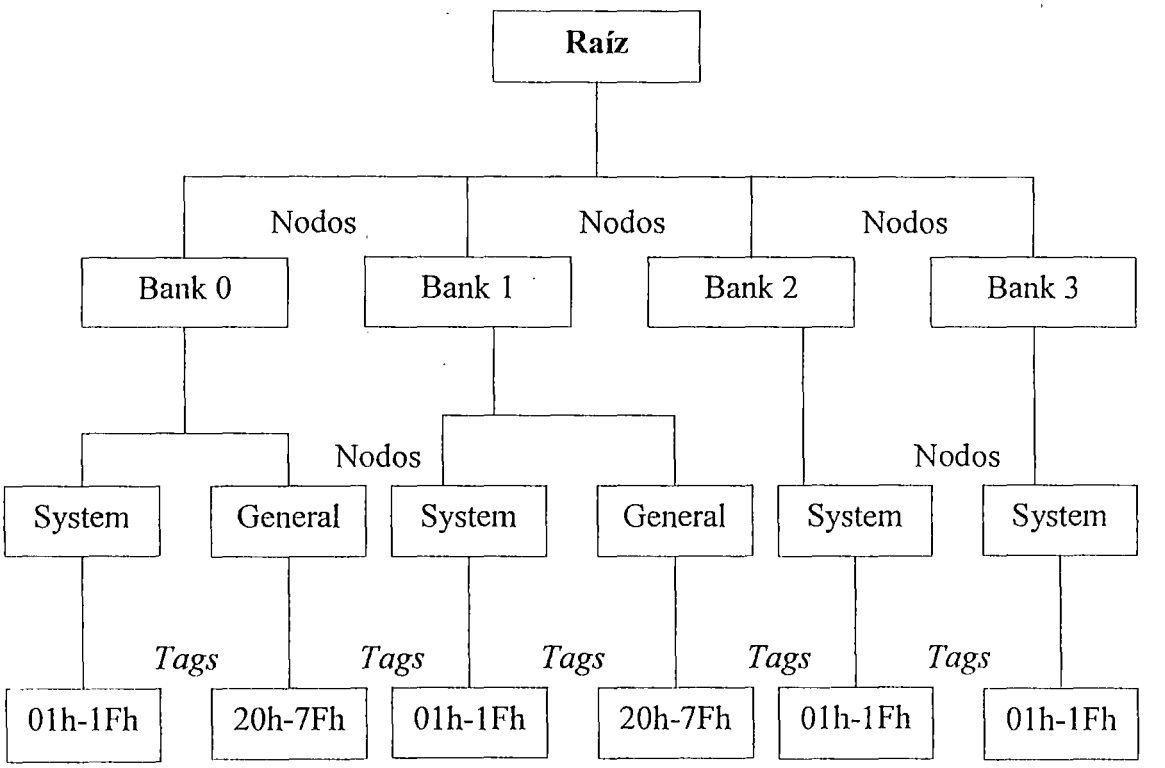


Figura 4.9. Esquema general del *namespace* que se implementará en el Servidor de OPC para el microcontrolador

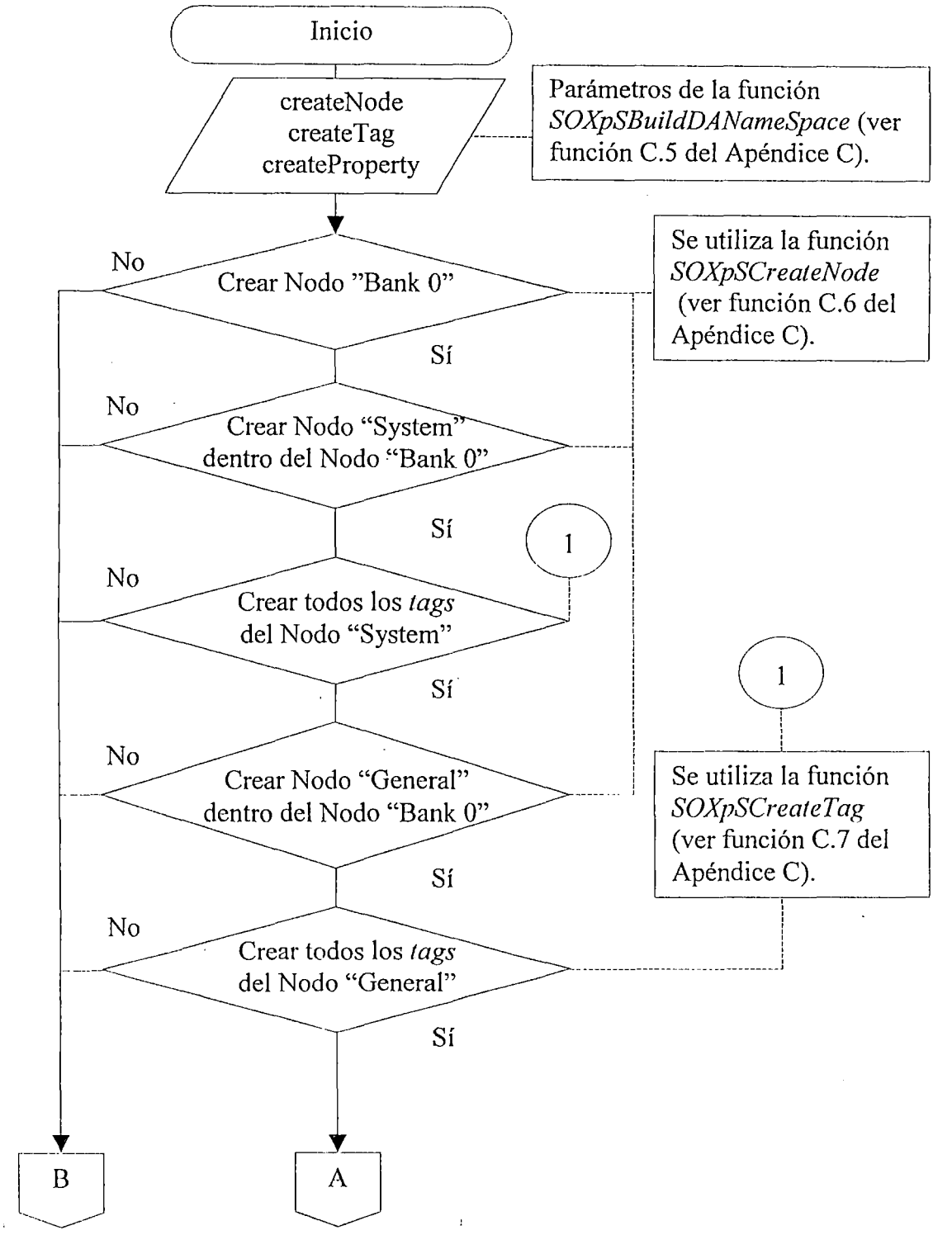


Figura 4.10 Algoritmo de la función *SOXpSBuildDANameSpace* para la creación del *namespace* de la figura 4.9 (*continúa...*)

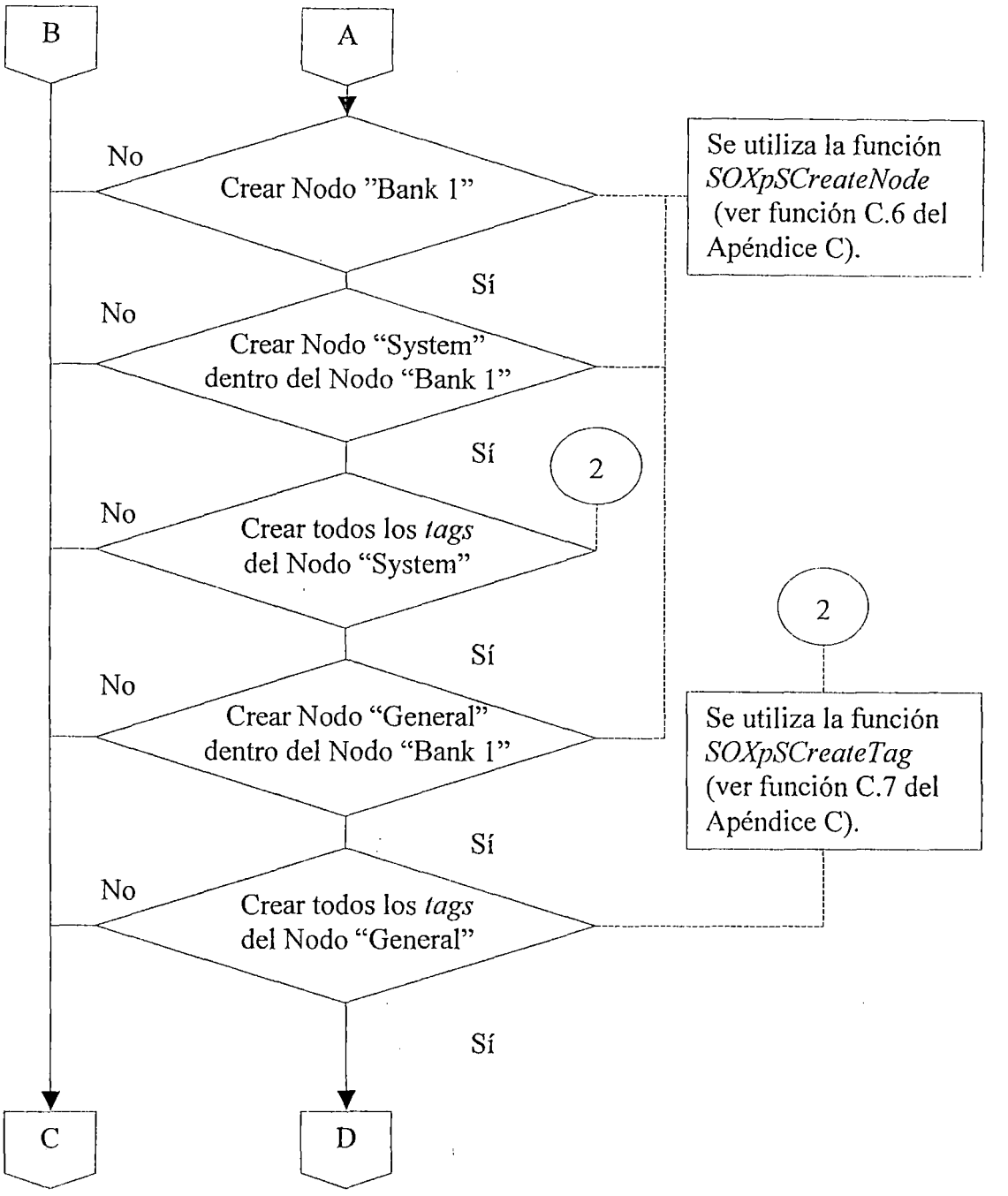


Figura 4.10 (*Continuación de la página 202.*) Algoritmo de la función *SOXpSBuildDANameSpace* para la creación del *namespace* de la figura 4.9

(*continúa...*)

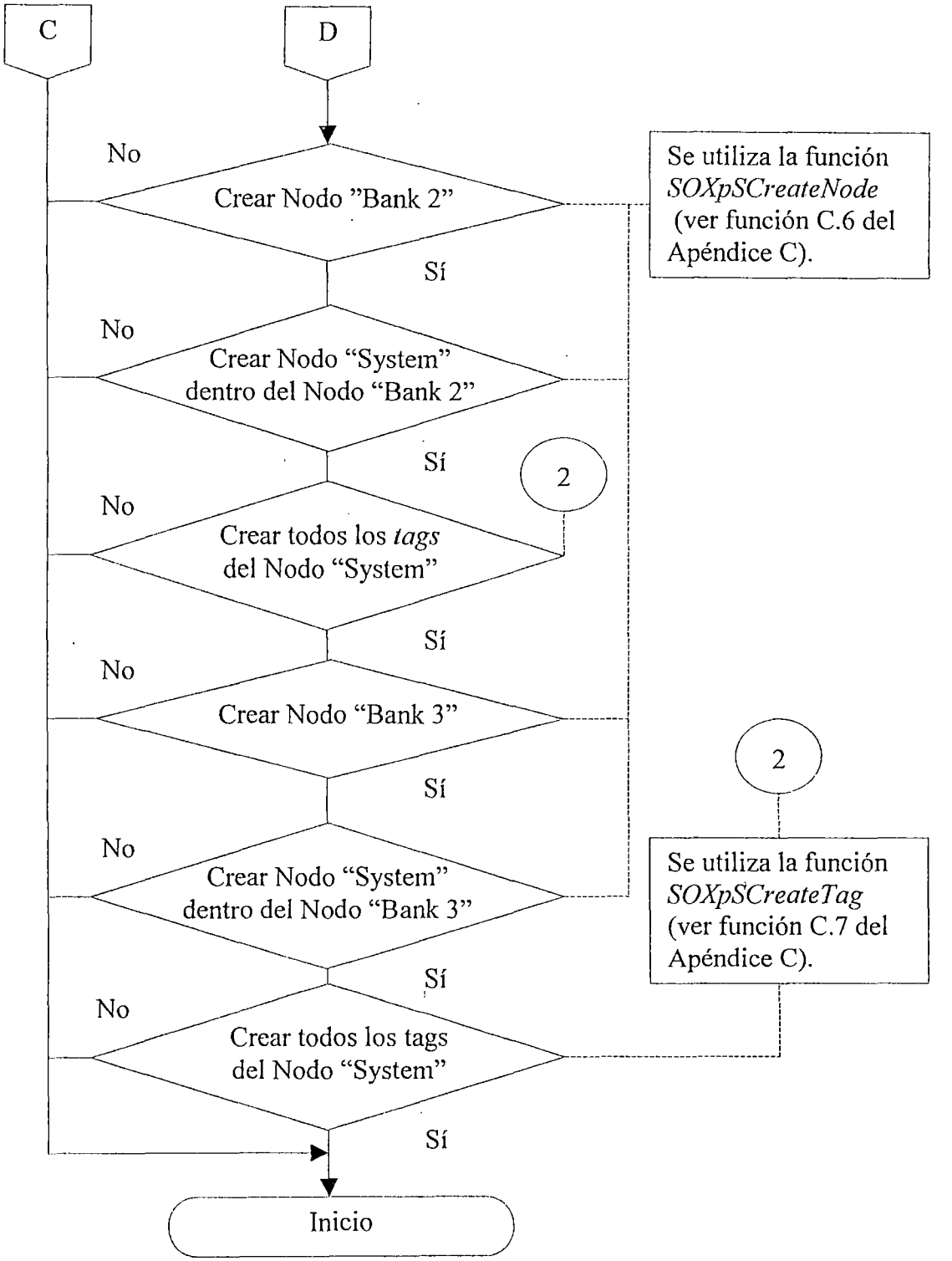


Figura 4.10 (*Continuación de la página 203.*) Algoritmo de la función *SOXpSBuildDANameSpace* para la creación del *namespace* de la figura 4.9

En la lista 4.6 se puede apreciar que se utiliza la función *SOXpSBuildDANameSpace* (ver función C.5 del Apéndice C), el cual permite crear el *namespace*, en las líneas 2 al 5 se define las variables que permitirán crear los nodos, *tags* y propiedades.

En la línea 8 de la lista 4.6 se crea el primer nodo "Bank 0" y en la línea 12 su respectiva propiedad, luego se crea el nodo siguiente "System" en la línea 16, el cual pertenece al nodo "Bank 0". Finalmente se crea los items pertenecientes al nodo "System" del nodo "Bank 0", en la línea 18 se crea el ítem "TMR0" para ello se usa la función *SOXpSCreateTag* (ver función C.7 del Apéndice C)

Este mismo procedimiento es utilizado para implementar todos los nodos y *tags* restantes del *namespace* de la figura 4.9.

```
1.  void SOXPSCFG_CALL SOXpSBuildDANameSpace(
        IN SOXpSCreateNode createNode,        // función para la creación
                                              // de nodos
        IN SOXpSCreateTag createTag,          // función para la creación
                                              // de tags
        IN SOXpSCreateProperty createProperty) // función para la creación
                                               // de propiedades
{
        // Definición de las variables
2.      SOXpSNodeHandle node1;
3.      SOXpSNodeHandle node2;
4.      SOXpSItemHandle tag;
5.      SOXpSItemHandle property;
6.      VARIANT descrPropVal;

7.      //-- Rama Banck 0 --
8.      if (!(createNode(0, _T("Bank 0"), &node1)))
9.      return;
10.     // Descripción del nodo Bank 0
11.     descrPropVal.vt = VT_BSTR;
12.     descrPropVal.bstrVal = SysAllocString(L"Todos los registros del
        Banco 0 del PIC 16F873");
13.     if (!(createProperty((SOXpSNodeOrItemHandle)node1, 101, NULL,
        VT_BSTR,SOXPS_ACCESSRIGHT_READABLE,&descrPropVal,
        0, &property)))
14.     return;
15.     VariantClear(&descrPropVal);

16.     if (!(createNode(node1, _T("System"), &node2)))
17.     return;

18.     if (!(createTag(node2, _T("TMR0"), VT_I2,
        SOXPS_ACCESSRIGHT_READABLE |
19.     SOXPS_ACCESSRIGHT_WRITEABLE, 0, &tag)))
20.     return;
21.     if (!(createTag(node2, _T("PCL"), VT_I2,
        SOXPS_ACCESSRIGHT_READABLE , 0, &tag)))
22.     return;
23.     if (!(createTag(node2, _T("STATUS"), VT_I2,
        SOXPS_ACCESSRIGHT_READABLE , 0, &tag)))
24.     return;
25. ...
```

Lista 4.6. Parte de la implementación de la función *SOXpSBuildDANameSpace* para la creación del *namespace* de la figura 4.9

4.4.3.3 Implementación del Pedido de Escritura y Lectura

Una vez inicializado el Servidor de OPC y configurado el *namespace*, es necesario implementar el acceso de datos. El modo de comunicación usado por el Servidor de OPC para el microcontrolador es el síncrono.

El algoritmo de la figura 4.11 muestra el manejo de los pedidos de lectura y escritura implementados en la función *SOXpSHandleDARequests* (ver función C.5 del Apéndice C). La implementación del algoritmo de la figura 4.11 se encuentra en las listas 4.7 y 4.8. La lista 4.7 se encarga del manejo de los pedidos de lectura y la lista 4.8 de los pedidos de escritura.

El procedimiento realizado para las operaciones de lectura y escritura de los items TMR0, PCL y STATUS es similar para el resto de items existentes dentro del *namespace*.

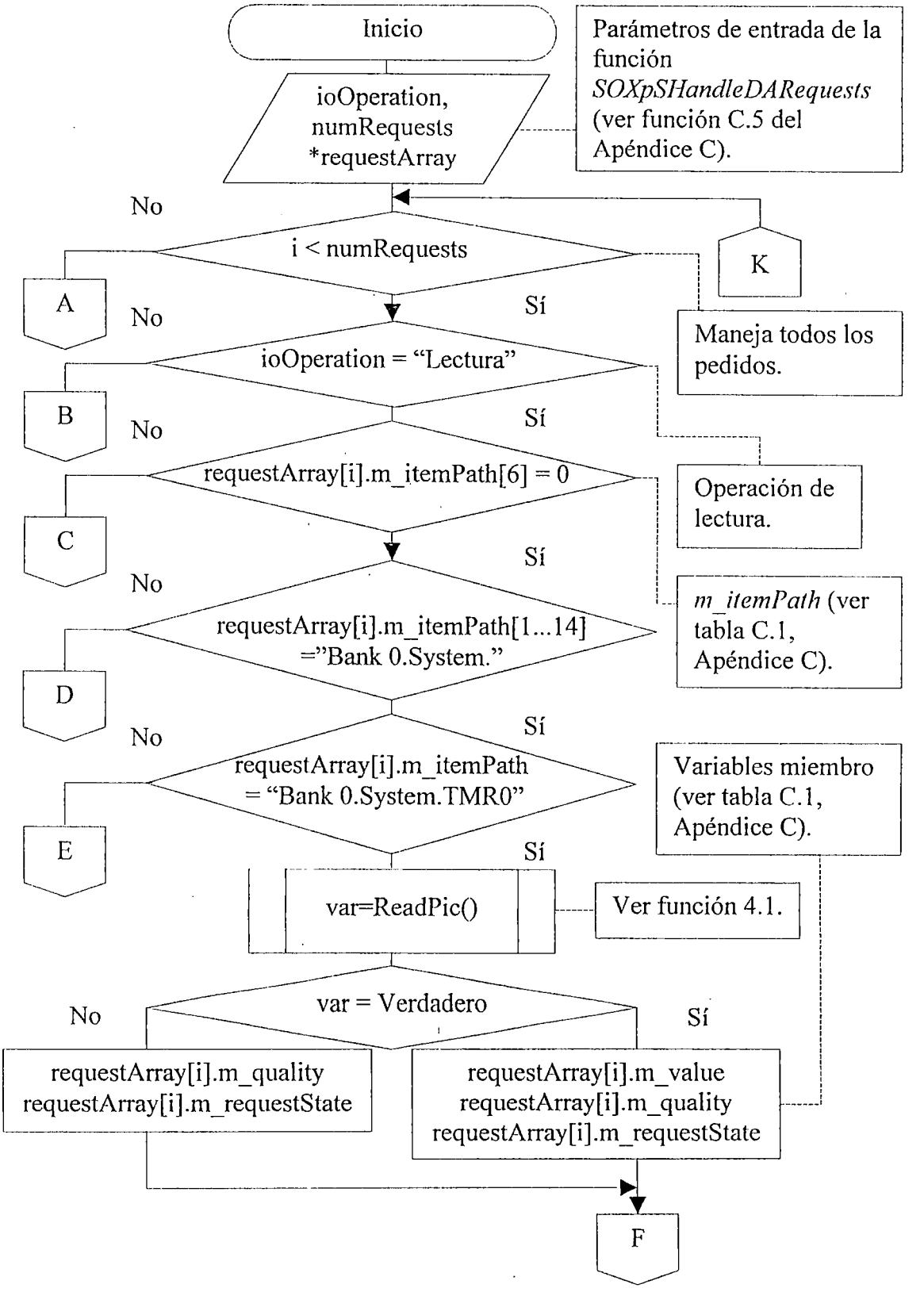


Figura 4.11 Algoritmo de la función *SOXpSHandleDARequests* para el pedido de lectura y escritura (*continúa...*)

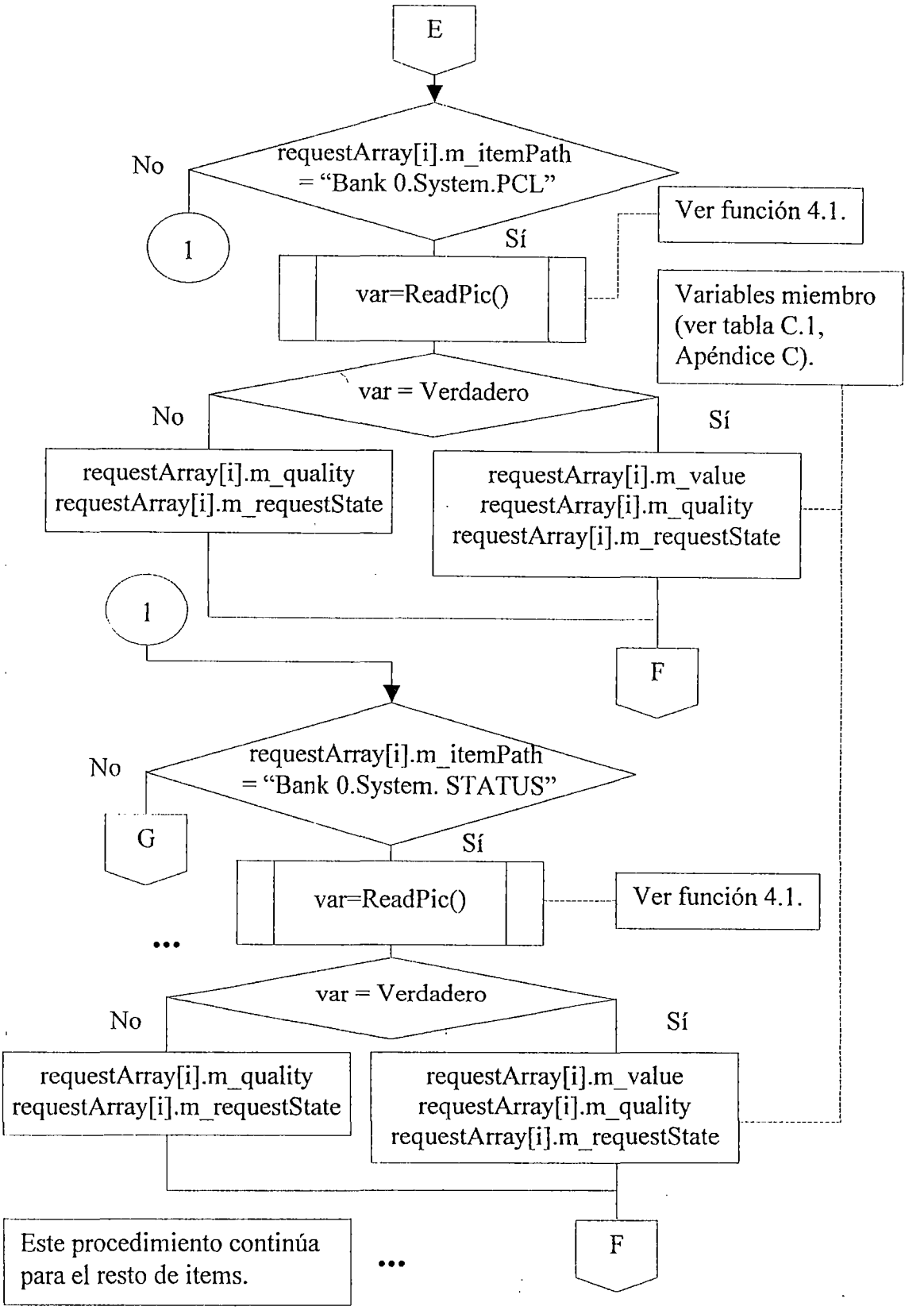


Figura 4.11 (*Continuación de la página 208.*) Algoritmo de la función *SOXpSHandleDARequests* para el pedido de lectura y escritura (*continúa...*)

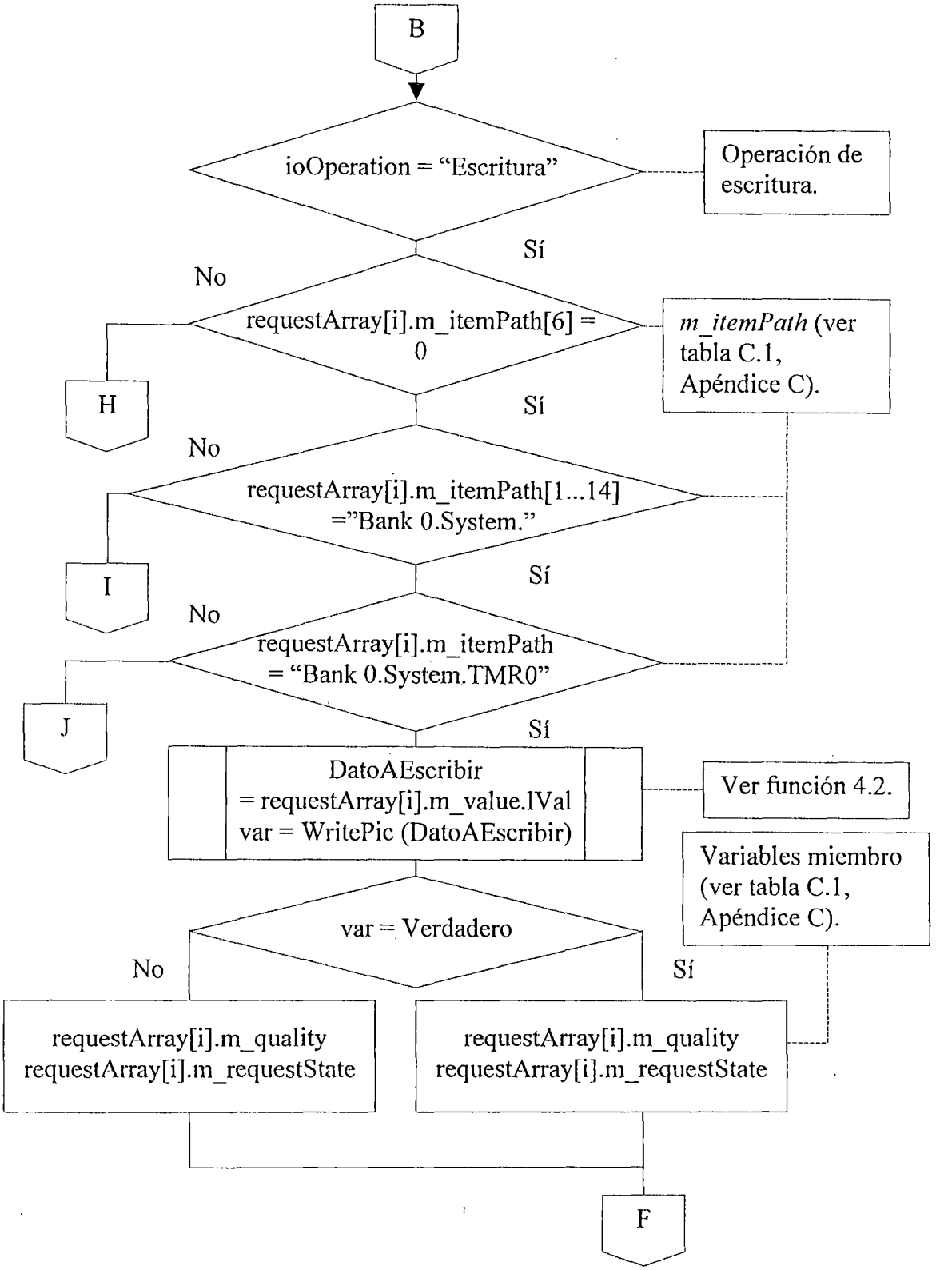


Figura 4.11 (*Continuación de la página 209.*) Algoritmo de la función *SOXpSHandleDARequests* para el pedido de lectura y escritura (*continúa...*)

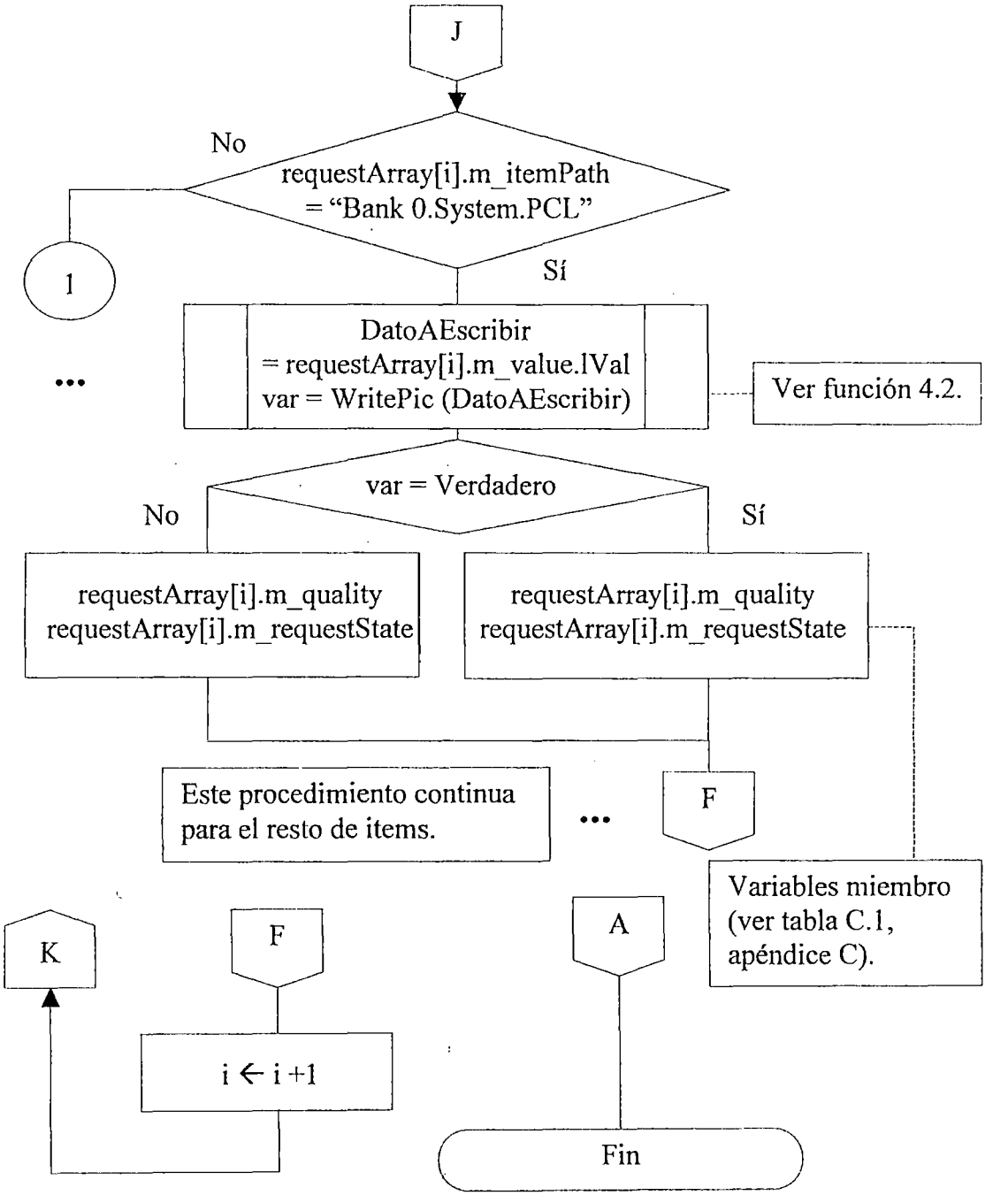


Figura 4.11 (*Continuación de la página 210.*) Algoritmo de la función *SOXpSHandleDARequests* para el pedido de lectura y escritura

```
1.  void SOXPSCFG_CALL SOXpSHandleDARequests(
2.      IN DWORD ioOperation,
3.      IN DWORD numRequests,
4.      IN OUT SOXpSDARequestData *requestArray)
5.  {
6.      DWORD i;
7.      for (i = 0; i < numRequests; i++)      // maneja todos los pedidos
8.      {
9.          int DataRead;
10.         int Bank;
11.         if (ioOperation ==
                    SOXPS_REQUEST_OPERATION_READ)
12.         {
13.             // Banco 0
14.             if (requestArray[i].m_itemPath[6] == _T('0'))
15.             {
16.             Bank=0;
17.                 if (_tcsncmp(requestArray[i].m_itemPath,
                    _T("Bank 0.System."), 15) == 0)
18.                 {
19.                     if
                        (_tcsncmp(requestArray[i].m_itemPath,
                        _T("Bank 0.System.TMR0"), 19) == 0)
20.                     {
21.                         if(::ReadPic(Bank,1,&DataRead))
22.                         {
23.                          requestArray[i].m_value.lVal
                            =(LONG)DataRead;
24.                          requestArray[i].m_quality =
                            SOXPS_QUALITY_GOOD;
25.                 requestArray[i].m_requestState =
                    SOXPS_REQUEST_STATE_HANDLED;
26.                         }
27.                         else
28.                         {
29.                            requestArray[i].m_quality =
                               SOXPS_QUALITY_BAD ;
30.                         requestArray[i].m_requestState =
                SOXPS_REQUEST_STATE_NOT_HANDLED ;
31.                         }
32.                     }
33. ...
```

Lista 4.7 Parte de la implementación de la función *SOXpSHandleDARequests* para el manejo de un pedido de lectura

```
1.  ...
2.      else
3.      { // Función de escritura
4.          int DataToWrite;
5.          if (requestArray[i].m_itemPath[6] == _T('0'))
6.          {
7.          Bank=0;
8.
9.              if (_tcsncmp(requestArray[i].m_itemPath, _T("Bank
                0.System."), 15) == 0)
10.             {
11.                 if (_tcsncmp(requestArray[i].m_itemPath,
                    _T("Bank 0.System.TMR0"), 19) == 0)
12.                 {
13.                 DataToWrite=requestArray[i].m_value.lVal ;
14.                     if(::WritePic(Bank,1,DataToWrite))
15.                     {
16.                         requestArray[i].m_quality =
                            SOXPS_QUALITY_GOOD;
17.
                        requestArray[i].m_requestState =
                    SOXPS_REQUEST_STATE_HANDLED;
18.                     }
19.                     else
20.                     {
21.                         requestArray[i].m_quality =
                            SOXPS_QUALITY_BAD ;
22.
                    requestArray[i].m_requestState =
                SOXPS_REQUEST_STATE_NOT_HANDLED ;
23.                     }
24.                 }
25.                 else if (_tcsncmp(requestArray[i].m_itemPath,
                    _T("Bank 0.System.PCL"), 18) == 0)
26.                 {
27. ...
```

Lista 4.8 Parte de la implementación de la función *SOXpSHandleDARequests* para el manejo de un pedido de escritura

4.4.3.4 Finalización del Servidor

La función *SOXpSStop* (ver función C.2, Apéndice C) es llamada cuando el servidor está apunto de cerrarse. Por lo tanto dentro de esta función se debe llamar a la función *CutConnection* (función 4.4), para liberar la memoria del objeto Serial232 definido inicialmente en la lista 4.5 línea 7.

4.4.3.5 Implementación de Seguridad para el Servidor de OPC

Los parámetros de seguridad se pueden configurar de dos formas:

- Usando el programa DCOMCNFG. Para usar este programa, el usuario debe ser el Administrador del sistema. Todos los cambios realizados se aplican al sistema y también a un componente DCOM completo (InProc o OutProc Server, ver Apéndice A). No es posible definir las configuraciones para partes de los componentes (ejemplo, objetos simples, interface, métodos).

- Usando métodos API para seguridad DCOM. Las configuraciones pueden ser hechas por el cliente o el servidor o por ambos. Las configuraciones de seguridad implementadas por medio de estos métodos no pueden ser sobrescritas usando el programa DCOMCNFG. Este procedimiento permite cambios específicos en la seguridad de los

objetos. La configuración puede ser definida incluso para pequeñas partes de los componentes (llamadas de método, fuente de datos, almacenaje de datos). Esta implementación de seguridad no se tomó en cuenta para el desarrollo del presente proyecto.

4.4.3.5.1 Configuración usando el programa DCOMCNFG

Una forma de configurar los parámetros de seguridad se muestra en la tabla 4.5, con estos parámetros el servidor y el cliente pueden siempre operar sin problemas.

Parámetro	Descripción
Default Properties Authentication Level	None
Default Properties Impersonation Level	Impersonate
Default Securtiy Launch Permissions	Everyone, System, Administrator e Interactive User
Default Securtiy Access Parmissions	Everyone, System, Administrator e Interactive User

Tabla 4.5. Parámetros de seguridad para el Servidor de OPC usando el programa DCOMCNFG

Si la configuración de la tabla 4.5 no satisface los requerimientos requeridos, se pueden cambiar los respectivos valores. Para más información ver el capítulo 10

del libro "The COM and COM+ Programming Primer". El código del programa de este servidor de OPC se encuentra en el CD adjunto a esta tesis.

CAPÍTULO V

APLICACIONES Y RESULTADOS

Las aplicaciones mostradas en ente capítulo permitirán tener una idea general del alcance que tiene el presente proyecto, y también mostrará como introducir el Cliente y el Servidor de OPC desarrollados en los capítulos anteriores en aplicaciones reales en la industria. Una aplicación implementada en la industria nacional es expuesta a detalle, además de sus beneficios obtenidos.

Este capítulo también abarca los resultados obtenidos al realizar las pruebas de rendimiento aplicados al Servidor de OPC para el microcontrolador PIC 16F873 y el software cliente OPC Client. Es importante notar que las pruebas realizadas sirven para saber cuales son los factores que influyen en el rendimiento de la comunicación, es por eso que las conclusiones obtenidas de estas pruebas serán de mucha utilidad.

5.1 Aplicaciones Industriales

Tal como se podrá apreciar en las aplicaciones siguientes existen diferentes soluciones donde se pueden aplicar en la industria. Todas las soluciones al final cumplen una función común, el cual es la integración de sistemas, tal como se planteó en el primer capítulo.

5.1.1 Aplicación en una Planta Productora de Fideos

El esquema mostrado en la figura 5.1 corresponde a una planta productora de fideos, donde se puede apreciar que existen 3 líneas de producción y un sistema de almacenaje y dosificación de materia prima.

5.1.1.1 Planteo del Problema

Se tiene la necesidad de obtener los datos reales de la planta, para el reporte de producción diario, estos datos son:

- Cantidad de materia prima consumida (harina, agua, vitaminas, productos de envasado, etc)
- Cantidad de producto terminado (cantidad de paquetes, cantidad de fardos, etc)

- Cantidad de producto rechazado (cantidad de paquetes con peso no ideal, cantidad de paquetes deformados, etc)

Este reporte tendrá que ser automático y será enviado al software de Gestión SAP R/3.

5.1.1.2 Planteo de la Solución

En vista que se tiene diferentes marcas de PLC y por ende diferentes protocolos de comunicación se opta por una solución basado en un estándar (ver figura 5.2).

La solución basado en OPC es mostrada en las figuras 5.2 y 5.3, el cual consta del software OPC Client, el cual obtiene la información de dos servidores de OPC locales (SIMATIC ProTool/Pro y RSLinx OPC Server). El software OPC Client obtiene la información de todas los datos necesitados los procesa y los envía al software de Gestión SAP R/3.

El servidor de OPC SIMATIC ProTool/Pro provee la información del PLC de la marca Siemens (Recepción de Materia Prima) y el servidor de OPC RSLinx OPC Server provee la información de todos los PLC de la marca Allen Bradley (líneas 1 ,2 y 3, envasadoras y enfardeladoras). (Ver figura 5.3.)

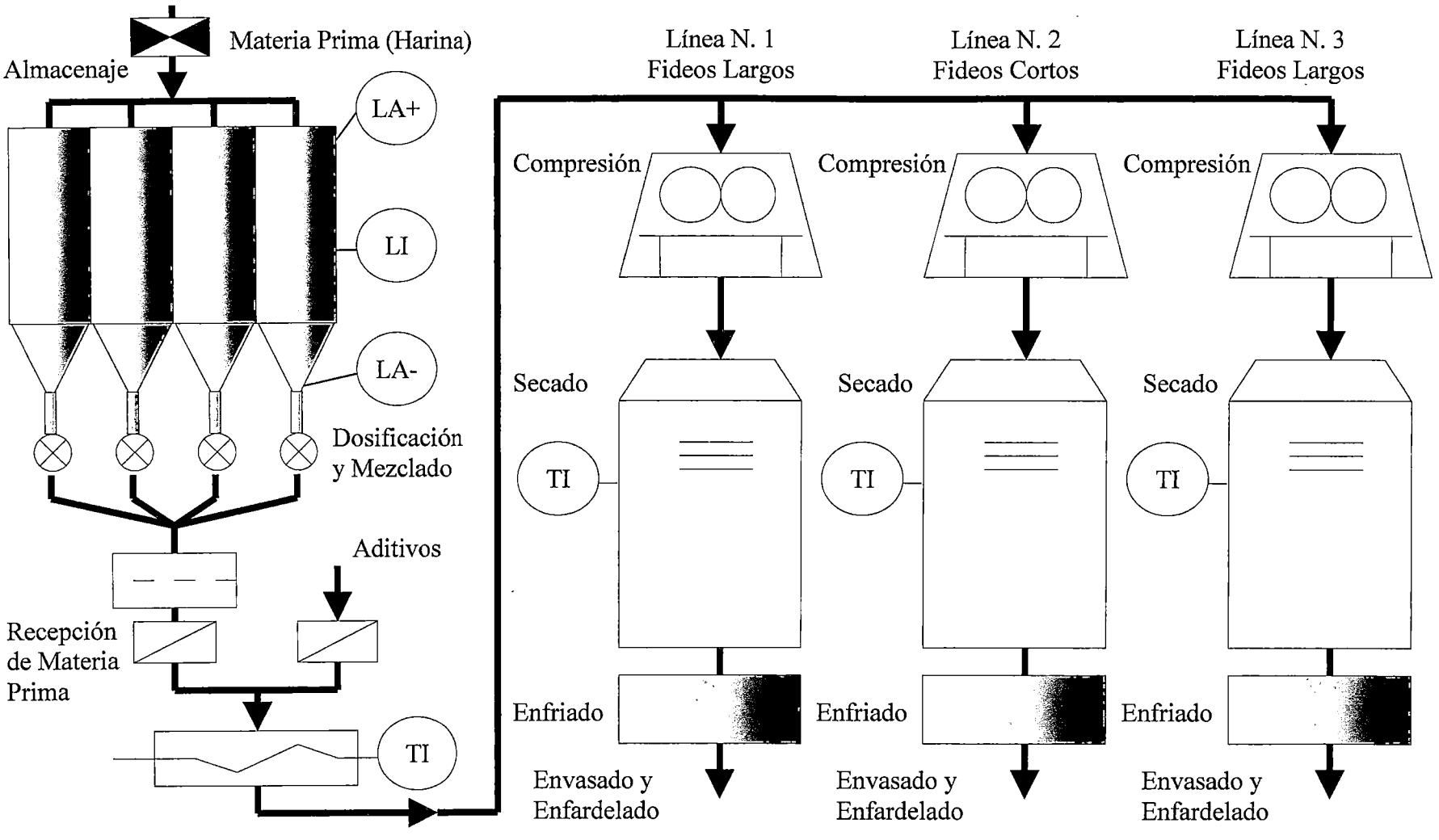


Figura 5.1 Esquema de una planta productora de fideos

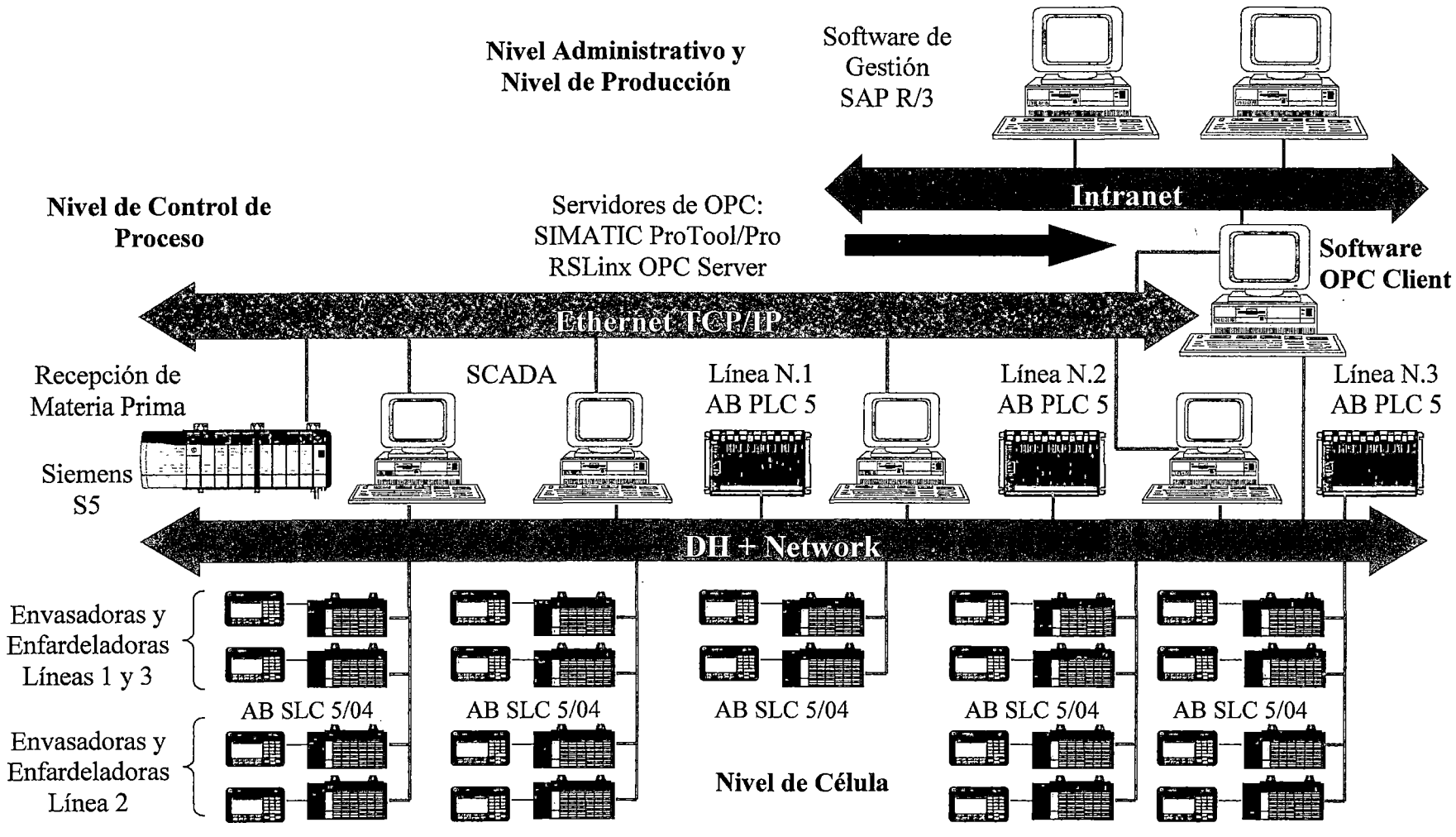


Figura 5.2 Esquema general de la configuración de la red de comunicación de una planta productora de fideos

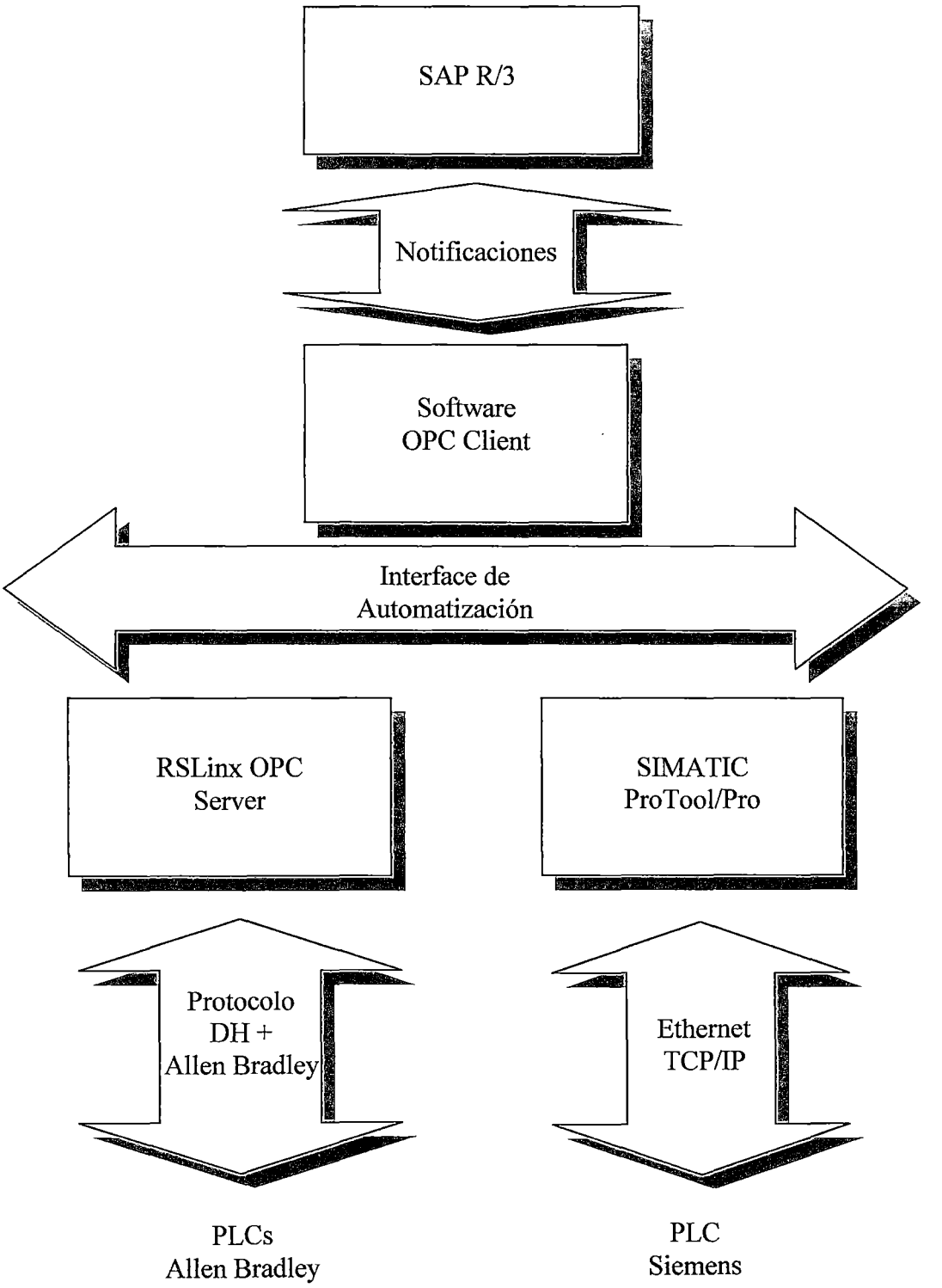


Figura 5.3 Esquema de la solución planteada

5.1.1.3 Ventajas

- Fácil integración de otros sistema que se puedan añadir (consumo de energía, consumo de combustible, etc).
- Mejor supervisión y control de variables que influyen en la producción y por ende se mejora la eficiencia de la planta.
- Gran facilidad para el cambio de nuevas variables requeridas.
- Bajo costo en comparación con sistemas propietarios existentes en la industria.

5.1.2 Supervisión y Visualización de Datos usando del Software OPC Client

Otra aplicación se puede apreciar en la figura 5.4, donde se tiene dos servidores de OPC (SIMATIC ProTool/Pro). Estos servidores proveen la información de tres diferentes máquinas (controlados cada uno con un PLC) al software OPC Client. La configuración fue realizada usando el software OPC Client y los datos son visualizados por medio de la funcionalidad que ofrece el software con Excel. (Para más detalle de la configuración ver el capítulo III).

Los datos obtenidos son los siguientes: Cantidad de producto producido, cantidad de merma, cantidad de materia prima, variables críticas, cantidad de horas de trabajo, variables de temperatura de motores, de niveles de aceite, etc.

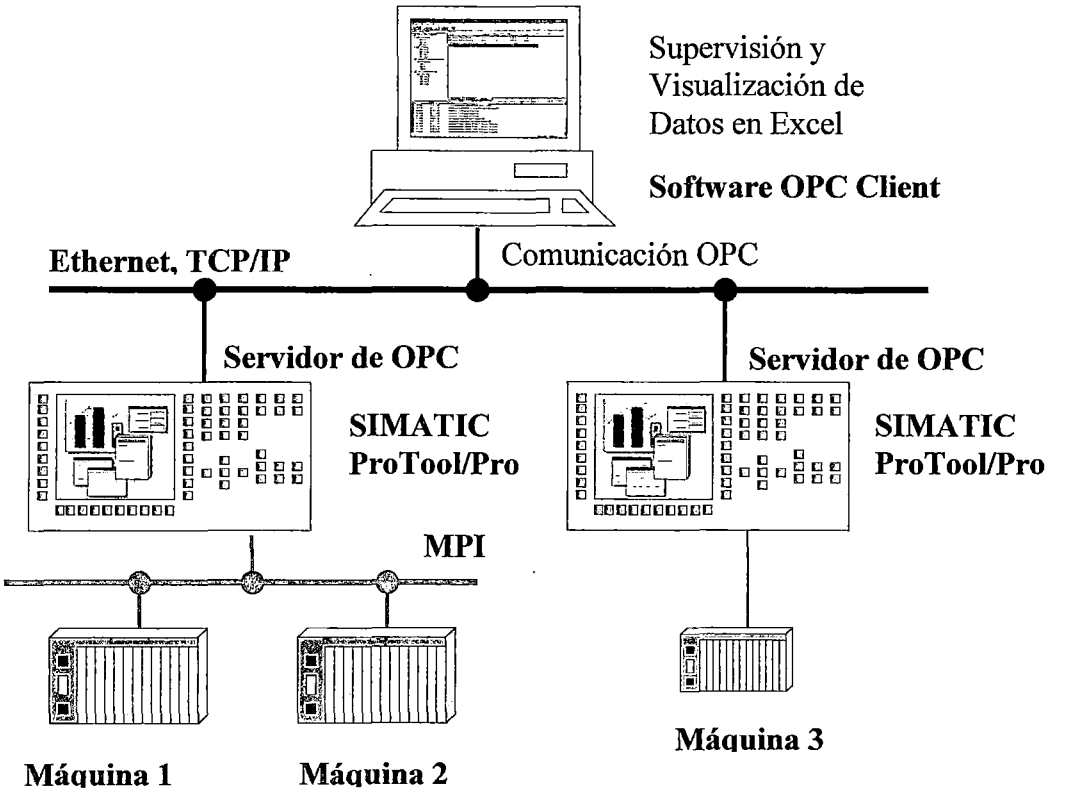


Figura 5.4 Software OPC Client usado para supervisión y visualización de datos de una planta industrial

Para la aplicación mostrada en la figura 5.4 se pueden realizar modificaciones acordes con cada necesidad. Estas modificaciones son llevadas a cabo al nivel de usuario de manera muy fácil utilizando las herramientas que ofrece Excel.

5.1.3 Aplicaciones Distribuidas con Internet

La figura 5.5 muestra una posible estructura que se puede dar en una aplicación distribuida con Internet. En general, los usuarios que trabajan con aplicaciones de este tipo tienen que lidiar con productos tales como *firewall*, Internet y una Red Privada Virtual (VPN). Tal como se muestra en la figura 5.5 los dos *firewall* están instalados entre la intranet y la Internet. Estos están destinados para el monitoreo y, si es requerido, para la restricción del acceso externo a los recursos internos.

Esta configuración permite utilizar el software OPC Client en forma distribuida, ya que el servidor de OPC se puede encontrar en la planta y el cliente en otro lugar muy distante.

Para más información acerca de la configuración del *firewall* y de DCOM se debe consultar con el capítulo 4 del libro "OPC Fundamentals, Implementation, and Application" y el capítulo 10 del libro "The COM and COM+ Programming Primer", respectivamente.

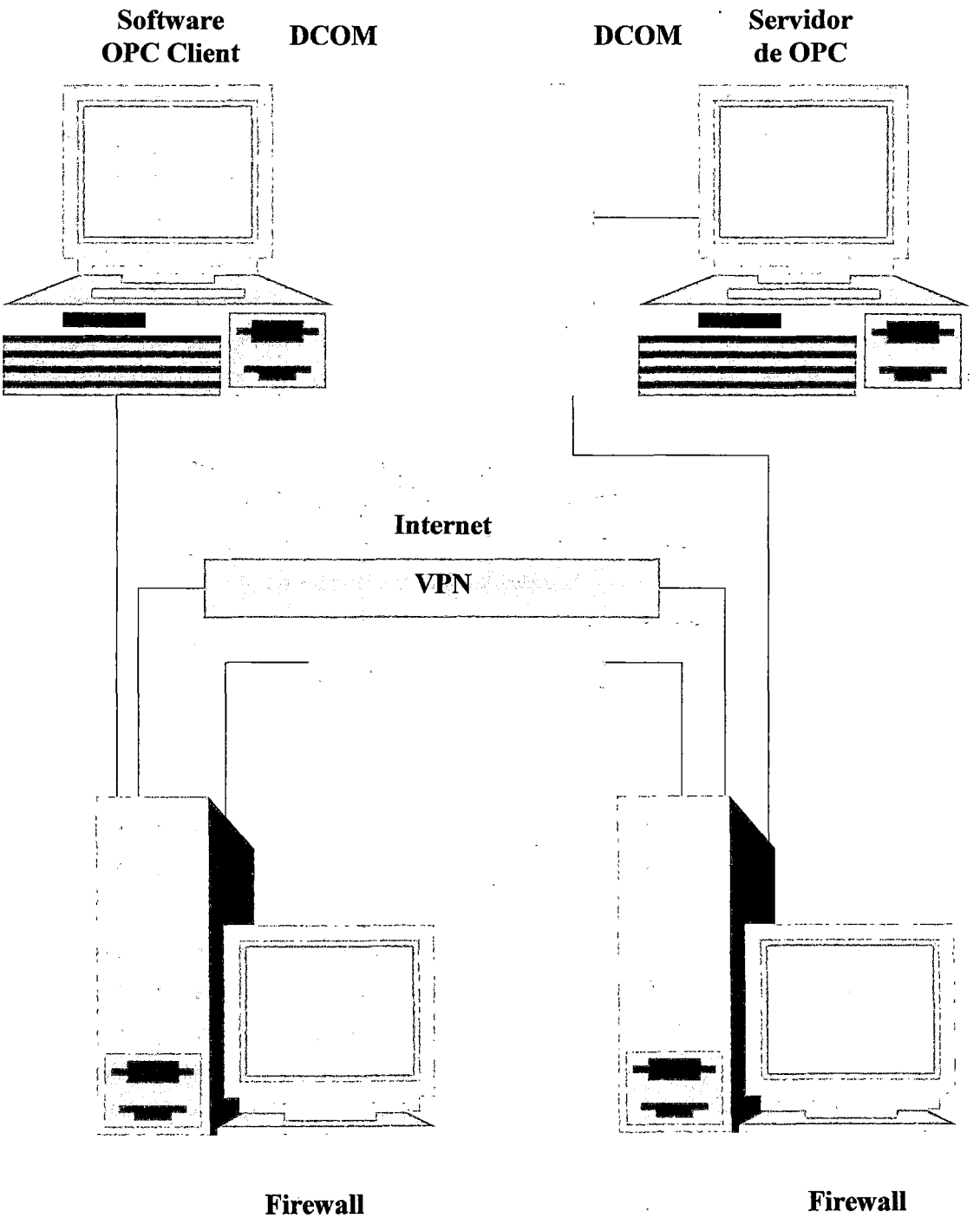


Figura 5.5 Aplicación distribuida en Internet usando el Software OPC Client

5.1.4 Aplicaciones con el Servidor de OPC para el Microcontrolador PIC 16F873

Este servidor puede usarse para integrar sistemas desarrolladas usando este microcontrolador (contadores, temporizadores, control de temperatura, control de nivel, etc) con sistemas industriales de una planta (procesos industriales, máquinas, etc).

5.2 Limitaciones

5.2.1 Limitaciones de la Especificación Acceso de Datos

- El método *Refresh* es disponible por la especificación, el cual permite leer todos los items de un grupo sin tener la necesidad de direccionar a cada ítem. También es conveniente usar un método *Write Refresh*, el cual debe permitir realizar operaciones de escritura de todos los objetos *OPCItem* que pertenecen a un objeto *OPCGroup*, sin tener la necesidad de direccionar a cada objeto *OPCItem*, sino directamente al objeto *OPCGroup* al que pertenecen. Este último método no está disponible por la especificación
- El *namespace* provisto por un servidor de OPC es estático, pero existen algunos casos donde no es conveniente usar este tipo de *namespace*:

- La información en el *namespace* es disponible (por medio de una base de datos) y no tiene que ser ingresado durante la configuración del servidor.
- El *namespace* es muy largo (ejemplo 20000 items a más), y su creación durante la configuración del servidor no es conveniente.
- El *namespace* no es prioridad para iniciar el servidor (aplicaciones dinámicas).

Por lo tanto se sugiere tener un *namespace* dinámico, el cual pueda variar siempre y no solamente al configurar el servidor. Este tipo de *namespace* no está definido por la especificación.

- Para algunas aplicaciones es necesario enviar valores de distintos objetos *OPCItem* hacia los dispositivos periféricos en ciertos intervalos de tiempo (de modo similar al usar el parámetro *UpdateRate* para recibir la notificaciones en ciertos intervalos de tiempo). Este tipo de operación es muy común al realizar un sistema de control controlado por una PC. Este tipo de operación no está definido por la especificación.
- Cuando se usa un tipo de dato estructurado (como un arreglo de tipo de datos string), la información, como el tipo de dato no es suficiente. Por ejemplo, cada arreglo puede tener información de cada elemento, el cual no se puede definir con la especificación actual. Una solución podría ser que la información de cada elemento puede ser definido como propiedad del arreglo.

5.2.2 Limitaciones del Software OPC Client

- No soporta la creación de grupos públicos (ver Apéndice A).
- Sólo es posible trabajar bajo sistemas operativos de Windows (ver especificaciones técnicas).
- No provee la funcionalidad de la interface para la configuración y utilización de políticas de seguridad (Especificación "OPC Security Specification"). Esta especificación no está disponible para la Interface de Automatización (*Automation Interface*).
- No provee la funcionalidad de la interface para el monitoreo de eventos (Especificación "OPC Alarms and Events Specification"). Esta especificación no está disponible para la Interface de Automatización (*Automation Interface*).
- No provee la funcionalidad de la interface para el acceso de datos históricos (Especificación "OPC Historical Data Access Specification").
- No provee la funcionalidad de la interface para el acceso de datos para un proceso por lotes o tipo *Batch* (Especificación "OPC Batch Specification").

Las limitaciones mencionadas no fueron implementadas en el Software OPC Client. Las funcionalidades no implementadas podrían ser nuevos temas de investigación para futuros trabajos.

5.2.3 Limitaciones del Servidor de OPC para el Microcontrolador PIC 16F873

- No soporta grupos públicos (ver Apéndice A).
- Sólo es posible trabajar bajo sistemas operativos de Windows (ver especificaciones técnicas).
- No provee la funcionalidad de la interface para la configuración y utilización de políticas de seguridad (Especificación "OPC Security Specification").
- No provee la funcionalidad de la interface para el monitoreo de eventos (Especificación "OPC Alarms and Events Specification").
- No provee la funcionalidad de la interface para el acceso de datos históricos (Especificación "OPC Historical Data Access Specification").
- No provee la funcionalidad de la interface para el acceso de datos para un proceso por lotes o tipo *Batch* (Especificación "OPC Batch Specification").

Las limitaciones mencionadas no fueron implementadas en el Servidor de OPC para el microcontrolador PIC. Las funcionalidades no implementadas podrían ser nuevos temas de investigación para futuros trabajos.

5.3 Especificaciones Técnicas

5.3.1 Especificaciones Técnicas del Servidor de OPC para el Microcontrolador 16F873

- Este servidor soporta la Especificación "OPC Data Access Specification" versión 2.05.
- Este servidor soporta los siguientes sistemas operativos: Microsoft Windows 95, 98, XP, CE, NT y 2000.

5.3.2 Especificaciones Técnicas para el Software OPC Client

- Este software soporta la Especificación "OPC Data Access Automation Interface Standard" versión 2.02.
- Este software soporta los siguientes sistemas operativos: Microsoft Windows 95, 98, NT 4.0 ó más y 2000.

5.4 Pruebas de Rendimiento

En esta sección se muestran las pruebas realizadas para el servidor de OPC del microcontrolador PIC 16F873 y para el software OPC Client, desarrollados en los capítulos anteriores.

5.4.1 Objetivos de las Pruebas

El objetivo de las medidas realizadas es encontrar la velocidad mínima que se puede alcanzar y los efectos que se obtendrán cuando las variables que influyen en la comunicación varían.

5.4.2 Consideraciones para las Pruebas Realizadas

Las pruebas fueron llevadas a cabo utilizando las siguientes computadoras:

- 2 Pentium II de 200 MHz
- 1 Pentium II de 400 MHz
- 1 Pentium II de 550 MHz

El sistema operativo utilizado fue Windows NT, además que el cliente y servidor fueron las únicas aplicaciones activas en las computadoras.

Las medidas fueron tomadas considerando lo siguiente:

- El cliente y el servidor están ubicadas en la misma computadora (200 MHz, 400 MHz y 550 MHz).
- El cliente y el servidor están ubicadas en dos diferentes computadoras (200MHz). En este caso, las computadoras son conectadas vía una red Ethernet de 10Mbit/s. Ninguna otra computadora fue conectada a la red a excepción de las computadoras de prueba. Esto asegura que ningún tráfico adicional ocurrió en la red.

Los siguientes factores fueron variados para las medidas tomadas:

- El número de objetos *OPCItem*. Este número fue variado desde 0 a 20,000 con incrementos de 400. Todos los objetos *OPCItem* fueron del mismo tipo de datos (variable tipo *Integer*). Los valores fueron variados desde el microcontrolador.
- El número de objetos *OPCGroup*. Los objetos *OPCItem* fueron divididos entre 1, 10, 50, y 100 objetos. El parámetro *PercentDeadband* fue puesto a 0, cuando se crearon los objetos *OPCGroup*; esto significa que un valor será transferido en cualquier momento que se produzca un cambio.
- El número de objetos *OPCItem* que cambian. En una aplicación real, los valores de todos los objetos *OPCItem* no cambian a la vez, por lo tanto las medidas fueron tomadas para 0%, 5 %, 10%, 50%, y 100% de los valores que cambian simultáneamente.

Las medidas fueron llevadas a cabo de la siguiente manera:

- Inicialmente, el servidor lee los valores de todos los objetos *OPCItem* que han sido creados para esta medida.
- Después el servidor verifica los valores que tienen que ser pasados hacia el cliente. Es importante considerar que los datos que el servidor va a leer se encuentran en el microcontrolador, por lo tanto se tiene que considerar el tiempo que se demora una transacción de lectura y la cantidad de ítem creados.
- En el siguiente paso, los valores que van a ser transferidos son pasados al cliente llamando el evento *DataChange*. El cliente no procesa estos valores, sólo los visualiza, por lo tanto la llamada regresa inmediatamente al servidor. Después de la lectura de todos los items del microcontrolador hasta el término de la llamada del evento *DataChange*, se considera el tiempo transcurrido para el análisis.

Esta secuencia fue repetida para todas las variantes.

5.4.3 Resultados de las Pruebas

La figura 5.6 muestra que el cambio en el promedio de la velocidad depende tanto del número de objetos *OPCItem* como de los objetos *OPCGroup*. En esta prueba tanto el cliente como el servidor están en la misma computadora (200 MHz).

La influencia que el número de objetos *OPCGroup* tiene sobre la velocidad mínima alcanzada es visible pero no significante. Para un número de 20 000 items (a 5% significa que 1 000 valores son transmitidos), una velocidad mínima de 528 ms es alcanzado para un objeto *OPCGroup* y una velocidad mínima de 632 ms para 100 objetos *OPCGroup* (ver figura 5.6). Como se puede apreciar también en la figura 5.6, cuanto más cercana es y la diferencia entre el número de grupos, las condiciones son casi idénticas tal como se puede apreciar para las gráficas de un grupo y de 10 grupos.

Con un número pequeño de objetos *OPCItem*, la diferencia entre las velocidades mínimas alcanzadas para cada número de objetos es menor.

La figura 5.7 muestra la velocidad mínima cuando el cliente y el servidor están localizados en diferentes computadoras (200 MHz). Las otras condiciones son similares que las del caso anterior. La velocidad alcanzada ha incrementado para este caso, comparado con el caso del servidor y el cliente ubicados en la misma computadora. Los valores para 1 y 50 objetos *OPCGroup* se diferencian claramente.

Al observar figura 5.8, es obvio que usando el cliente y el servidor en una misma computadora resulta una velocidad mínima menor que cuando se encuentran en diferentes computadoras. El valor para la velocidad mínima para 20 000 objetos *OPCItem* y un 5% de cambio para 1 objeto *OPCGroup* es 528 ms en el primer caso y 584 ms en el segundo caso y, para 50 objetos *OPCGroup*, 577 ms en el primer caso y 646 ms en el segundo caso (ver figura 5.8).

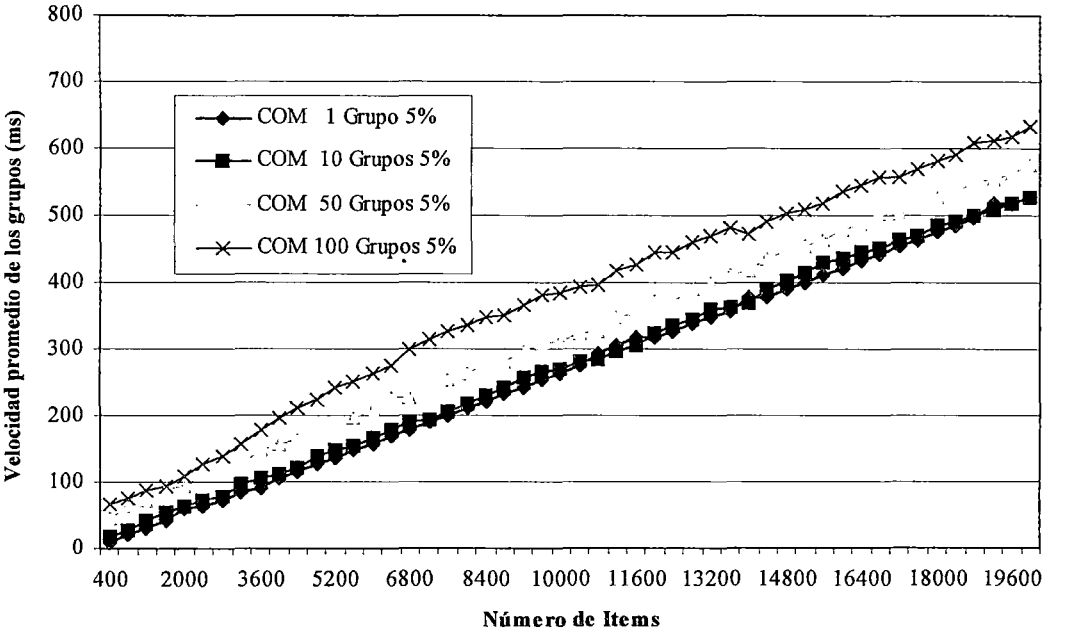


Figura 5.6 Velocidad mínima para diferentes números de objetos *OPCGroup* (el cliente y el servidor están en la misma computadora)

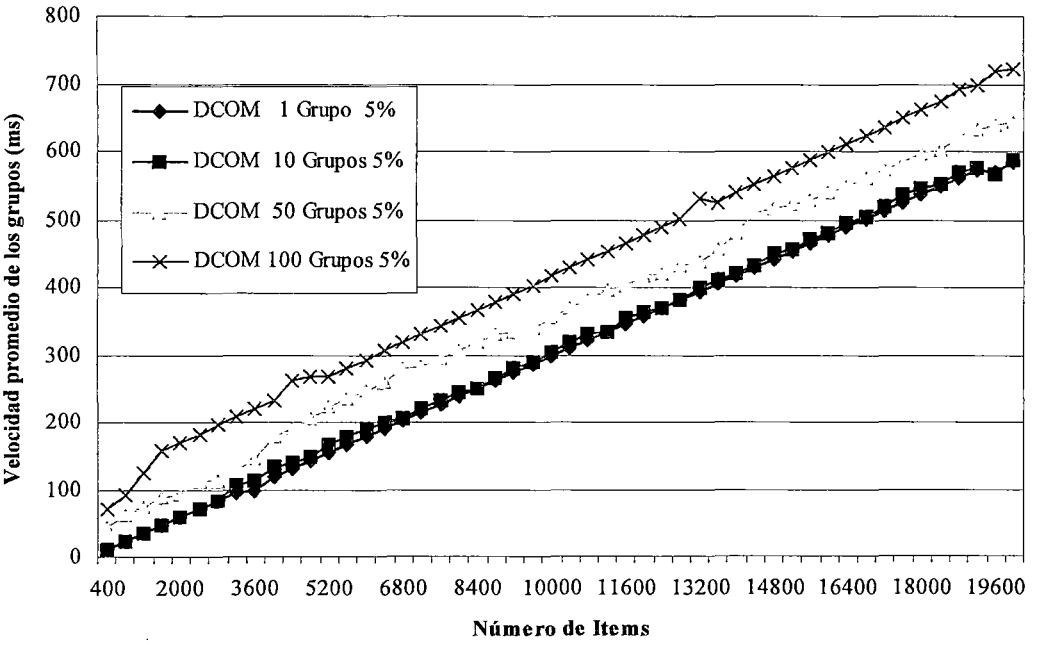


Figura 5.7 Velocidad mínima para diferentes números de objetos *OPCGroup* (el cliente y el servidor están en diferentes computadoras)

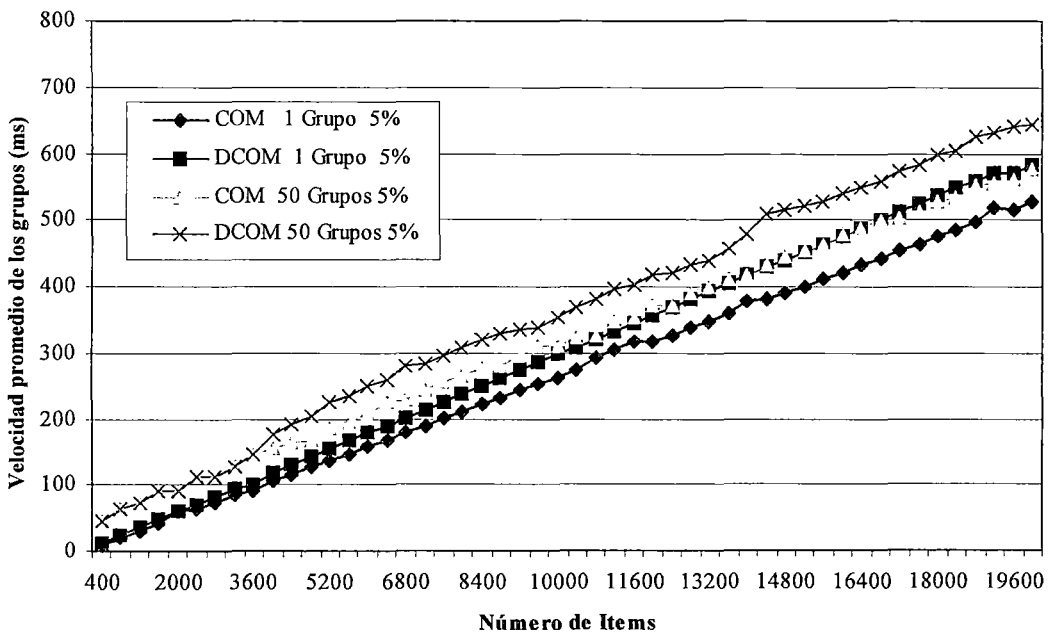


Figura 5.8 Velocidad mínima para diferentes números de objetos *OPCGroup* en diferentes configuraciones (local y remota)

Las figuras 5.9 y 5.10 reflejan las velocidades mínimas encontradas cuando se varía el porcentaje de objetos *OPCItem* que cambian de valor, pero se mantiene constante el número de objetos *OPCGroup*.

La figura 5.9 muestra una configuración del cliente y el servidor en una misma computadora, a diferencia de la figura 5.10 que muestra una configuración del cliente y el servidor en diferentes computadoras. La figura 5.11 muestra una comparación entre las medidas tomadas para las dos tipos de configuraciones (local y remoto).

La tabla 5.1 muestra todos los bytes que son transmitidos hacia el cliente, el primer grupo corresponde a los bytes transmitidos por cada objeto *OPCGroup* y el segundo grupo corresponde a los bytes transmitidos por cada objeto *OPCItem* (para más información del tema, ver la especificación "Data Access Custom Interface Standard" versión 2.05).

Con los datos de la tabla 5.1 y las figuras 5.9 y 5.10 se puede encontrar una velocidad de transmisión con la que trabaja el Servidor para el microcontrolador PIC para los dos tipos de configuraciones (local y remoto).

Bytes por Transacción de un objeto *OPCGroup*	
TransactionIdentifier	4 bytes
GroupHandle	4 bytes
MasterQuality	4 bytes
MasterError	4 bytes
Numero de objetos *OPCItem* en la llamada por objeto *OPCItem*	4 bytes
Subtotal	**20 bytes**
Bytes por cada objeto *OPCItem*	
ClientHandle	4 bytes
Value	6 bytes (4 bytes para el valor y 2 bytes para la información del tipo)
Quality	2 bytes
TimeStamp	8 bytes
Error	4 bytes
Subtotal	**24 bytes**
Total	**480 200 bytes para 20 000 objetos *OPCItem***

Tabla 6.1. Cantidad de bytes transmitidos en la transacción del Servidor hacia el Cliente

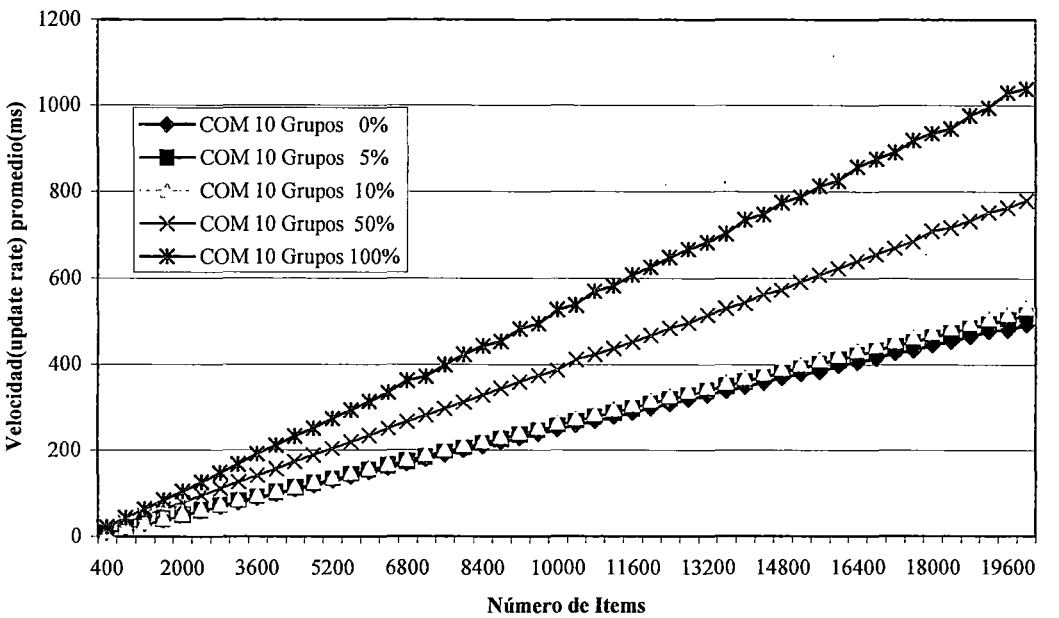


Figura 5.9 Velocidad mínima para diferentes porcentajes de objetos *OPCItem* que cambian de valor (el cliente y el servidor en la misma computadora)

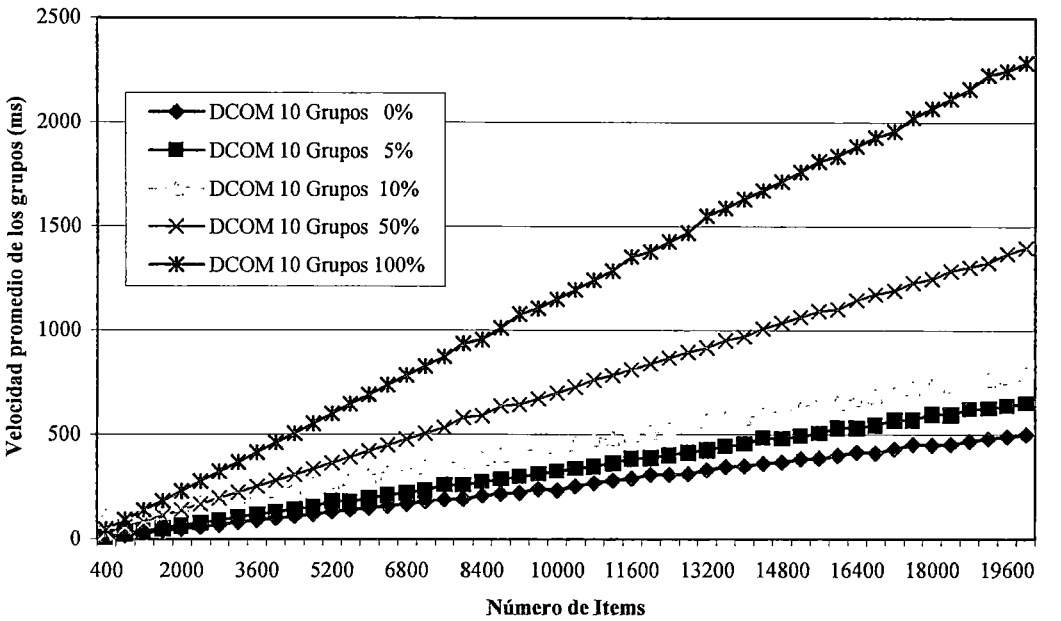


Figura 5.10 Velocidad mínima para diferentes porcentajes de objetos *OPCItem* que cambian de valor (el cliente y el servidor diferentes computadoras)

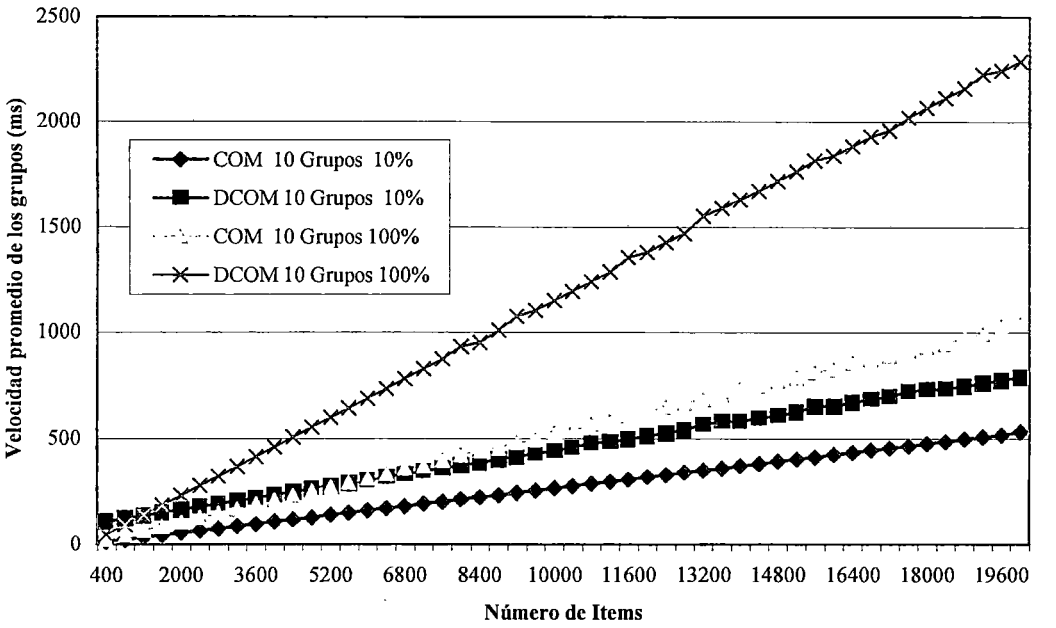


Figura 5.11 Velocidad mínima para diferentes porcentajes de objetos *OPCItem* que cambian de valor en diferentes configuraciones (local y remoto).

Cálculo de la velocidad cuando el cliente y el servidor están en la misma computadora, considerando 20 000 objetos *OPCItem*, 10 objetos *OPCGroup* y 100% de variación en el número de objetos *OPCItem* que varían de valor.

Tiempo promedio = 1 040ms (ver figura 5.9)

Tiempo promedio de lectura de los datos del microcontrolador al Servidor de OPC = 5.21 ms x 20 000 = 104 200 ms (ver tabla 4.1)

Número de bits transmitidos = 480 200 bytes x 8 bits / bytes = 3 841 600 bits

Velocidad de transmisión promedio = 3 841 600bits / 105 240ms

= 36 503 bits/s

Cálculo de la velocidad cuando el cliente y el servidor están en diferentes computadoras, considerando 20 000 objetos *OPCItem*, 10 objetos *OPCGroup* y 100% de variación de los valores de los objetos *OPCItem*.

Velocidad promedia = 2 288 ms (ver figura 5.10)

Tiempo promedio de lectura de los datos del microcontrolador al Servidor de OPC = 5.21 ms x 20 000 = 104 200 ms (ver tabla 4.1)

Número de bits transmitidos = 480 200 bytes x 8 bits / bytes = 3 841 600 bits

Velocidad de transmisión promedio = 3 841 600 bits / 106 488 ms

= 36 075 bits/s.

Para la figura 5.12 muestra la velocidad mínima para diferentes números de objetos *OPCItem* con diferentes velocidades de procesadores. Los objetos *OPCItem* también están divididos en 10 objetos *OPCGroup*.

Es importante observar que al trabajar con un procesador de 200 MHz se obtiene para 20 000 objetos *OPCItem* una velocidad de 580ms y al trabajar con el de 400 MHz se obtiene casi la mitad 295ms (ver figura 5.12). Por tal razón la velocidad del procesador influirá enormemente el rendimiento de la comunicación.

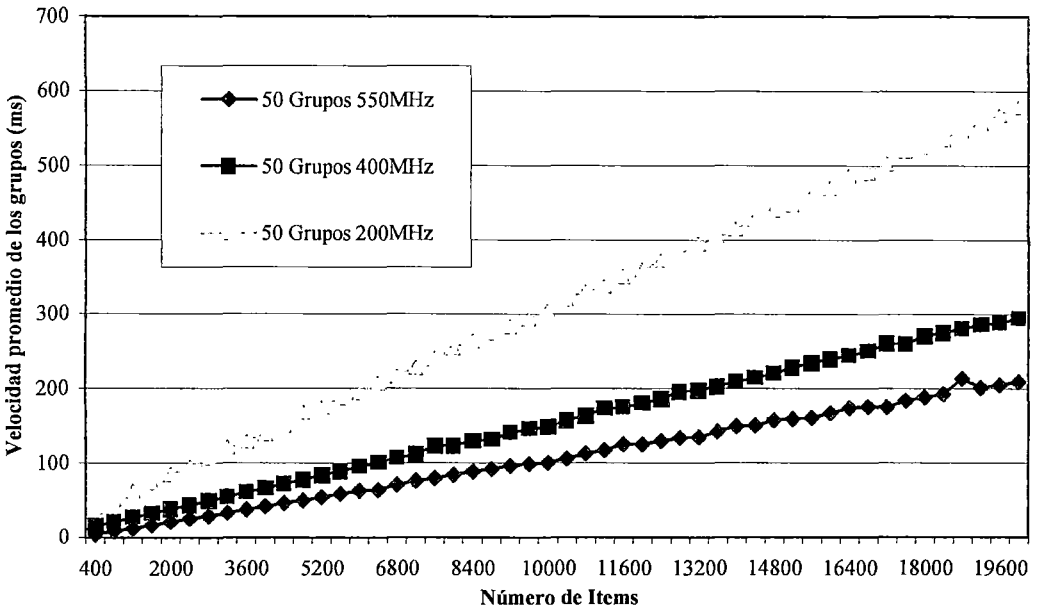


Figura 5.12 Velocidad mínima alcanzada para diferentes velocidades de procesadores

CONCLUSIONES Y RECOMENDACIONES

1. Al utilizar un estándar para el desarrollo tanto del Servidor para el microcontrolador y del Cliente de OPC, se garantiza que las aplicaciones obtendrán las características de interoperabilidad, rendimiento, acceso remoto e interacción. Por lo tanto el producto final tendrá un valor económico.

2. Como la aplicación OPC Client tiene una interfaz muy sencilla de usar y por sobre todo es genérico, permitirá aplicarlo en la industria de una manera muy fácil y rápida.

3. Las dos aplicaciones, tanto el cliente como el servidor de OPC para el microcontrolador no depende del hardware ni software con el que van ha interactuar, sino tan sólo requiere que estos estén basados en el estándar OPC. Esto es una gran ventaja para la integración de sistemas, ya que la gran

mayoría de productos para la industria de la automatización están basados en OPC.

4. Sobre la base de los objetos desarrollados en el presente proyecto se pueden desarrollar aplicaciones de acorde a cada requerimiento específico. Para lograrlo sólo es necesario añadirlo, pero la estructura original será la misma. Una muestra de ello es el desarrollo de la funcionalidad con Excel del software OPC Client, el cual usa los mismos objetos desarrollados en este proyecto con la diferencia que la Interfaz de Usuario es diferente.

5. La funcionalidad que tiene el software OPC Client con Excel permite obtener todas las características que ofrece Excel orientado al control de procesos, por lo tanto esta herramienta será de gran utilidad al realizar reportes, monitoreo, adquisión de datos y hasta control.

6. El desarrollo del Servidor para el microcontrolador permitirá tener una idea de como implementar servidores de OPC, ya que la estructura será la misma porque se usa un estándar. Por ello, con este conocimiento se puede implementar distintos dispositivos inteligentes con la funcionalidad de OPC.

7. Tal como se pudo apreciar en las gráficas de las pruebas de rendimiento, la eficiencia en la comunicación depende de la cantidad de objetos *OPCItem* creados en el servidor y del número de objetos *OPCGroup*, en menor escala. También es importante tener presente que el rendimiento depende de la

velocidad del procesador con el que se está trabajando. En los resultados de las pruebas de rendimiento también se encontró que al duplicar la velocidad del procesador se obtuve una reducción de casi la mitad de la velocidad promedio de comunicación.

8. Otro punto para tener presente al diseñar cuantos objetos *OPCItem* y *OPCGroup* debe tener la aplicación para que ésta pueda trabajar de acorde con los requerimientos de comunicación, es considerar cuantos objetos *OPCItem* cambian a la vez de valor. El análisis realizado fue con 0%, 5% 10%, 50% y 100%, este valor es muy importante ya que influye en el rendimiento de la comunicación.

9. El efecto que se obtiene al distribuir las aplicaciones es evidente, tal como se puede apreciar en las pruebas de rendimiento, pero esta desventaja en el rendimiento de la comunicación es contrapesada con la cantidad de aplicaciones que se puede llevar a cabo al distribuir dichas aplicaciones. También se debe tomar en cuenta el efecto producido por la cantidad de objetos *OPCItem* y *OPCGroup*, ya que esto influye en el rendimiento de la comunicación y por ende es necesario considerarlo para tomar decisiones de diseño.

10. Las conclusiones obtenidas de las pruebas de rendimiento, no sólo son aplicables para el servidor y el cliente desarrollados sino para todos los productos basados en el Acceso de datos de OPC, pero es importante

considerar que la comunicación con cada dispositivo es diferente para cada caso, por lo tanto se debe considerar ese tiempo diferente para el análisis.

BIBLIOGRAFÍA

LIBROS

1. Frank Iwanitz y Jürgen Lange "OPC Fundamentals, Implementation, and Application" 2da Edición, Hüthig GmbH & Co. KG Heidelberg , 2002
2. Alan Gordon "The COM and COM+ Programming Primer", Prentice Hall PTR, 2001.
3. Anthony Abbott, Ralf Jäger, Silvio Apppoloni, Darko Lalics, Manfred Bader, L.M. Lelieveld, Patrick Berdillon, Christian Lerdung, Peter Berry, Patrick Meyer, Maurice Birger, Günther Oehler, Olav Braster, Guglielmo Rossi, Heldga Bühnert, John Slusbury, Olav Cantadore, Dieter H. Schmidt, Georg A. Endress, Hermann Straub, Tony Grassby y Geoff Wickens "Food and Beverages Measurement and Automation", Endress+Hauser Consult Kägenstrasse 7 CH-41533 Reinach.

4. Raimund K. Ege, "Programming in an Object-Oriented Environment", Academic Press (AP), 1992.
5. Josep Balcells, José Luis Romeral, "Autómatas Programables", ALFAOMEGA GRUPO EDITOR, S. A. de C. V., 1998, PARTE III.
6. Javier Ceballos "Visual C++ Programación Avanzada en Win32". RA-MA, 1999
7. Luis Joyanes "Programación en C++, Algoritmos, estructuras de datos y objetos ", McGRAW-HILL/Interamericana de España, S. A. U. 2000.
8. Luis Joyanes "Fundamentos de Programación", McGRAW-HILL/Interamericana de España, S. A. U. 1996.
9. José M. Angulo e Ignacio Angulo "Microcontroladores PIC. Diseño Práctico de Aplicaciones". McGRAW-HILL/Interamericana de España, S. A. U. 1997.

ESPECIFICACIONES

1. OPC Foundation "OPC Overview. Version 1.0",27 de Octubre de 1998.
2. OPC Foundation "OPC Common Definitions and Interfaces Version 1.0", 27 de Octubre de 1998.
3. OPC Foundation "OPC Data Access Custom Interface Standard Version 2.05", Diciembre del 2001.
4. OPC Foundation "OPC Data Access Automation Interface Standard Version 2.02", Febrero de 1999.

INTERNET

1. Fundación de OPC: http://www.opcfoundation.org.
2. Softing AG: http://www.softing.com
3. Microchip: http://www.microchip.com

APÉNDICE A

FUNDAMENTOS DE OPC

1 Requisitos de Instalación

Los servidores y clientes de OPC comparten algunos componentes en común tales como:

- Archivos Proxy/Stub, los cuales proveen la comunicación de los punteros de las interfaces y parámetros de los métodos.
- Componente OPCServerBrowser. Habilita a los Clientes de OPC para que ellos puedan informarse de Servidores de OPC que se encuentran en computadoras remotas. La instalación de este componente según la especificación de OPC está establecida por el servidor de OPC.

- Automation Wrapper. Además de la funcionalidad del acceso a la funcionalidad del servidor de OPC usando la Interface de Cliente (*Custom Interface*), este acceso también es posible vía la Interface de Automatización (*Automation Interface*), el cual usa la librería *Automation Wrapper*.

2 Inscripción en el Registro del Sistema

Cada servidor de OPC tiene que hacer por lo menos tres entradas en el registro del sistema, tal como se puede apreciar en la figura A.1. Los identificadores ingresados son los siguientes:

1- El ProgramIdentifier (*ProgID*) contiene una cadena de sólo lectura que describe el componente. La estructura de la cadena es predefinida: <Fabricante>.<Nombre del Servidor>.

   Por ejemplo: Softing.OPCToolboxDemo_ServerDA en la figura A.1.

2- El ClassID (*CLSID*) es un valor de 128 bits el cual define en forma única a un servidor.

   Por ejemplo: {2E565242-B238-11B3-842D-0008C779D775} para el servidor OPCToolboxDemo_ServerDA en la figura A.1

3- El ApplicationID (*AppID*) contiene otra descripción de sólo lectura del servidor. La estructura y contenido del identificador *AppID* no están predefinido.

Por ejemplo: {2E565242-B238-11B3-842D-0008C779D775} para el servidor OPCToolboxDemo_ServerDA en la figura A.1

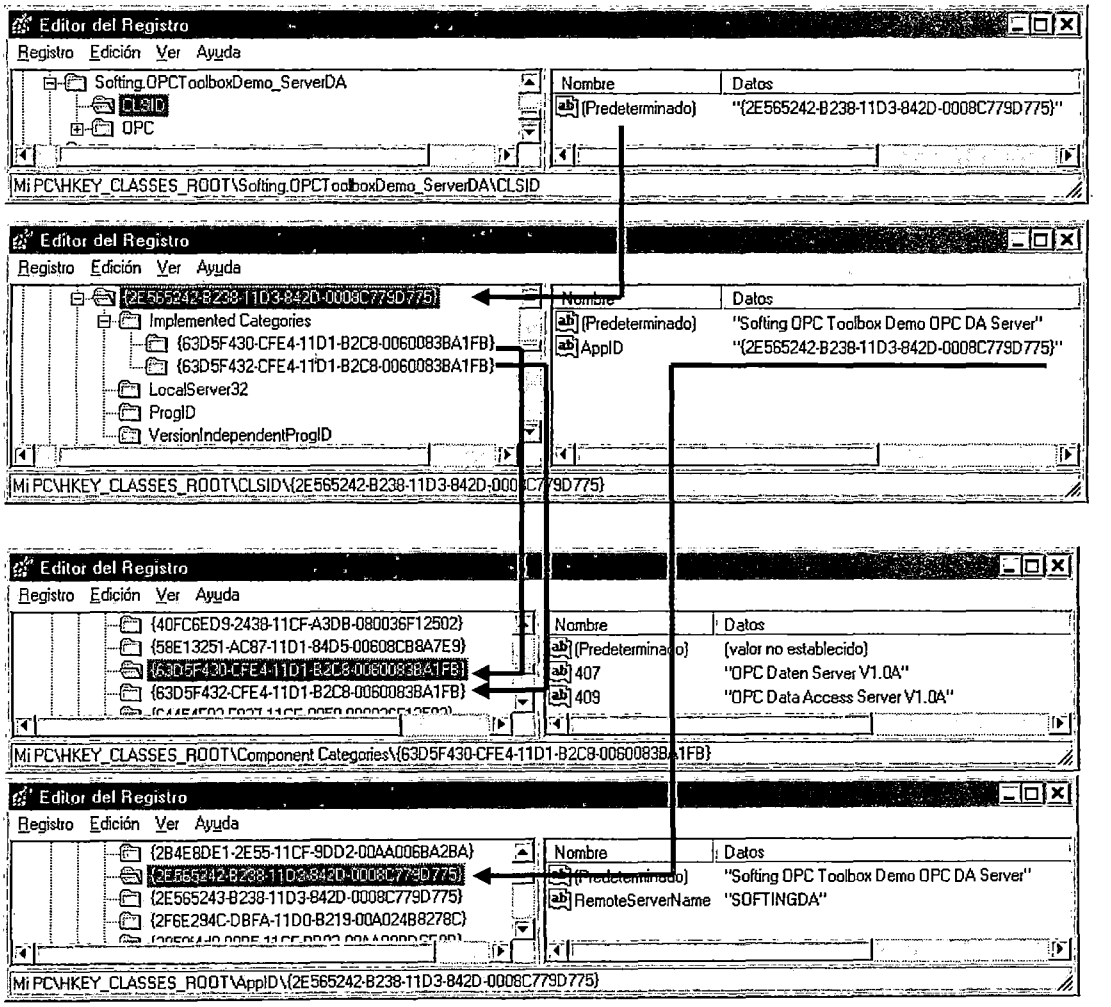


Figura A.1 Entradas al registro del sistema especificadas por OPC

Como hay muchas especificaciones y por eso muchas categorías de productos de OPC disponibles. Por esto las distinciones han sido hechas entre las especificaciones por medio del identificador *CategoryID* (*CATID*). Este identificador *CATID* es situado de la siguiente manera *CLSID.CATID*. La tabla A.1 muestra los valores actuales. Por ejemplo: el servidor OPCToolboxDemo_ServerDA usa la especificación "OPC Data Access Server V1.0A" el cual tiene como identificador *CATID* el número {63D5F430-CFE4-11D1-B2C8-0060083BA1FB} en figura A.1

Especificación	***CATID***
Especificación *Data Access 1.0A*	{63D5F430-CFE4-11D1-B2C8-0060083BA1FB}
Especificación *Data Access 2.0*	{63D5F432-CFE4-11D1-B2C8-0060083BA1FB}
Especificación *Alarms and Events 1.02*	{58E13251-AC87-11D1-84D5-00608CB8A7E9}
Especificación *OPC Batch 1.0*	{A8080DA0-E23E-11D2-AFA7-00C04F539421}
Especificación *Historical Data Access 1.0*	{7DE5B060-E089-11D2-A5E6-000086339399}
Especificación *OPC Batch 2.0*	{843DE67B-B0C9-11d4-A0B7-000102A980B1}

Tabla A.1 Especificaciones de OPC y sus respectivos CATID

3 Componente *OPCServerBrowser*

La información necesaria que necesita un cliente para iniciar a un servidor está disponible en registro del sistema, pero se tiene algunos problemas cuando se usa una máquina remota por las restricciones de acceso. Estos problemas pueden ser resueltos con el componente *OPCServerBrowser* habilitado por la Fundación OPC.

El componente *OPCServerBrowser* ofrece los siguientes métodos en la interface *IOPServerList*:

- **EnumClassesOfCategory:** El cliente transfiere uno o varios identificadores *CATID* y retorna uno o más listas de identificadores *CLSID* de los servidores que están implementados en la especificación correspondiente.
- **GetClassDetails:** El cliente transfiere el identificador *CLSID* y obtiene el identificador *ProgID* y la descripción del servidor. Esta descripción puede ser encontrado por el componente *OPCServerBrowser* por medio del identificador *AppID*.
- **CLSIDFromProgID:** El cliente transfiere el *ProgID* y obtiene el *CLSID*.

El cliente de OPC usará la funcionalidad del componente *OPCServerBrowser* en la computadora local y remota.

4 Pedido de Cierre del Servidor al Cliente de OPC (*shutdown*)

Cuando un servidor de OPC ha sido cerrado mientras está siendo usado por un Cliente de OPC, el servidor informa al cliente de este evento. El servidor llama a un método en la correspondiente interface de comunicación de vuelta (*callback*) del cliente e informa al cliente del cierre (*shutdown*). Como un parámetro del método, el servidor puede pasar una cadena de texto el cual no está definido por OPC. El comportamiento del cliente y el servidor después de la llamada del método tampoco no está definido.

5 *Namespace*

Este contiene todos las fuentes de datos habilitados por un servidor. El *namespace* puede ser de dos tipos:

- *Namespace* plano (*flat*), donde todos los elementos están en un nivel simple. No hay nodos.
- *Namespace* jerárquico (*hierarchical*), donde la estructura es definida como un árbol, en el cual todos los nodos (ramas) pueden ser usados como una estructura; Ellos por ejemplo pueden representar dispositivos en una instalación. Las hojas son disponibles en los nodos, representando

fuentes de datos y generadores de datos. Las hojas son también llamadas *Items* (ver figura A.2).

Los atributos llamados *Properties* (Propiedades), son localizados en los nodos y hojas. El nombre del fabricante de un dispositivo, por ejemplo, puede ser designado como una propiedad. La propiedad *AccessPath* puede también ser disponible en un *ítem*. Este puede ser usado para describir la ruta de comunicación entre el Servidor y el dispositivo (ejemplo: COM1, 9600kBit/s) en más detalle. La propiedad *AccessPath* es opcional y puede ser omitido.

6 Jerarquía de los Objetos de OPC

Un cliente con Acceso de Datos de OPC puede crear muchos objetos de OPC en un servidor para definir el proceso. El objeto *OPCServer* está en el nivel superior del objeto en la jerarquía de los objetos de OPC. Los objetos *OPCGroup* forman el siguiente nivel. El último nivel es formado por los objetos *OPCItem*, tal como se puede apreciar en la figura A.2. Los objetos *OPCItem* representan hojas o propiedades de hojas del *namespace*. Los objetos *OPCGroup* son usados para estructurar los objetos *OPCItem*.

La figura A.2 muestra un objeto jerárquico y las relaciones de los items con el *namespace*, también muestra que se han creado dos objetos *OPCGroup* y hay dos objetos *OPCItem* por cada objeto *OPCGroup*.

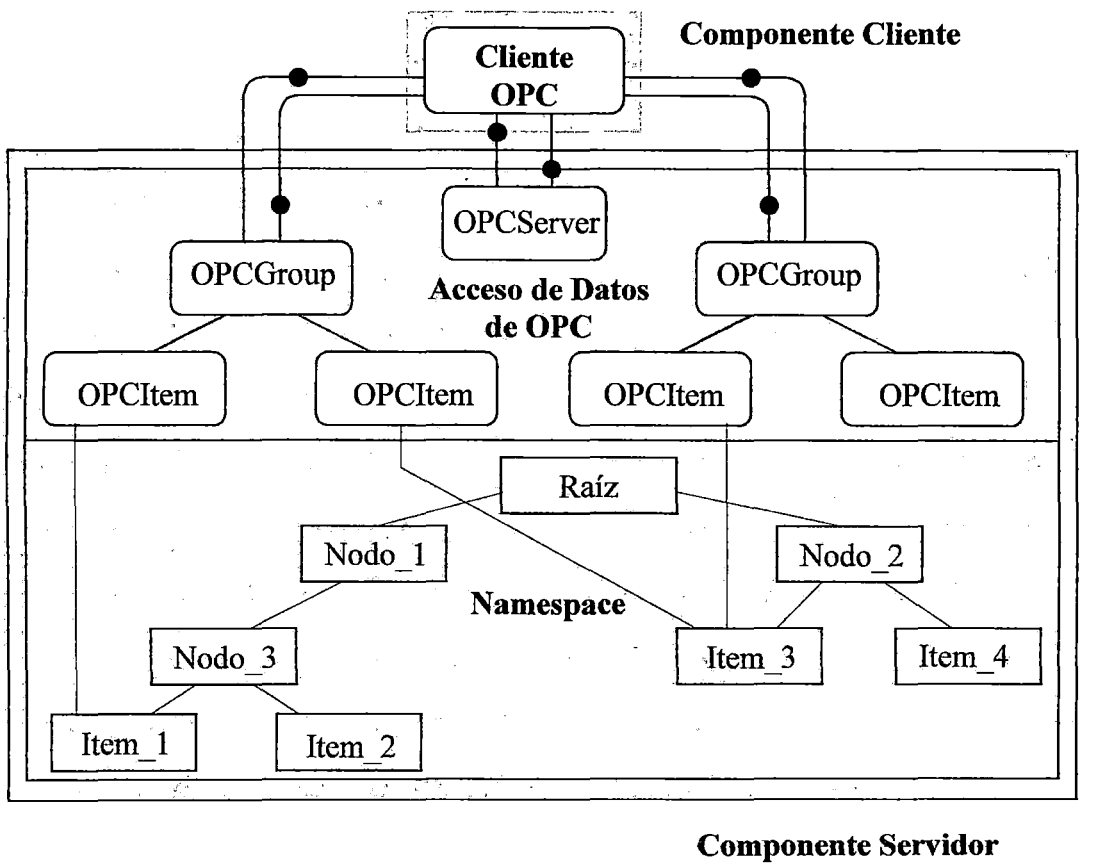


Figura A.2 Relación entre el *namespace* y la jerarquía de los objetos de OPC

7 Formato de los Datos que Intercambian el Cliente y el Servidor de OPC

El formato en el cual los datos son intercambiados entre el servidor y el cliente de OPC está establecida. Este formato es mostrado en la figura A.3 el cual consiste en tres componentes:

- **El dato actual.** Es usado para el intercambio de datos entre el servidor y el cliente y viceversa, todas las especificaciones de OPC usan los tipos de datos de DCOM llamados tipos de datos VARIANT.
- **Tiempo de Actualización (*TimeStamp*).** Esta propiedad tiene una longitud de 8 bytes e indican los tiempos desde la fecha 01.01.1601 en intervalos de 100ns.
- **Información del Estado.** Está compuesto por 2 bytes que contienen la información del estado para el dato transmitido. Por ahora, sólo el uso del byte menos significativo es especificado. Dos bits son usados para describir la calidad (*Quality*) del valor. El valor puede ser *Good*, *Bad*, o *Uncertain*.

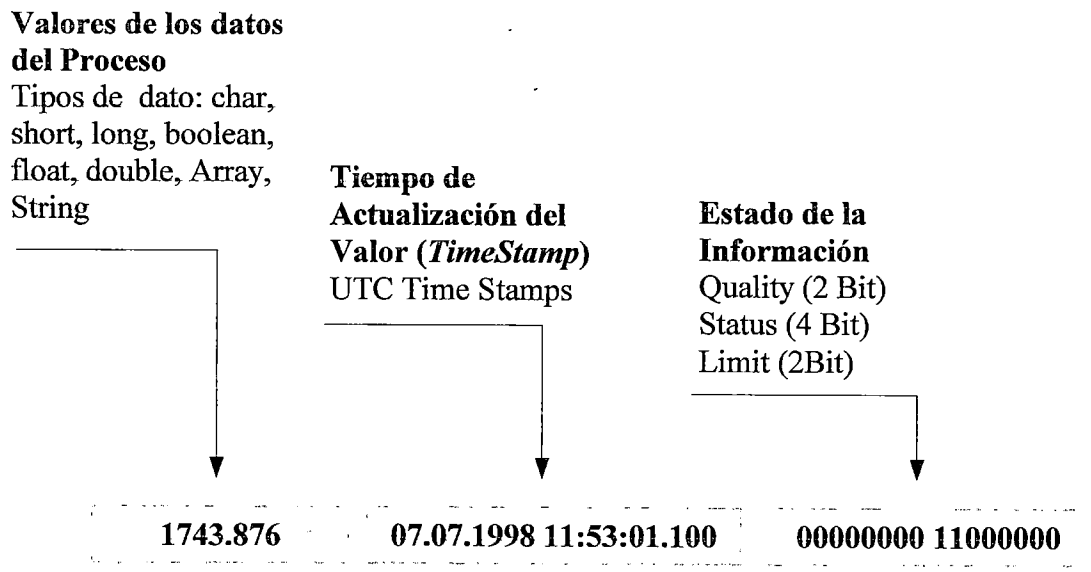


Figura A.3. Formato de los datos de la especificación acceso de datos de OPC

8 Acceso a la Información dentro del *Namespace*

El cliente primeramente pide la estructura del *namespace*. A modo de ejemplo se puede apreciar la figura A.4, donde el resultado de esta llamada es del tipo jerárquico. Comenzando desde la raíz, el cliente obtendrá la información de los nodos y hojas. El pedido puede pasar diferentes valores de parámetros dependiendo del método, el cual se puede habilitar para obtener lo siguiente:

- Sólo nodos o ramas debajo del nodo actual (ejemplo, Sala_1 en la figura A.4)
- Sólo hojas debajo del nodo actual (ejemplo, Temperatura, humedad en la figura A.4)
- Todos debajo del nodo actual, el cliente recibe una cadena conteniendo todos los identificadores de los nodos y hojas debajo del nodo actual. En esta forma, un *namespace* plano (*flat*) puede ser modelado en el cliente.

Con respecto a las hojas, se podrían realizar restricciones que pueden ser hechas usando los siguientes filtros:

- Usando un específico tipo de dato (ejemplo, unsigned integer de 2 bytes)
- Usando un específico acceso (ejemplo, de sólo lectura o de sólo escritura)
- Usando el correspondiente nombre (ejemplo, Temperatura*)

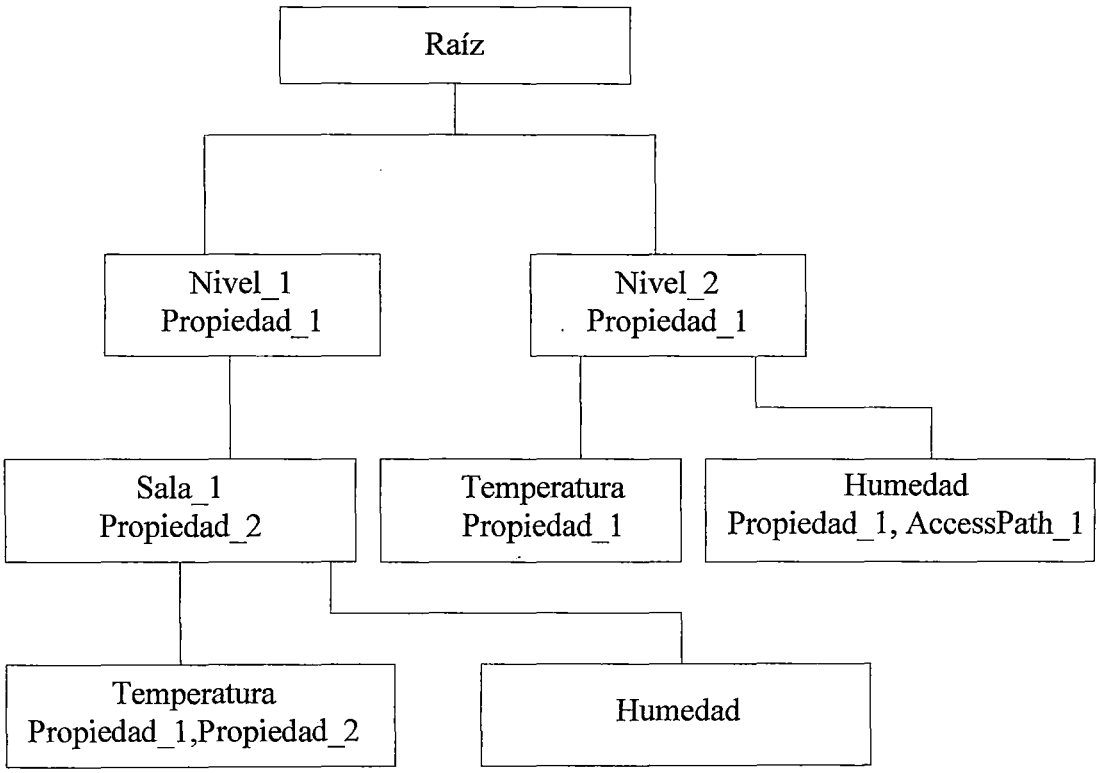


Figura A.4. *Namespace* de un servidor de OPC

9 Creación del Objeto *OPCGroup*

Cuando se hace el pedido para la creación de un objeto *OPCGroup*, el cliente transmite los valores de los siguientes parámetros al servidor:

- *Nombre simbólico*, describe al grupo. Este nombre debe de ser único.
- *RequestedUpdateRate*, este parámetro influencia en la velocidad con el cual los valores de los objetos *OPCItem* activos son escaneados.
- *PercentDeadBand*, este parámetro influye cuando los valores son automáticamente enviados hacia el cliente. Éste especifica el porcentaje del valor del ítem que se tomará en cuenta, para notificar al cliente de OPC de un cambio producido en el valor del ítem.
- *ActiveState*, con un estado inactivo del objeto *OPCGroup* (ActiveState = False), las variables de proceso no son leídas automáticamente.

Los parámetros *RequestedUpdateRate, PercentaDeadBand,* y *ActiveState* controlan la transferencia automática de datos hacia el cliente de OPC.

10 Creación del Objeto *OPCItem*

A diferencia de la llamada del método para la creación de los objetos *OPCGroup*, el cliente puede generar más de un objeto *OPCItem*. Al hacer esto, el cliente pasa los valores de los siguiente párametros:

- *fully qualified ItemId,* identifica claramente un ítem en el *namespace.* Con este identificador el servidor puede crear un objeto *OPCItem* para este ítem.
- *ActiveState*, este puede ser activo o inactivo, el cual significa que el objeto puede ser tomado en cuenta o no, respectivamente, cuando se realiza la lectura automática de la variable del proceso.
- *RequestedDataType*, con este parámetro el cliente puede pedir un tipo de dato con el cual desea leer o escribir. Luego el servidor levará a cabo una conversión durante las llamadas. Si el servidor no es capaz de llevar a cabo la conversión requerida, informará al cliente usando un parámetro de retorno.
- *AccessPath* (Opcional).
- *ClientHandle,* identificador creado por el cliente.

Los servidores retornan los siguientes parámetros al cliente después de terminar la llamada del método:

- *CanonicalDataType*, tipo de dato original.
- *Serverhandle*, identificador creado por el servidor.

El parámetro *fully qualified ItemId* no es suficiente para una única referencia del objeto *OPCItem*, ya que con el mismo identificador, un cliente puede crear muchos objetos *OPCItem* en diferentes o idénticos objetos *OPCGroup*. Además, muchos clientes pueden también crear muchos objetos *OPCItem* en un sólo

componente del servidor con el mismo identificador *fully qulified ItemId*. En estos dos casos es muy conveniente usar los parámetros *ClientHandle* y *Serverhandle* para procedimientos de lectura y escritura.

Dos clientes pueden usar el mismo parámetro *ClientHandle*. El servidor asigna diferentes parámetros *Serverhandles* con la finalidad de garantizar que sean únicos con su otro par. Otra razón para la existencia de estos parámetros o manejadores, es simplificar el acceso de información con los objetos *OPCItem* el cual el cliente y servidor tienen que manejar. El cliente usa el manejador del servidor (*Serverhandle*) para borrar y cambiar las propiedades del objeto en el servidor. Si el servidor transfiere valores hacia el cliente, el último usa el manejador del cliente *ClientHandle* para la localización del ítem.

El cliente de OPC es capaz de borrar los objetos *OPCItem* invocando el método *RemoveItems*. El direccionamiento es hecho vía los manejadores *ClientHandles* y *ServerHandles*.

11 Lectura y Escritura de valores desde el Cliente de OPC

Existen 2 tipos de operaciones de lecturas especificadas por OPC, los cuales son:

- Lectura *síncrona*, el cliente llama a este método y espera el valor retornado, sólo se usan si el acceso a los datos es rápido.
- Lectura *asíncrona*, el cliente llama a este método e inmediatamente regresa. Después de un cierto intervalo el cliente obtiene el valor deseado. La lectura asíncrona es usada si la obtención del dato al servidor toma un tiempo considerable.

Otro tipo de lectura es posible con el método *Refresh*, el cual trabaja cuando el cliente lee todos los valores de los objetos *OPCItem* activos de un objeto *OPCGroup* también activo. La tabla A.2 muestra los dos tipos de fuentes de dato (cache interna del servidor de datos o *Cache* y fuente de datos o *Device*) especificados por OPC, los cuales pueden ser accedidos por los métodos de lectura.

	Cache	**Dispositivo (*Device*)**
Lectura síncrona	*	*
lectura asíncrona		*
Refresh	*	*

Tabla A.2 Tipos de fuentes de dato para los métodos de lectura

Otro tipo de intercambio de datos es producido cuando un valor de un dato ha cambiado y el servidor transfiere al cliente este suceso. Este intercambio es realizado por medio del evento *DataChange*.

Existen 2 tipos de operaciones de escritura especificadas por OPC, los cuales son:

- Escritura *síncrona*, Este procedimiento siempre tiene que ser direccionado al dispositivo.
- Escritura *asíncrona*, El cliente inicializa la ejecución del método, éste a su vez retorna inmediatamente después del llamado del método. El pedido actual es puesto en una cola en el servidor de OPC. Después de procesar la información por el servidor, el cliente recibe el resultado vía una comunicación de vuelta (*callback*). El pedido y la respuesta son hechos por medio de un identificador de la transacción, el cual es llamado *transactionID*.

La figura A.5 explica el más frecuente intercambio de datos entre el servidor y el cliente (evento *Datachange*). El servidor calcula el valor absoluto de la diferencia del rango de las unidades de ingeniería (30, figura A.5); el resultado es multiplicado por el valor del parámetro *PercentDeadband* para un objeto *OPCGroup* (en la figura A.5 el valor es 0.1 y el resultado es 3). El servidor lee los objetos *OPCItem* activos en un intervalo de tiempo determinado por el valor del parámetro *UpdateRate* (1000ms, figura A.5).

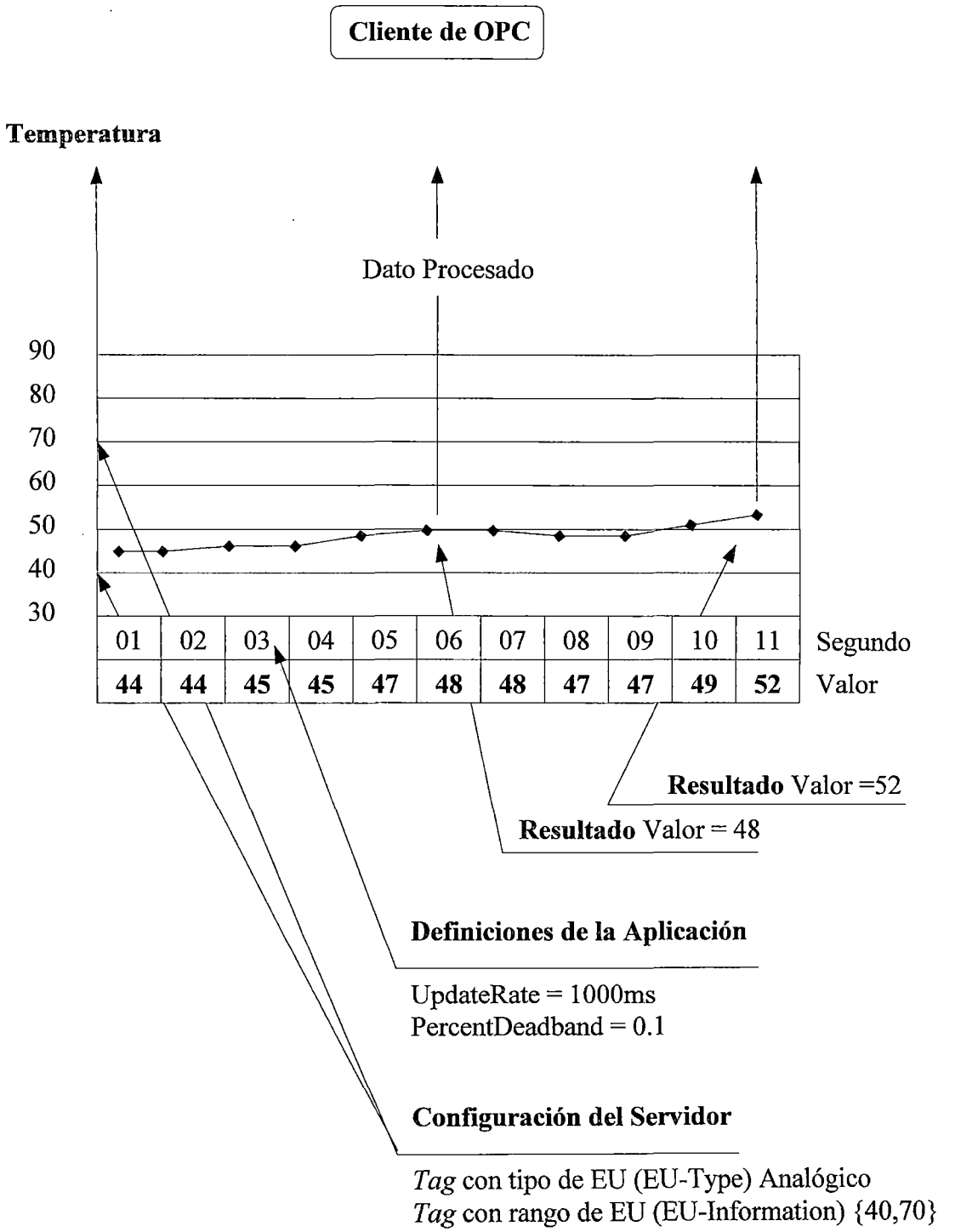


Figura A.5 Configuración de parámetros y flujo en el tiempo para la transmisión de datos

Además, la diferencia entre el último valor enviado y el valor actual leído de un objeto *OPCItem* es calculada en ciclos. Si el valor absoluto de esta diferencia es mayor que el valor calculado en la primera parte (valor 3), entonces el valor actual leído del objeto *OPCItem* es enviado hacia el cliente. Para la figura A.5 ocurre dos veces (valores 48 y 52). El servidor lleva a cabo este cálculo para todos los objetos *OPCItem* activos del objeto *OPCGroup* también activo. Todos los valores que cumplan con el requerimiento ya mencionado son enviados al cliente.

Este tipo de adquisición de datos puede solamente ser usado en la forma descrita arriba. No es aplicable para los siguientes casos:

- Con tipos de datos cuyo tipo de unidad de ingeniería (EU) no es igual a la analógica (la especificación también lo define como "Enumerated"), Estos valores son pasados al cliente cada vez que ellos han cambiado.
- Con valores de tipo de datos estructurados (ejemplo una cadena de variables tipo byte). Los valores estructurados son pasados por el servidor al cliente cada vez que ellos han cambiado de valor.

Los datos también son transferidos automáticamente hacia el cliente si el estado (propiedad *Status*) relacionado al objeto *OPCItem* cambia. Por ejemplo: pasa de *Good* a *Bad.*

12 Grupos Públicos y Privados

Durante el desarrollo de la presente tesis cuando se habla de grupos implícitamente ser refiere a los privados, porque pertenecen al cliente que los creó. A veces es necesario tener una visión de los grupos, pero de otros clientes. En este caso, se usan los grupos públicos, los cuales son visibles y además pueden ser usados por otros clientes.

13 Modelo de los objetos de la Interface de Automatización para el Acceso de Datos

La figura A.6 muestra el modelo de los objetos que son disponibles cuando se usa la librería dinámica "Automation Wrapper" y sus respectivas descripciones son mostradas en la tabla A.3.

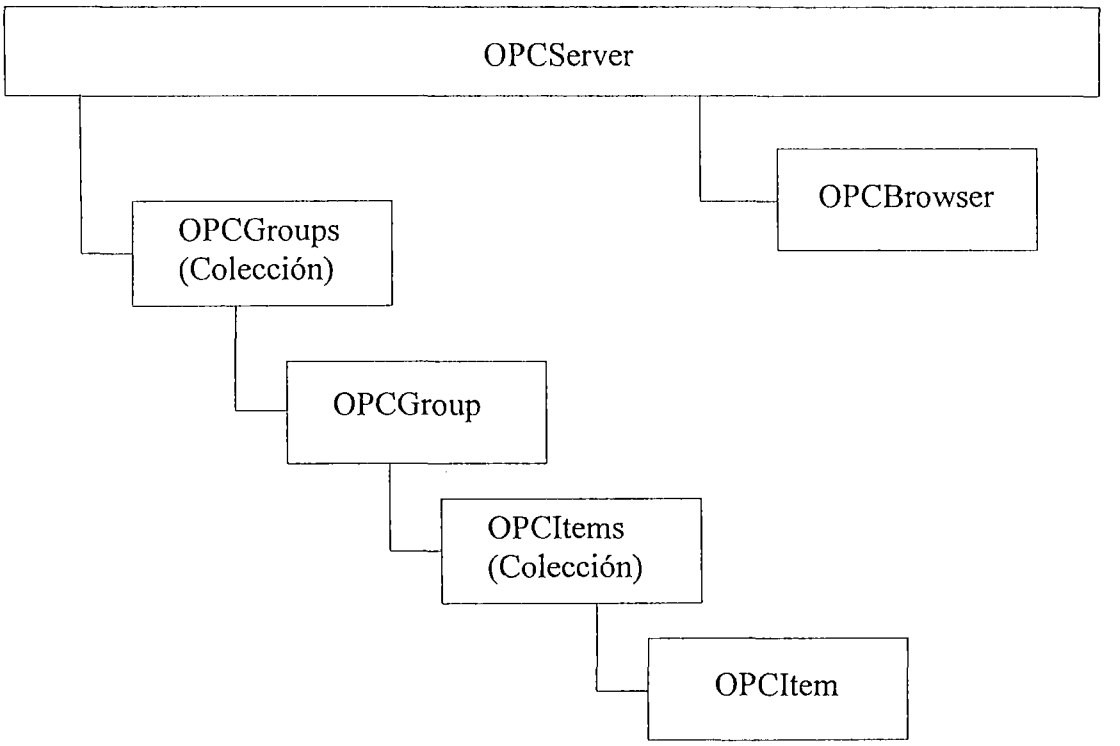


Figura A.6 Jerarquía de los objetos de la Interface de Automatización para el acceso de datos

Objeto	Descripción
OPCServer	Se debe crear un objeto *OPCServer* antes de poder obtener referencias de otros objetos. Éste contiene la colección *OPCGroups* y permite crear el objeto *OPCBrowser*.
OPCGroups	Una colección que contiene todos los objetos *OPCGroup* que el cliente ha creado, dentro del alcance de un objeto *OPCServer*.
OPCGroup	El propósito de este objeto es mantener la información y proveer el mecanismo para la adquisición de datos.
OPCItems	Una colección que contiene todos los objetos *OPCItem* que el cliente ha creado dentro del alcance del objeto *OPCServer*.
OPCItem	Un objeto que contiene las definiciones del ítem, valor actual, información del estado del mismo y el tiempo de actualización. Se debe tener en cuenta que la Interface de Cliente (*Custom Interface*) no provee un objeto separado para el ítem.
OPCBrowser	El objeto *OPCBrowser* es una colección de ramas o nombres de items que existen en un servidor. Este objeto es opcional. Si el servidor no soporta este objeto, el método *CreateBrowser* no creará este objeto.

Tabla A.3 Descripcion de los objetos de la Interface de Automatización

14 Tipos de Implementación para los Servidores de OPC

Un servidor de OPC puede ser implementado en tres versiones:

- **Librería Dinámica o DLL (InProc Server):** En este caso, el servidor se ejecuta en el mismo espacio de proceso con el Cliente de OPC.
- **Servidor Local o Remoto (OutProc Server):** El servidor es implementado como un ejecutable independiente. Éste es iniciado por el Cliente de OPC. El servidor puede estar localizado en la misma PC con el cliente o en una PC remota.
- **Servicio para Windows NT/2000:** Este servicio es un tipo especial de Servidor Local o Remoto. Éste puede ser inicializado inmediatamente cuando el sistema operativo se carga.

Las ventajas y desventajas para cada tipo de implementación son mostradas en la tabla A.4.

	Ventajas	Desventajas
Servidor InProc	Mejor rendimiento	No se puede acceder simultáneamente a muchos clientes si los recursos usados están protejidos.
Servidor OutProc	El servidor puede ser conectado directamente.	Mal rendimiento comparado con el servidor *InProc*.
Servicio	Las tareas de iniciación de larga duración pueden ejecutarse en paralelo con la carga del proceso de la PC.	No es portátil y no puede ser usado en cualquier lugar. Para sistemas operativos que no son de Windows, como Unix, Linux, etc. Para estos sistemas este servicio no existe.

Tabla A.4 Ventajas y desventajas para cada tipo de servidor

Nota:

Para más detalle consultar con las especificaciones:

- "Data Access Automation Interface Standard" versión 2.02
- "Data Access Custom Interface Standard" versión 2.05

APÉNDICE B

OBJETOS, DEFINICIONES Y SÍMBOLOS DE LA INTERFACE DE AUTOMATIZACIÓN ESTÁNDAR ACCESO DE DATOS DE OPC VERSIÓN 2.02

1 Objeto OPCServer

1.1 Resumen de Propiedades

Propiedad	Descripción
Start Time	Propiedad de sólo lectura que retorna el tiempo en que el servidor ha comenzado ha trabajar.
MajorVersion	Propiedad de sólo lectura que retorna la parte mayor del número de la versión del servidor (ejemplo, "1" en la versión 1.32.33).
VendorInfo	Propiedad de sólo lectura que retorna una cadena con la información del fabricante del servidor.

Tabla B 1 Resumen de propiedades del objeto *OPCServer* (*continúa...*)

Propiedad	Descripción
Bandwidth	Propiedad de sólo lectura. Se usa como un porcentaje del Ancho de Banda (*Bandwidth*) disponible. El valor será hFFFFFFFF, cuando el servidor no calcula un valor para esta propiedad.
ServerName	Propiedad de sólo lectura que retorna el nombre del servidor con el cual el cliente se ha conectado por el método *Connect*.
Current Time	Propiedad de sólo lectura que retorna el tiempo actual del servidor.
MinorVersion	Propiedad de sólo lectura que retorna la parte menor del número de la versión del servidor (ejemplo, "32" en la versión 1.32.33).
ServerState	Propiedad de sólo lectura que retorna el estado del servidor, el cual puede ser uno de los valores de *OPCServerState* (ver Apéndice B, tabla B 19)
OPCGroups	Propiedad de sólo lectura. Esta propiedad es una colección de objetos *OPCGroup*.
ServerNode	Propiedad de sólo lectura que retorna el nombre del nodo de la máquina, donde del servidor se encuentra ubicado.
LastUpdateTime	Propiedad de sólo lectura que retorna el último tiempo de actualización desde el servidor.
BuildNumber	Propiedad de sólo lectura que retorna el número final de la versión del servidor (ejemplo, "33" en la versión 1.32.33).
LocaleID	Propiedad de lectura y escritura que define el lenguaje con el cual se mostrarán las descripciones de los errores. Esta propiedad será usada por el método *GetErrorString*.
PublicGroupNames	Propiedad de sólo lectura que retorna los nombres de los *grupos públicos* (ver Apéndice A) presentes en un servidor. Estos nombres pueden ser usados en el método *ConnectPublicGroup*.
ClienteName	Propiedad de lectura y escritura que permite opcionalmente que el cliente se registre con un nombre con el servidor.

Tabla B 1 (*Continuación de la página 278.*) Resumen de propiedades del objeto

OPCServer

1.2 Resumen de Métodos

Método	Descripción
GetOPCServers	Retorna los nombres (*ProgID*) de los servidores de OPC registrados. Se usa uno de estos nombres como parámetro del método *Connect*.
CreateBrowser	Crea un objeto *OPCBrowser*.
QueryAvailableProperties	Retorna los lenguajes (*LocaleID)* disponibles por el servidor.
Connect	Se debe llamar este método para establecer la conexión con un Servidor que soporta la especificación Acceso de Datos de OPC (OPC Data Access).
GetErrorString	Convierte un código de error en una cadena que se puede leer. El servidor retornará la cadena en el leguaje que es especificada con la propiedad *LocaleID*.
GetItemProperties	Retorna una lista de propiedades de los items.
Disconnect	Se usa este método para poder desconectarse desde un servidor de OPC.
QueryAvailableLocaleIDs	Retorna una lista de los lenguajes disponibles por el servidor de OPC.
LookupItemIDs	Retorna una lista de identificadores del ítem (*ItemID*) disponibles.

Tabla B 2 Resumen de métodos del objeto *OPCServer*

1.3 Resumen de Eventos

Evento	Propiedades
ServerShutDown	Este evento es activado cuando el servidor está siendo cerrado y notifica a todos los clientes activos para liberar recursos. Cuando el servidor notifica este evento el cliente debe de eliminar todos los grupos e items creados.

Tabla B 3 Resumen de eventos del objeto *OPCServer*

2 Objeto OPCBrowser

2.1 Resumen de Propiedades

Propiedades	Descripción
Organization	Propiedad de sólo lectura que retorna dos constantes *OPCHierarchical* o *OPCFlat* (ver Apéndice B, tabla B16).
AccessRights	Propiedad de lectura y escritura, que permite filtrar los items pedidos (método *ShowLeafs*) por medio de la propiedad *AccessRight* (ver Apéndice B, tabla B 18).
Filter	Propiedad de lectura y escritura, que permite aplicar un filtro a los nombres pedidos por medio de los métodos *ShowBranches* y *ShowLeafs*.
CurrentPosition	Propiedad de sólo lectura, que indica la posición actual en el *namespace*. Este valor será "" si la propiedad *Organization* es OPCFlat (ver Apéndice B, tabla B 16).
DataType	Propiedad de lectura y escritura, que permite filtrar los items pedidos (método *ShowLeafs*) por medio de la propiedad *DataType*. Esta propiedad por defecto es VT_EMPTY, el cual acepta cualquier tipo de datos.
Count	Propiedad de sólo lectura que retorna el número de items en la colección.

Tabla B 4 Resumen de propiedades del objeto *OPCBrowser*

2.2 Resumen de Métodos

Métodos	Descripción
Item	Retorna un nombre de una hoja o rama. Esto depende de la llamada previa (*ShowBranches* o *ShowLeafs*).
MoveUp	Mueve el buscador un nivel hacia arriba dentro del *namespace*.
MoveTo	Mueve el buscador a una posición absoluta dentro del *namespace*.
ShowBranches	Obtiene una colección con nombres de las ramas que se encuentran en la posición actual dentro del *namespace*.
MoveToRoot	Mueve el buscador hasta la posición inicial del *namespace*.
GetItemID	Retorna la propiedad *ItemID*.
ShowLeafs	Obtiene una colección con nombres de las hojas que se encuentran en la posición actual dentro del *namespace*.
MoveDown	Mueve el buscador un nivel hacia abajo, a partir de la rama actual donde se encuentra dentro del *namespace*.
GetAccessPaths	Retorna la propiedad *AccessPath* para un ítem identificado por la propiedad *ItemID*.

Tabla B 5 Resumen de métodos del objeto *OPCBrowser*

3 Objeto OPCGroups

3.1 Resumen de Propiedades

Propiedades	Descripción
Parent	Propiedad de sólo lectura que retorna la referencia del objeto padre *OPCServer*.
DefaultGroupDeadband	Propiedad de lectura y escritura, el cual provee la propiedad *DeadBand* por defecto para el objeto *OPCGroups* y es expresado como un porcentaje (0 a 100).
Count	Propiedad de sólo lectura que especifica el número de objetos *OPCGroup* en esta colección.
DefaultGroupIsActive	Propiedad de lectura y escritura que provee el estado por defecto y que será usado en la creación de objetos *OPCGroups* por medio del método *Add*. Esta propiedad por defecto es verdadera.
DefaultGroupLocaleID	Propiedad de lectura y escritura que provee el lenguaje por defecto para el objeto *OPCGroups*, y que será usado con el método *Add*.
DefaultGroupUpdateRate	Propiedad de lectura y escritura que provee la velocidad (*UpdateRate*) por defecto en milisegundos para el objeto *OPCGroups*, y será usado con el método *Add*. Esta propiedad por defecto es 1000 milisegundos (1 segundo).
DefaultGroupTimeBias	Propiedad de lectura y escritura que especifica la diferencia del tiempo, en minutos, existente entre el cliente o servidor y el dispositivo. Esta propiedad por defecto es 0 minutos.

Tabla B 6 Resumen de propiedades del objeto *OPCGroups*

3.2 Resumen de Métodos

Métodos	Descripción
Item	Retorna un objeto *OPCGroup*.
RemovePublicGroup	Elimina un *grupo público* (ver Apéndice A).
Remove	Elimina un objeto *OPCGroup* usando un manejador o índice.
RemoveAll	Elimina todos los objetos *OPCGroup* y *OPCItem*.
GetOPCGroup	Retorna un objeto *OPCGroup*.
ConnectPublicGroup	Permite conectarse con un *grupo público* (ver Apéndice A).
add	Crea un nuevo objeto *OPCGroup* y lo adiciona a la colección. Las propiedades de este nuevo grupo son determinadas por las definiciones por defecto. Después de que un grupo es adicionado, sus propiedades pueden ser cambiadas.

Tabla B 7 Resumen de métodos del objeto *OPCGroups*

3.3 Resumen de Eventos

Evento	Descripción
GlobalDataChange	Este evento recibe los datos que cambian entre múltiples grupos.

Tabla B 8 Resumen de eventos del objeto *OPCGroups*

4 Objeto OPCGroup

4.1 Resumen de Propiedades

Propiedades	Descripción
Parent	Propiedad de sólo lectura que retorna el objeto padre *OPCServer*.
IsActive	Esta propiedad es de lectura y escritura que controla el estado (*Active*) del grupo. Si un grupo está activo puede adquirir los datos desde el servidor, y en caso contrario no lo hará.
ServerHandle	Propiedad de sólo lectura que el servidor asigna para el grupo. Este manejador identifica de manera única este grupo.
DeadBand	Propiedad de lectura y escritura que es expresada como un porcentaje (0 a 100).
Name	Propiedad de lectura y escritura que indica el nombre del grupo. El nombre del grupo debe de ser único, con respecto a los demás grupos creados por el cliente.
IsSubscribed	Propiedad de lectura y escritura que controla la notificación asíncrona del grupo. Un grupo que es *subscrito* recibe los cambios de los datos desde el servidor.
LocaleID	Propiedad de lectura y escritura que define el idioma, y que será usado por el servidor para retornar una cadena de texto con la descripción de los códigos de error.
UpdateRate	Propiedad de lectura y escritura que especifica que tan rápido trabajará el evento *DataChange*. Esta velocidad es expresada en milisegundos.
IsPublic	Propiedad de sólo lectura que retorna verdadero si este grupo es un *grupo público*, en el otro caso es falso.
ClientHandle	Propiedad de lectura y escritura, este valor es de tipo *long* que es asociada con el grupo. El propósito de esta propiedad es facilitar la ubicación del destino del dato.
TimeBias	Propiedad de lectura y. Esta propiedad especifica la diferencia del tiempo, en minutos, existente entre el cliente o servidor y el dispositivo actual.
OPCItems	Esta propiedad obtiene una colección de objetos *OPCItem*.

Tabla B 9 Resumen de propiedades del objeto *OPCGroup*

4.2 Resumen de Métodos

Método	Descripción
SyncRead	Esta función lee las propiedades *Value*, *Quality* y *TimeStamp* para uno o más items en un grupo.
AsyncWrite	Realiza el proceso de escritura asíncrona de uno o más items en un grupo. Los resultados son retornados por el evento *AsyncWriteComplete* asociado con el objeto *OPCGroup*.
SyncWrite	Realiza el proceso de escritura de uno o más items en un grupo. Los valores son escritos en el dispositivo, y la función no debe retornar hasta que haya verificado que el dispositivo ha aceptado o rechazado el dato.
AsyncRefresh	Genera un evento para todos los items activos en el grupo (si han cambiado o no). Los items inactivos no son incluidos en esta llamada. Los resultados son retornados por medio del evento *DataChange* asociado con el objeto *OPCGroup*, y también por medio del evento *GlobalDataChange* asociado con el objeto *OPCGroups*.
AsyncRead	Lee en forma asíncrona uno o más items en un grupo. Los resultados son retornados por medio del evento *AsyncReadComplete* asociado con el objeto *OPCGroup*. Las lecturas son desde el dispositivo y no son afectados por la propiedad *ActiveState* de un grupo o ítem.
AsyncCancel	Realiza el pedido para que el servidor cancele una transacción. Un evento *AsyncCancelComplete* ocurrirá indicando si se ha cancelado satisfactoriamente o no.

Tabla B 10 Resumen de métodos del objeto *OPCGroup*

4.3 Resumen de Eventos

Eventos	Descripción
DataChange	Este evento es activado cuando la propiedad *Value* o *Quality* de un ítem dentro del grupo han cambiado. Este evento no trabajará más rápido que la propiedad *UpdateRate* del grupo. Solamente los items activos y grupos activos se consideran en el evento *DataChange.*
AsyncCancelComplete	Este evento se produce cuando un método *AsyncCancel* ha finalizado.
AsyncReadComplete	Este evento se activa cuando el método *AsyncRead* ha finalizado.
AsyncWriteComplete	Este evento se produce cuando un método *AsyncWrite* ha finalizado.

Tabla B 11 Resumen de métodos del objeto *OPCGroup*

5 Objeto OPCItems

5.1 Resumen de Propiedades

Propiedades	Descripción
Parent	Propiedad de sólo lectura que retorna un objeto padre *OPCGroup*.
DefaultIsActive	Propiedad de lectura y escritura. Permite cambiar el estado por defecto y será usada en la llamada del método *Add*. Esta propiedad por defecto es verdadera.
DefaultRequestedDataType	Propiedad de lectura y escritura. Permite cambiar el tipo de dato por defecto y será usada en la llamada del método *Add*. Esta propiedad por defecto es VT_EMPTY.
Count	Propiedad de sólo lectura que indica el número de items en la colección.
DefaultAccessPath	Propiedad de lectura y escritura. Permite cambiar la propiedad *AccessPath* por defecto y será usada en la llamada del método *Add*. Esta propiedad por defecto es "".

Tabla B 12 Resumen de propiedades del objeto *OPCItems*

5.2 Resumen de Métodos

Métodos	Descripción
Item	Retorna un objeto *OPCItem* de la colección.
AddItems	Crea objetos *OPCItem* y los adiciona a la colección. Las propiedades de cada nuevo objeto *OPCItem* son determinadas por las propiedades por defecto.
SetActive	Permite la activación y desactivación de un objeto *OPCItem* de una colección *OPCItems*.
GetOPCItem	Retorna un objeto *OPCItem* usando el manejador *ServerHandle,* el cual fue retornado por el método *Add.*
Remove	Elimina un objeto *OPCItem.*
SetClientHandles	Cambia el manejador *ClientHandles* de uno o más items en un grupo.
AddItem	Crea un nuevo objeto *OPCItem* y lo adiciona a la colección. Las propiedades de este nuevo objeto *OPCItem* son determinadas por las propiedades por defecto. Después que un objeto *OPCItem* es adicionado, sus propiedades pueden ser modificadas.
Validate	Determina si uno o más objetos *OPCItem* puedan ser creados satisfactoriamente al usar el método *Add.*
SetDataTypes	Cambia la propiedad *DataType* para uno o más items.

Tabla B 13 Resumen de métodos del objeto *OPCItems*

6 Objeto OPCItem

6.1 Resumen de Propiedades

Propiedades	Descripción
Parent	Propiedad de sólo lectura que retorna un objeto padre *OPCGroup*.
AccessPath	Propiedad de sólo lectura, el cual fue especificado por el cliente por medio del método *Add*.
IsActive	Propiedad de lectura y escritura. Si el valor de esta propiedad es falso el ítem estará inactivo, en caso contrario estará activo.
Quality	Propiedad de sólo lectura que retorna el último valor leído de la propiedad *Quality* desde el servidor.
EUType	Propiedad de sólo lectura que indica el tipo de información de las unidades de ingeniería (EU) contenidas en la propiedad *EUInfo*. Si el valor es 0 entonces la información no está disponible (EUInfo retorna VT_EMPTY), si el valor es 1 (*Analog*) la propiedad *EUInfo* contendrá un arreglo, el cual será el rango de la unidad de ingeniería. Si el valor es 2 (*Enumerated*) la propiedad *EUInfo* contendrá una cadena, el cual será una lista de cadenas (por ejemplo, "OPEN", "CLOSE", etc.) correspondientes a los valores numéricos (0, 1, etc.).
ClientHandle	Propiedad de lectura y escritura cuyo propósito es poder tener un fácil acceso al dato. Este valor normalmente es un índice.
AccessRights	Propiedad de sólo lectura el cual retorna la propiedad *AccessRight* de este ítem.
RequestedDataType	Propiedad de lectura y escritura, el cual indica el valor de la propiedad *DataTtype*.
TimeStamp	Propiedad de sólo lectura que retorna el último valor de la propiedad *TimeStamp* leído desde el servidor.

Tabla B 14 Resumen de propiedades del objeto *OPCItem* (*continúa...*)

Propiedades	Descripción
EUInfo	Propiedad de sólo lectura que contiene la información de las unidades de ingeniería.
ServerHandle	Propiedad de sólo lectura que el servidor asigna a un objeto *OPCItem*. Éste es usado en algunos métodos del objeto *OPCItem* tales como *Remove*.
ItemID	Propiedad de sólo lectura que identifica de manera única a un ítem.
Value	Propiedad de sólo lectura que retorna el último valor leído desde el servidor.
CanonicalDataType	Propiedad de sólo lectura que retorna el tipo de dato nativo del ítem en el servidor.

Tabla B 14 (*Continuación de la página 290.*) Resumen de propiedades del objeto *OPCItem*

6.2 Resumen de Métodos

Métodos	Descripción
Read	Realiza la lectura síncrona (*blocking call*) del ítem desde el servidor.
Write	Realiza el proceso de escritura síncrona (*blocking call*).

Tabla B 15 Resumen de métodos del objeto *OPCItem*

7 Definiciones y Símbolos

7.1 Tipos de Namespace (*OPCNamespaceTypes*)

Símbolo	Descripción
OPCHierarchical	Constante que vale uno e indica que el *namespace* es del tipo árbol, es decir los items son mostrados como hojas dentro de ramas.
OPCFlat	Constante que vale dos e indica que el *namespace* tiene un sólo nivel, es decir los items son mostrados en forma de lista.

Tabla B 16 Constantes para los tipos de *namespace*

7.2 Tipos de Fuentes de datos (*OPCDataSource*)

Símbolo	Descripción
OPCCache	Constante que vale 1 e indica que los datos se obtienen desde la *cache*.
OPCDevice	Constante que vale 2 e indica que los datos se obtienen desde el dispositivo (*device*).

Tabla B 17 Constantes para los tipos de fuentes de datos

7.3 Tipos de Acceso (*OPCAccessRights*)

Símbolo	Descripción
OPCReadable	Constante que vale 1 que indica que el ítem es solamente de lectura.
OPCWritable	Constante que vale 2 que indica que el ítem es solamente de escritura

Tabla B 18 Constantes para los tipos de acceso

7.4 Estados del Servidor (*OPCServerState*)

Símbolo	Valor	Descripción
OPCRunning	1	El servidor está trabajando normalmente. Este es el estado usual para un servidor.
OPCFailed	2	Un error fatal de un fabricante específico ha ocurrido en el servidor. El servidor ya no funcionará. El procedimiento de recuperación de esta situación depende de cada fabricante. Un código de error E_FAIL generalmente se retorna desde cualquier otro método del servidor.
OPCNoconfig	3	El servidor está trabajando, pero no tiene la información de configuración cargado, y por eso no puede trabajar normalmente. Este estado implica que el servidor necesita la información de configuración para poder funcionar. Los servidores que no requieren la información de configuración no deben retornar este estado.

Tabla B 19 Símbolos para los estados del servidor de OPC (*continúa...*)

Símbolo	Valor	Descripción
OPCSuspended	4	El servidor ha sido temporalmente suspendido vía algún método de cada fabricante específico y no está recibiendo ni enviando datos. La propiedad *Quality* será OPC_QUALITY_OUT_OF_SERVICE.
OPCTest	5	El servidor está en modo de prueba. Las salidas están desconectadas del hardware, pero el servidor trabaja normalmente. Las entradas pueden ser reales o pueden ser simuladas dependiendo de la implementación del servidor.
OPCDisconnected	6	El objeto servidor de la Interface de Automatización (*Automation Interface*) no está conectado a una Interface de Cliente (*Custom Interface*) de OPC.

Tabla B 19 (*Continuación de la página 293.*) Símbolos para los estados del servidor de OPC

7.5 Tipos de Errores (*OPCErrors*)

Error de OPC	Valor	Descripción
OPCInvalidHandle	0xC0040001L	El valor del manejador no es válido. Nota: Un cliente nunca debe pasar un manejador inválido al servidor.
OPCBadType	0xC0040004L	El servidor no puede convertir el dato entre el formato específico o tipo de dato pedido al tipo de dato *canonical* o nativo.
OPCPublic	0xC0040005L	La operación de pedido no puede ser hecha en un *grupo publico*.
OPCBadRights	0xC0040006L	La propiedad *AcessRight* del ítem no permiten esta operación.
OPCUnknownItemID	0xC0040007L	El identificador del ítem (*ItemID*) no está definida en el *namespace* del servidor (para la adición o validación) o no existe (para la lectura o escritura).

Tabla B 20 Símbolos para los tipos de errores (*continúa...*)

Error de OPC	Valor	Descripción
OPCInvalidItemID	0xC0040008L	El identificador del ítem (*ItemID*) no tiene la sintaxis del servidor.
OPCInvalidFilter	0xC0040009L	La cadena del filtro no es válida.
OPCUnknownPath	0xC004000AL	La ruta de acceso del ítem (*AccessPath*) no es conocida por el servidor.
OPCRange	0xC004000BL	El valor está fuera del rango.
OPCDuplicateName	0xC004000CL	Nombre duplicado no es permitido.
OPCUnsupportedRate	0x0004000DL	El servidor no soporta la velocidad del dato pedido, pero usará la velocidad más cercana disponible.
OPCClamp	0x0004000EL	Un valor ingresado para la escritura fue aceptado, pero la salida fue mantenida.
OPCInuse	0x0004000FL	La operación no puede ser llevada a cabo porque el objeto está siendo referenciada.
OPCInvalidConfig	0xC0040010L	El archivo de la configuración del servidor tiene un formato inválido.
OPCNotFound	0xC0040011L	El objeto pedido (ejemplo un grupo público) no fue encontrado.
OPCInvalidPID	0xC0040203L	El identificador de la propiedad (*propertyID*) ingresado no es válido para el ítem.

Tabla B 20 (*Continuación de la página 294.*) Símbolos para los tipos de errores

Nota:

Para más detalle consultar con las especificaciones:

- "Data Access Automation Interface Standard" versión 2.02
- "Data Access Custom Interface Standard" versión 2.05

APÉNDICE C

DEFINICIÓN DE LAS FUNCIONES, ESTRUCTURAS Y DEFINICIONES LA HERRAMIENTA *SOFTING XPRESS OPC SERVER* V.3.10

1 Funciones DLL

1.1 SOXpSInitialize

Esta función provee los datos de la aplicación específica del servidor de OPC.

```
SOXpSInitData *SOXpSInitialize(
IN SOXpSCompletedDARequests completedReq,
IN SOXpSFireAEEvents fireEvents,
IN SOXpSChangeAEConditions changeCond,
IN SOXpSShutdown shutdown,
IN SOXpSSetDeviceState setDeviceState)                    (C.1)
```

Parámetros:

completedReq	Un puntero a una función usado para reportar el final de un pedido de lectura o escritura.
fireEvents	Un puntero a una función usado para lanzar los eventos de la especificación Alarmas y Eventos de OPC.
changeCond	Un puntero a una función usado para reportar los estados de cambio de las condiciones de la especificación Alarmas y Eventos de OPC.
shutdown	Un puntero a una función usado para cerrar el servidor de OPC.
setDeviceState	Un puntero a una función para poner la información de diagnóstico.

Valores Retornados:

Un puntero a una estructura *SOXpSInitData* estática que contiene los datos de la aplicación específica.

1.2 SOXpSStart

Esta función es llamada después de la iniciación del servidor de OPC.

DWORD SOXpSStart(void) (C.2)

Valores Retornados:

TRUE	El Servidor de OPC debería entrar al modo de operación
FALSE	El Servidor de OPC debería cerrarse.

1.3 SOXpSStop

Esta función es llamada cuando el servidor de OPC está apunto de cerrarse.

```
void SOXpSStop(void)                                    (C.3)
```

1.4 SOXpSBuildDANameSpace

Con esta función se puede crear el *namespace* de la especificación Acceso de Datos de OPC.

```
void SOXpSBuildDANameSpace(
IN SOXpSCreateNode createNode,
IN SOXpSCreateTag createTag,
IN SOXpSCreateProperty createProperty)                  (C.4)
```

Parámetros:

createNode	Un puntero a una función para la creación de nodos.
createTag	Un puntero a una función para la creación de *tags*.
createProperty	Un puntero a una función para la creación de propiedades.

1.5 SOXpSHandleDARequests

Esta función es llamada para el procesamiento de los pedidos pendientes de escritura y lectura de la especificación Acceso de Datos de OPC.

```
void SOXpSHandleDARequests(
IN DWORD ioOperation,
IN DWORD numRequests,
IN SOXpSDARequestData *requestArray)                    (C.5)
```

Parámetros:

ioOperation	Tipo de operación (escritura o lectura). Éste puede ser uno de las banderas *SOXPS_REQUEST_OPERATION* (ver manual de usuario del *ToolKit*).
numRequests	Especifica el número de estructuras en el arreglo apuntado por el parámetro *requestArray*.

requestArray — Puntero a un arreglo de estructuras *SOXpSDARequestData* que contienen el pedido de datos.

2 Funciones *Xpress*

2.1 SOXpSCreateNode

La librería de cliente (librería desarrollada para la comunicación con el dispositivo) llama esta función para crear un objeto *Nodo* en el *namespace* de la especificación Acceso de Datos de OPC.

```
DWORD SOXpSCreateNode(
IN SOXpSNodeHandle parentNodeHandle,
IN LPCTSTR name,
OUT SOXpSNodeHandle *nodeHandle)                    (C.6)
```

Parámetros:

parentNodeHandle — Manejador para el objeto padre. Éste debe ser un manejador de nodo válido, el cual es retornado por una llamada previa a la función *SOXpSCreateNode* o NULL para crear un nodo del nivel superior.

name — El nombre del nodo.

nodeHandle	Manejador retornado para el nodo creado. Si el puntero al *nodeHandle* es NULL, no es retornado ningún manejador.

Valores Retornados:

TRUE	El objeto fue creado satisfactoriamente.
FALSE	La creación del objeto falló.

2.2 SOXpSCreateTag

La librería de cliente (librería desarrollada para la comunicación con el dispositivo) llama esta función para crear un objeto *Tag* en el *namespace* de la especificación Acceso de Datos de OPC.

```
DWORD SOXpSCreateTag(
IN SOXpSNodeHandle parentNodeHandle,
IN LPCTSTR name,
IN VARTYPE datatype,
IN DWORD accessRights,
IN DWORD userData,
OUT SOXpSItemHandle *tagHandle)                    (C.7)
```

Parámetros:

parentNodeHandle	Manejador para el objeto padre. Éste debe ser un manejador de nodo válido, retornado por una llamada previa de la función *SOXpSCreateNode* o NULL para crear un *tag* en el nivel más alto.
name	Nombre del *tag*.
datatype	Tipo de dato del valor del *tag*. Se puede utilizar cualquiera de los tipos de datos VARIANT tales como: VT_UI1, VT_I2, VT_I4, VT_R4, VT_R8, VT_BOOL, VT_BSTR, VT_DATE y VT_ARRAY.
accessRights	El tipo de acceso para este *tag* (sólo lectura, sólo escritura, lectura y escritura).
userData	Cualquier valor. Este valor es guardado con el objeto *tag* y luego es posible acceder a ello por medio de los pedidos de lectura o escritura por medio el miembro *m_itemUserData* de la estructura *SOXpSDARequestData* (ver Apéndice C, tabla C 1).
tagHandle	Manejador retornado para el objeto creado, si el *tagHandle* apunta a NULL, no es retornado ningún manejador.

Valores Retornados:

TRUE	El objeto ha sido creado satisfactoriamente

FALSE	El objeto no ha sido creado

2.3 SOXpSCreateProperty

La librería de cliente (librería desarrollada para la comunicación) llama esta función para crear un objeto propiedad (*Property*) en el *namespace* de la especificación Acceso de Datos de OPC.

```
DWORD SOXpSCreateProperty(
IN SOXpSNodeOrItemHandle parentHandle,
IN DWORD propertyID,
IN LPCTSTR description,
IN VARTYPE datatype,
IN DWORD accessRights,
IN LPVARIANT constantValue,
IN DWORD userData,
OUT SOXpSItemHandle *propertyHandle)                    (C.8)
```

Parámetros:

parentHandle	Maneja el objeto padre. Este parámetro debe ser un manejador válido para el *tag* o nodo, el cual fue retornado por una llamada previa de la función *SOXpSCreateTag* o

	SOXpSCreateNode, respectivamente (ver Apéndice C, funciones C 6 y C 7).
propertyID	Identificador de la propiedad, definido por la especificación Acceso de Datos de OPC.
description	Descripción de la propiedad. Tipo de dato de la propiedad. Se puede utilizar cualquiera de los tipos de datos VARIANT tales como: VT_UI1, VT_I2, VT_I4, VT_R4, VT_R8, VT_BOOL, VT_BSTR, VT_DATE y VT_ARRAY.
accessRights	El tipo de acceso a esta propiedad (sólo lectura, sólo escritura, lectura y escritura).
constantValue	Un puntero a una variable tipo VARIANT el cual contiene el valor constante de esta propiedad o a un valor NULL, si la propiedad se lee o escribe vía un pedido de escritura o lectura.
userData	Cualquier valor. Este valor es guardado con el objeto propiedad y luego es posible acceder a ello por medio de los pedidos de lectura o escritura por medio el miembro *m_itemUserData* de la estructura *SOXpSDARequestData (*ver Apéndice C, tabla C 1*)*.
propertyHandle	Manejador retornado para el objeto propiedad creado, si el *propertyHandle* apunta a NULL, no es retornado ningún manejador.

Valor Retornado:

TRUE	El objeto fue creado satisfactoriamente.
FALSE	El objeto creado ha fallado.

3 Estructuras

3.1 SOXpSDARequestData

Esta estructura contiene los datos de un pedido de lectura o escritura de la especificación Acceso de Datos de OPC. Esto es usado para transferir los valores del proceso de interés entre la librería de cliente (librería desarrollada para la comunicación) y la herramienta *Xpress OPC Server*.

Miembro	Tipo	Descripción
m_itemHandle	SOXpSItemHandle	Manejador del ítem (puede ser un manejador de un *tag* o propiedad), que su valor va ha ser leído o escrito.
m_itemPath	LPCTSTR	Ruta completa del ítem. Las partes de la ruta del ítem son separadas por un punto.
m_requestHandle	SOXpSRequestHandle	Manejador de este pedido de escritura o lectura.

Tabla C.1 Estructura SOXpSDARequestData (*continúa...*)

Miembro	Tipo	Descripción
m_requestState	DWORD	El estado actual de este pedido. Éste puede uno de los estados *SOXPS_REQUEST_STATE* definidos (ver Apéndice C, tabla C 7).
m_result	HRESULT	Resultado de este pedido.
m_value	VARIANT	El valor que va ha ser leído o escrito.
m_quality	WORD	La calidad de la variable del proceso representado por este ítem después de la escritura o lectura.
m_timeStamp	FILETIME	El tiempo de actualización del valor, ó 0 si el tiempo actual es usado.
m_itemUserData	DWORD	Dato de la aplicación para los items.

Tabla C.1 (*Continuación de la página 305.*) Estructura SOXpSDARequestData

3.2 SOXpSInitData

Esta estructura contiene los datos de la aplicación específica del Servidor de OPC.

Miembro	Tipo	Descripción
m_xpsInitDataVersion	WORD	Versión de la herramienta *XPress OPC Server* en uso.
m_clsidDA	CLSID	Identificador de clase (*ClassID*) del servidor de OPC con Acceso de Datos o CLSID_NULL, si no se provee la funcionalidad Acceso de Datos. Éste debe de ser único.
m_progIdDA	LPCTSTR	*ProgID* de la especificación Acceso de Datos de OPC o NULL, si no se provee la funcionalidad de Acceso de Datos.

Tabla C 2 Estructura SOXpSInitData (*continúa...*)

Miembro	Tipo	Descripción
m_verIndProgIdDA	LPCTSTR	Versión independiente del *ProgID* del servidor de OPC con Acceso de Datos o NULL, si no se provee dicha funcionalidad
m_descriptionDA	LPCTSTR	Descripción del servidor de OPC o NULL, si no se provee la funcionalidad Acceso de Datos.
m_clsidAE	CLSID	Identificador de clase (*ClassID*) del servidor de OPC con Alarmas y Eventos o CLSID_NULL, si no se provee de la funcionalidad Alarmas y Eventos.
m_progIdAE	LPCTSTR	*ProgID* del servidor de OPC con Alarmas y Eventos o NULL, si no se provee dicha funcionalidad.
m_verIndProgIdAE	LPCTSTR	Versión independiente del *ProgID* del servidor de OPC con Alarmas y Eventos o NULL, si no se provee dicha funcionalidad.
m_descriptionAE	LPCTSTR	Descripción del servidor de OPC con Alarmas y Eventos o NULL, si no se provee dicha funcionalidad
m_majVersion	WORD	Mayor versión del servidor.
m_minVersion	WORD	Menor versión del servidor.
m_buildNumber	WORD	Número *Build* del servidor.
m_vendorInfo	LPCTSTR	Información del fabricante.
m_visible	DWORD	TRUE si la ventana de aplicación se va ha mostrar, FALSE en el otro caso.
m_appTitle	LPCTSTR	Título de la ventana de aplicación.
m_resIdIcon	UINT	Identificador de un icono. Si este miembro es 0 se mostrará el icono de Windows estándar.
m_serviceName	LPCTSTR	Nombre del servicio NT o NULL, si el servidor de OPC trabajará como una aplicación estándar.
m_serviceDescription	LPCTSTR	Descripción del servicio NT o NULL, si el servidor de OPC trabajará como una aplicación estándar.
m_serviceDependencies	LPCTSTR	Dependencias del servicio NT, o NULL, si el servidor de OPC trabajará como una aplicación estándar.
m_minUpdateRateDA	DWORD	Mínima *UpdateRate* de un grupo.

Tabla C 2 (*Continuación de la página 306.*) Estructura SOXpSInitData

4 Definiciones

4.1 SOXPS_ACCESSRIGHT

Estas banderas especifican el tipo de acceso para los *tags* y las propiedades en el *namespace* de la especificación Acceso de Datos de OPC.

Operación Pedida	Descripción
SOXPS_ACCESSRIGHT_READABLE	Solamente se habilita la operación de lectura por el cliente.
SOXPS_ACCESSRIGHT_WRITEABLE	Solamente se habilita la operación de escritura por el cliente.

Tabla C 3 Definición SOXPS_ACCESSRIGHT.

4.2 SOXPS_QUALITY

Estas banderas especifican la calidad de una variable de proceso.

Calidad (Quality)	Descripción
SOXPS_QUALITY_GOOD	La calidad del valor es buena.
SOXPS_QUALITY_LOCAL_OVERRIDE	Esto significa que la entrada ha sido desconectada y un valor manual a entrado, el cual ha sido forzado.

Tabla C 4 Banderas que indican la calidad buena de los valores procesados

Calidad (Quality)	Descripción
SOXPS_QUALITY_BAD	El valor es malo, pero no se especifica la razón.
SOXPS_QUALITY_CONFIG_ERROR	Hay algún problema con la configuración del servidor.
SOXPS_QUALITY_NOT_CONNECTED	No conectado.
SOXPS_QUALITY_DEVICE_FAILURE	Un fallo del dispositivo ha sido detectado.
SOXPS_QUALITY_SENSOR_FAILURE	Un fallo de un sensor ha sido detectado. Esta bandera puede ser combinado con las banderas *SOXPS_LIMIT_XXX* (ver Apéndice C, tabla C6).
SOXPS_QUALITY_LAST_KNOWN	La comunicación ha fallado. Pero, el último valor es disponible.
SOXPS_QUALITY_COMM_FAILURE	La comunicación ha fallado. No hay un último valor disponible.
SOXPS_QUALITY_OUT_OF_SERVICE	Este valor no es actualizado.

Tabla C 5 Banderas que indican la calidad mala de los valores

4.3 SOXPS_LIMIT

Estas banderas especifican los límites de una variable de proceso.

Bandera límite	Descripción
SOXPS_LIMIT_OK	El valor no tiene límites
SOXPS_LIMIT_LOW	El valor ha alcanzado un límite inferior.
SOXPS_LIMIT_HIGH	El valor ha alcanzado un límite superior.
SOXPS_LIMIT_CONST	El valor es una constante y no se puede mover.

Tabla C 6 Banderas que indican los límites de una variable de proceso.

4.4 SOXPS_REQUEST_STATE

Estas banderas especifican el estado actual de un pedido de lectura o escritura.

Estado el Pedido	Descripción
SOXPS_REQUEST_STATE_NOT_HANDLED	El pedido del dato no ha sido procesado todavía.
SOXPS_REQUEST_STATE_HANDLED	El pedido del dato ha sido procesado.
SOXPS_REQUEST_STATE_HANDLED_USE_CACHE	El pedido del dato ha sido procesado usando el valor actual de la cache.

Tabla C 7 Banderas que indican los estados de los pedidos de lectura y escritura.

Nota:

Para más detalle consultar con el manual de usuario de la herramienta *Softing Express OPC Server versión 3.10.*

TABLA Nº 4.3.1 a DENSIDAD @ TEMPERATURA ACEITE SAE 10W

TEMP. °C	DENSIDAD gr/cm3
15	0.876
20	0.8681
30	0.8609
40	0.8537
45	0.850
50	0.8465
55	0.843