Universidad Nacional Autónoma de México

Escuela Nacional de Estudios Profesionales Aragón

Análisis de Elementos que Intervienen en el Funcionamiento de un Controlador de Dispositivo de impresión en Windows NT Server

Tesis

QUE PARA OBTENER EL TITULO DE INGENIERO EN COMPUTACION

PRESENTA:

PEDRO GABRIEL RAMIREZ HERNANDEZ.

ASESOR:
INGENIERO ERNESTO PEÑALOZA ROMERO

SAN JUAN DE ARAGON 2003

TESIS
CON
FALLA DE
ORIGEN

Indice **Página**

INTRODUCCION

¿ Porque la Necesidad de un Controlador de Impresión?

La necesidad de un controlador surge en primera instancia cuando se tiene la necesidad de utilizar un dispositivo de impresión semi-compatible, cuyo controlador no es proporcionado por el fabricante, o el controlador proporcionado no tiene soporte para impresión bajo Windows o, en el mejor de los casos, no todas las capacidades de la impresora se pueden acceder desde un sistema operativo como Windows 2000. Este es el caso de algunos impresores de punto de venta, del tipo matriz de puntos o impresión de transferencia térmica.

En este caso, se deberá contar con un controlador bajo Windows, creado por el desarrollador ó programador, que satisfaga nuestras necesidades de control e impresión

¿ Que no todo se vende de ellos?

En este caso, se deberá contar con un controlador bajo Windows, creado por el desarrollador ó programador, que satisfaga nuestras necesidades de control e impresión.

Se tratan indistintamente la terminología de controladores Windows NT/2000, ya que Windows 2000 en cuanto al "kernel" se refiere es prácticamente la versión NT 5.0

¿ Que es un Controlador de Impresora?

Básicamente, los controladores de dispositivo, y en particular los controladores de impresión estándares controlan todas y cada una de sus funciones hardware mediante el envío de comandos de acción llamados ***Comandos Escape (ESC)***.

Son llamados de esta manera, porque básicamente se sigue el estándar de anteponer el carácter ASCII Escape a cualquier comando que se le desee enviar al dispositivo hardware. Cuando el dispositivo impresor recibe el valor hexadecimal del ASCII Escape (1Bh) ó (00011011), significa una orden al dispositivo para que se

prepare a recibir un comando para la ejecución de alguna actividad por parte del impresor. Esto no quiere decir que se le envíe información para imprimir, sino un comando de acción.

De esta manera se puede afirmar que es posible controlar un dispositivo hardware de impresión que cumpla con ciertos estándares internacionales, única y básicamente desde cualquier PC típica y genérica, desde donde se pueda tener acceso a la edición de archivos a escala hexadecimal.

La idea es crear un archivo (binario) que contenga las secuencias de comandos de "Escape" necesarios para ejecutar las acciones de control de hardware que necesitemos ejecute la impresora. Este archivo del que hablamos se debe crear byte por byte con un editor que permita esto a nivel hexadecimal, con el fin de manipular a gusto y conveniencia, el contenido real del archivo, sin ningún tipo de limitante que establece el formato especial.

Una vez que se ha creado, editado y salvado el archivo de secuencias "Escape", este es enviado al dispositivo hardware de impresión por medio de algún puerto de comunicaciones conectado local ó remotamente a nuestra computadora. La forma en que será enviado este archivo, depende de las funciones que el propio Sistema Operativo tenga para enviar el archivo a algún puerto.

Lo anterior es conocido como uno de los controles más básicos que se puede tener de algún dispositivo de impresión desde una PC donde, obviamente es necesario conocer los comandos "Escape" del dispositivo impresor, ya que estos los define y establece el fabricante. En la mayoría de los casos los comandos son genéricos y con el propósito de cumplir con los estándares internacionales.

Hasta aquí, el proceso es relativamente fácil, pero... ¿Realmente es lo que necesitamos?. En la práctica, no todo queda ahí, fácil y sencillo. De manera real y práctica se requiere que exista una aplicación residente en memoria, que se encuentre presente siempre que requiramos los servicios de impresión, para que realice las funciones de control de dispositivo así como las traducciones necesarias de información a comandos de impresión.

Entonces, la situación ya no es tan sencilla, puesto que también se requiere de algún administrador de recursos de memoria para realizar los accesos y movimientos de código e información, y es ahí donde nace la necesidad de crear el denominado ***Controlador de Dispositivo de Impresión.***

Básicamente un Controlador de Impresión para un entorno operativo gráfico y multitarea como Windows 9X, "Millenium" Me y Windows 2000 es código binario ejecutable que cumple con las siguientes características genéricas:

A) Es el encargado de convertir la información del usuario que se desea imprimir en comandos que entienda el dispositivo impresor.

B) Es el encargado de suministrar las funciones hardware del dispositivo para que este funcione adecuadamente (se realiza mediante una comunicación bidireccional). Funciones como transportar la ó las cabezas de impresión (matriz de puntos e inyección de tinta), rodar el rodillo, alimentar la página, guardar ó liberar el buffer, seleccionar resolución, etc.

De manera tentativa, podemos definir a un dispositivo físico como un elemento hardware conectado a nivel periférico a una computadora. Por ser un elemento periférico, no forma parte de la CPU y, por lo tanto no es posible acceder al periférico de manera directa. Esto es en parte, por la gran variedad de fabricantes de dispositivos, quienes de alguna manera no manejan un estándar en cuanto a los comandos de comunicación y funcionales se refiere, es decir, estos comandos de comunicación varían de un fabricante de dispositivos a otro, aunque ya se han establecido algunos estándares en el IEEE en lo que se refiere a "Plug and Play".

Son estas diferencias y particularidades de cada dispositivo las que hacen necesaria la existencia de controladores de Dispositivos, los cuales básicamente funcionan a manera de interfaz entre el entorno Windows y el dispositivo en sí. Son los encargados de convertir las llamadas que el GDI realiza al dispositivo, para que el dispositivo hardware pueda leerlas, interpretarlas y ejecutarlas como comandos ó información. Cabe señalar que cualquier aplicación Windows, realiza las peticiones de Dispositivo, no directamente a este, sino al GDI y, es este quien se encarga de llamar al controlador de dispositivo y preguntarlo si puede ejecutar la petición de la aplicación; en caso afirmativo, el GDI le pide al controlador que ejecute la acción solicitada, mientras que en caso negativo, el GDI lo reporta como no posible a la aplicación.

Para Windows existen básicamente 3 tipos de controladores de impresión (Printer Driver):

1.- RasDD (raster Device Driver).

2.- PostScript Driver.
3.- Plotter Driver.

Los cuales abarcan a la mayoría de las familias de impresoras disponibles en el mercado actual. Las excepciones pueden tratarse de Impresoras que contienen aceleradores de dibujo que son controlados por secuencias de propietarias de comandos.

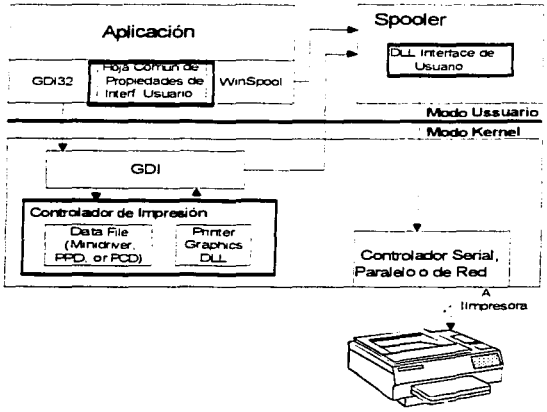
De manera independiente, cada uno de esos controladores tiene tablas de datos que les permiten soportar distintos clases de dispositivos, puesto que los controladores personalizan su operación para distintos modelos de impresoras, de manera que pueden abarcar mas con solo cambiar el archivo de descripción de datos, en vez de crear un nuevo controlador. Ese nuevo archivo de datos es conocido como el ***minidriver.***

Si se tiene algún dispositivo de impresión, el cual no sea soportado por Windows, es posible que se pueda utilizar alguno de los controladores antes mencionados, sin necesidad de escribir el controlador de nuevo, con tan sólo crear un ***minidriver***, el cual a su vez va a interactuar con alguno de los controladores antes mencionados.

Las diferencias básicas entre los distintos tipos de controladores se tratarán mas adelante, pero es conveniente anticipar la diferencia básica que los distingue. Un Controlador Raster habilita un Mapa de Bits GDI Estándar, permitiendo que el GDI administre y maneje la construcción de los puntos que conforman la imagen, mientras que un Controlador PostScript "escribe" a una superficie controlada por dispositivo, el cual debe instrumentar ciertos puntos de entrada (Entry Points) para la construcción de la imagen

Debido la independencia existente entre las aplicaciones y los dispositivos, el GDI accesa a las características de la impresora a través del controlador de dispositivo (Printer Driver), de manera que el Controlador es el intermediario que provee instrumentaciones específicas de dispositivo para las funciones de la impresora.

A continuación se muestra un diagrama del lugar que ocupa el Controlador de Impresión en la arquitectura de Impresión de Windows.



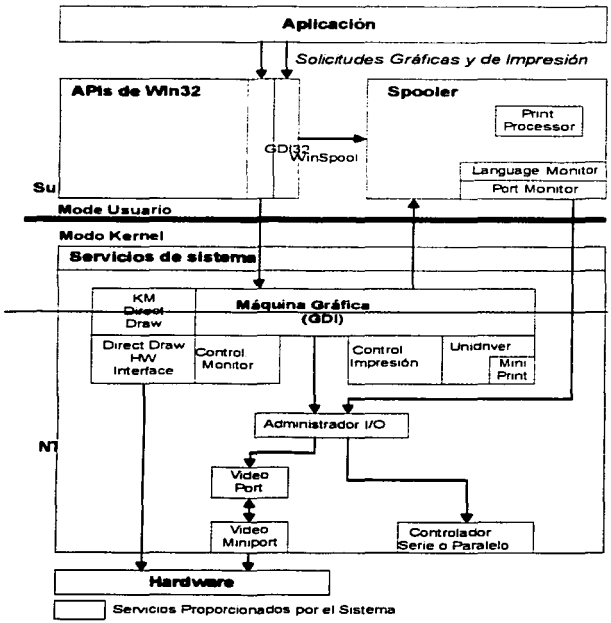
Como se puede observar, la única forma en que El Controlador de Impresión se pueda comunicar con la aplicación es a través de el GDI y, es por esa forma que el ***Spooler*** envía instrucciones de impresión a el dispositivo, esto significa que el Controlador de Impresión reacciona a las llamadas hechas por el GDI. De esta manera, si el Controlador determina que el GDI puede procesar adecuadamente una petición de impresión, este le regresa la petición para que el GDI la procese, y de esta manera el tamaño del controlador sea significativamente más pequeño, puesto que sólo se hace cargo de peticiones particulares a la impresora

Capitulo 1

Controladores de dispositivos de impresión en el entorno operativo Windows 2000.

1.1.a) Arquitectura Gráfica de Windows

A continuación se muestra de manera esquemática, como la información de impresión es tratada, desde las peticiones de la aplicación, pasando por el Subsistema de Win32 (User Mode) que es a donde la aplicación puede acceder, quien a su vez se "engancha" con el sistema de servicios NT Executive y, este con el Dispositivo Físico.



Como se puede observar en la gráfica anterior, la aplicación puede solamente accesar al subsistema llamado Win32 (Windows 32 bits), en donde se encuentran todas las funciones y rutinas API (Application Program Interface) de 32 bits, que la aplicación puede llamar, no sólo para la impresión, sino para las salidas por monitor, memoria, archivos, o para ejecutar acciones propias de la aplicación. Dentro de estas rutinas y funciones llamadas ***APIs***, se encuentra el GDI32, quien a su vez posee el grupo de llamadas al "Spooler de Impresión" (WinSpool). Este Subsistema Win32, recibe el nombre de ***User Mode***, puesto que es a lo que la aplicación puede acceder. De esta manera, se le permite al desarrollador de aplicaciones Windows, elaborar código independiente a los Dispositivos Hardware que se vayan a utilizar.

Por otro lado, El Subsistema Win32 realiza llamadas hacia la Máquina Gráfica (Graphic Engine), también conocida como GDI, estas llamadas son a través de sistema de servicios de los módulos ejecutables de NT (NT Executive). Es aquí, donde el GDI realiza el procesamiento de esas llamadas, en conjunción con los controladores gráficos, ya sea de impresión (Printer Driver), o de Monitor (Display Driver). Este Sistema GDI no puede ser reemplazado por ningún otro subsistema.

La forma en que el GDI se comunica con el Controlador Gráfico, es a través de un conjunto de funciones DDI (Device Driver Interface), las cuales para poder identificarse llevan antepuesto el prefijo "***Drv***", y la información es pasada mediante parámetros de entrada y salida en estas funciones. El Controlador debe soportar un mínimo de funciones "***Drv***", por las peticiones que el GDI realice a este, con las cuales podrá realizar algunas operaciones antes de responder al GDI.

El GDI puede realizar muchas operaciones de salidas gráficas, eliminando la necesidad para el controlador de soportar estas capacidades y de esta manera reducir el tamaño del controlador, lo que significa que el GDI, al igual que el controlador también exporta funciones que el controlador puede llamar. De esta manera, las rutinas soportadas por el GDI esta identificadas por el prefijo "***Eng***", mientras que las funciones que brindan acceso a las estructuras cuya responsabilidad corre a cargo del GDI, llevan nombres del tipo "XXXOBJ_Xxx".

1.1.b) Principales Operaciones Gráficas que Corren a Cargo del GDI

Operaciones de construcción Gráficas (Rendering Engine Operations). Para esto, el Controlador, primero debe construir y habilitar una superficie por cada representación lógica de dispositivo físico (PDEV), que llamaremos PDEV. Si el hardware del dispositivo físico es capaz de procesar mapas de bits de manera estándar, en el formato ***"Standard format bitmap"***, entonces puede dejar que el GDI se encargue del procesamiento y construcción de todas las imágenes.

Capacidades de Relleno en Degradado (GDI-Halftoning Capabilities). El relleno en degradado produce imágenes con calidad en color ó escala de grises, para dispositivos que no tienen la capacidad de construir estos por cuenta propia.

Mapas de bits administrador por GDI (GDI-managed Bitmaps). El GDI es capaz de procesar mapas de bits en todos los formatos DIB, de 1, 4, 8, 16, 24 y 32 bits por pulgada, así como también operaciones de dibujo de líneas, curvas, rellenos, y copiado de bloques, lo que propicia que el controlador se haga cargo del soporte de funciones especiales por parte del dispositivo impresor. Todo esto es posible si el dispositivo físico impresor lo permite, de cualquier otra manera, es responsabilidad del controlador el implementar las rutinas necesarias para el procesamiento y dibujo.

Líneas y Curvas de Mapa de Bits administradas por GDI (GDI-Managed bitmaps lines and curves). El GDI ofrece soporte para las líneas que no requieren tener puntos de finalización de tipo entero en coordenadas de dispositivo. La curva básica y fundamental en GDI es una curva Bezier. Todas las operaciones internas del GDI son manejadas mediante curvas del tipo Bezier, las cuales son soportadas por la mayoría de los dispositivos de impresión mas recientes. Para dispositivos que no las soporten, el GDI transforma las curvas a segmentos de línea antes de llamar al controlador para que los dibuje.

Atributos manejados por GDI: Los Pinceles. (Brushes). GDI también es capaz de manejar los atributos de una imagen, cuya denominación en el GDI es de Pinceles (Brushes). El controlador ejecuta los pinceles por conversión a sus formas internas, las cuales mantienen estados, redondeos, correlaciones, posiciones y estilos de línea.

1.1.c) Comunicación del GDI hacia el Controlador

Para fines de comunicación, el controlador exporta sólo dos funciones al GDI: ***"DrvEnableDriver"*** y ***"DrvDisableDriver"***, ya que cualquier otra llamada a función es a través de un arreglo de apuntadores que el GDI puede ver. Como es de esperarse, el GDI utiliza la primera rutina para iniciar el controlador, quien a su vez le envía de retorno una lista de funciones DDI soportadas por el controlador, de esta manera, el GDI se encargara de administrar las funciones que no soporte el controlador. La llamada a ***"DrvDisableDriver"*** se realiza cuando el controlador debe ser descargado del sistema. Mas adelante se discutirán aspectos relevantes acerca de las funciones DDI.

El GDI divide los servicios disponibles para el controlador ***en Objetos de Usuario y Rutinas de Servicio***.

Objetos de Usuario. Pueden ser definidos como ciertas partes (campos) de las estructuras que el GDI administra, pero que le deja acceso al controlador para que los utilice como Objetos de Usuario. Estos proveen una interfaz entre las estructuras internas que maneja el GDI y el controlador, ya que este último necesita acceder a la información de estas estructuras para llenar la información pertinente. Una vez hecho esto, el controlador retorna al GDI un apuntador al Objeto de Usuario para solicitar información ó servicios.

Los siguientes Objetos de Usuario están disponibles a un Controlador de Impresión:

1.	BRUSHOBJ	Define los objetos de pincel.
2.	CLIPOBJ	Accesa áreas de rectángulo.
3.	FLOATOBJ	Emula al controlador el punto flotante.
4.	FONTOBJ	Informa acerca de una fuente.
5.	PALOBJ	Contiene paletas de colores RGB.
6.	PATHOBJ	Especifica rutas que serán dibujadas.
7.	STROBJ	Especifica como líneas de texto son dibujadas.
8.	SURFOBJ	Identifica una superficie.
9.	XFORMOB	

Ya anteriormente hemos señalado que las rutinas que el GDI exporta llevan el prefijo "***Eng***", y son a estas rutinas a las que el controlador de impresión se enlaza de manera dinámica (en tiempo de ejecución) mediante el uso de la librería ***win32k.sys***.

1.2 Componentes Básicos de un Controlador de Impresión

Podemos definir básicamente tres componentes que integran la funcionalidad de un Controlador de Impresión en Windows, y son el DLL Controlador Gráfico de Impresión, el DLL Interfaz de Controlador de Impresión y, el Archivo de datos del Controlador de Impresión.

- El ***DLL Controlador Gráfico de Impresión*** implementa las funciones DDI necesarias para la construcción de salidas gráficas que no puedan ser correctamente manejadas y administradas por la máquina gráfica.
- El ***DLL Interfaz de Controlador de Impresora*** es el responsable de crear una interfaz que permite al usuario tener acceso a las propiedades físicas de la Impresora, así como las propiedades lógicas del documento. Es el responsable de interactuar con la Hoja de Usuario Común de Propiedades (Common Property Sheet User Interface) para proveer de nuevas características, en respuesta a una petición de usuario.
- El ***Archivo de Datos del Dispositivo*** de Impresión proporciona información especifica acerca de la impresión gráfica y componentes de la interfaz de usuario. El tipo del archivo, así como sus componentes. Cabe señalar que en las anteriores versiones al Windows NT y W2K los controladores de impresora se ejecutaban en modo usuario, mientras que las versiones actuales se ejecutan en modo kernel, lo que se refleja en una mejora al rendimiento del sistema

Con esto, definimos que los controladores de Impresión son específicos para un tipo particular de familias de dispositivos de impresión y, como se mencionó antes, existen tres tipos de controladores manejados por Windows.

TIPO DE CONTROLADOR	CONTROLADOR	DLL PRINTER UI	DATA FILE
RasDD	rasdd.dll	rasddui.dll	Minidriver
PostScript	pscript.dll	psui.dll	.PPD
Plotter	plotter.dll	plotui.dll	.PCD

RasDD (Raster Device Driver). Este controlador es conocido como el Controlador Universal ó "Unidriver", ya que soporta la impresión de gráficos en modo raster, que es la mayoría de los dispositivos de impresión. Como parte de este subsistema cada proveedor de impresoras suministra un minicontrolador ó minidriver que contiene un ***Archivo de Datos del Dispositivo***, que trabaja en conjunto con el minicontrolador ó minidriver para comunicarse con el Dispositivo de Impresión. Como características elementales podemos establecer que incluye el manejo de colores de 24 bits, fuentes escalables en modo real, manejo de fuentes de dispositivo, codificación para compresión run-length (RLE) e imágenes en formato TIFF 4.0, así como mecanismos para el manejo de mapas de bits más pequeños y eficientes. Dentro de estos mecanismos, se incluye la posibilidad de ignorar los espacios en blanco e incluir reglas (Las reglas de repetición en una página y las listas de marcas por medio de la repetición de un simple bit en el código fuente son mecanismos utilizados por las impresoras LaserJet de Hewlett-packard y sus compatibles).

PostScript. Como lo menciona la tabla anterior los archivos de descripción de Datos del Dispositivo son del tipo PPD (PostScript Compatible Adobe 4.2) que brinda soporte a características como la compresión de transferencia binaria nivel II, resolución y fuente de papel.

Plotter(Tipo Trazador Gráfico HP-GL/2)..

Estos archivos son instalados en la PC cuando se instala su correspondiente impresora, los cuales soportan a la mayoría de los dispositivos de impresión existentes en el mercado y, sólo es estrictamente necesario escribir un nuevo controlador cuando ningún controlador proporcionado por Windows sea compatible con el dispositivo de impresión a controlar.

1.2.a) DLL Controlador Gráfico de Impresión

Una aplicación, para imprimir un documento llama a una variedad de funciones del tipo Win32 y, muchas de estas funciones dependen de que los componentes del controlador gráfico ejecuten las acciones pertinentes a la construcción del gráfico y que son específicas al dispositivo que se este utilizando. Las funciones llamadas básicamente son para:

- Construcción de dibujos en gráficos y texto.
- Retorno de información especifica del controlador

- Señalización de comienzo y fin de documentos y páginas.
- Creación de objetos compatibles con el dispositivo como fuentes, pinceles y paletas.

Windows incluye un controlador del tipo Raster, denominado RasDD, el cual es capaz de realizar las funciones básicas de impresión gráfica y de texto, así como de avance de página en la mayoría de las impresoras de este tipo, lo que se traduce en que no es necesario codificar esta parte. En vez de ello, si se debe hacer un ***Minidriver*** para este dispositivo de impresión en particular, y para ello, deberá utilizar una herramienta que viene con el DDK (Device Driver Kit) cuyo nombre es el MDT (MiniDriver Development Tool). Este Minidriver contiene información específica de esta impresora una vez que este es creado.. Cabe hacer mención que este RasDD no soporta dispositivos que utilicen algún lenguaje similar al PostScript o al del Plotter.

1.2.b) DLL Interface de Controlador de Impresión

La información referente a parametrización del controlador de impresora, puede ser obtenida y modificada mediante el uso de una interfaz de usuario, que se provee a través del mismo Windows Ciertas propiedades del impresor pueden ser alteradas, como el puerto de conexión, la alimentación del papel, etc. Las propiedades que pueden ser modificadas son las del documento y las del impresor.

Una aplicación puede llamar a la hoja común de propiedades del dispositivo de impresión con el fin de modificar las propiedades del documento o del dispositivo. Destinado a administrar esto, tenemos un sistema incluido dentro del mismo Windows, llamado CPSUI, por sus siglas en ingles "Common Property Sheet User Interface" quien específicamente, es responsable de crear y destruir las hojas de propiedades de una aplicación, que esta pueda requerir. Cuando un usuario selecciona el elemento de menú "Propiedades de Impresora", el CPSUI muestra un cuadro de diálogo. La parte "Configuración de dispositivo", fue obtenida del Controlador de Impresión a través de una llamada a ***DrvDevicePropertySheets*** y ***DrvDocumentPropertySheets***.

1.2.c) Archivo de datos del dispositivo de impresión.

Como ya se ha mencionado antes, el archivo de datos contiene las características de la impresora, así como sus capacidades. El tipo de archivo de datos puede ser del tipo ***Minidriver (GPD)***, ***PPD*** o ***PCD***, según sea el tipo de impresora en cuestión.

Un ***Minidriver*** es un archivo de datos que el controlador de impresión del tipo Raster (RasDD) utiliza para obtener información de un grupo de estructuras de datos, conocidas como tabla de datos de impresora y, que contiene características como resolución, capacidades de color, tipos de fuentes, directivas de como crear primitivas de mapas de bits, y comandos propios para el control del dispositivo de impresión. Para crear un archivo de esta naturaleza, y por razones de comodidad y compatibilidad, se debe utilizar una herramienta incluida en el DDK para Windows, llamada MDT, quien automáticamente enlaza este archivo de descripción con la plantilla básica del controlador de tipo Raster (RasDD: Raster Device Driver), y a su vez el resultado es el DLL controlador de impresión.

Un archivo del tipo ***PPD***, es un archivo de datos que utiliza el controlador de tipo PostScript. La diferencia hacia con el Minidriver, es que este no es enlazado para obtener un DLL, sino que es un archivo del tipo ASCII que es leído por el controlador durante la llamada a ***DrvEnablePDEV*** que forma parte de la inicialización de la impresora. Cabe señalar que este tipo de controlador puede soportar cualquier dispositivo PostScript, con sólo cambiar el archivo PPD adecuado a este dispositivo.

Un archivo del tipo ***PCD***, es un archivo que utiliza un controlador del tipo Plotter, el cual debe ser creado en un principio como un archivo del tipo ASCII, pero que debe ser convertido a formato PCD con ayuda de la utilería ***plotgpc***, incluida en el DDK y cuyo proceso no será tratado aquí.

1.2.d) Interface entre aplicaciones de usuario e impresión

Haciendo un análisis de la impresión, ya no desde el punto de vista del controlador o del GDI, sino desde alguna aplicación o desde el programador de una aplicación x que requiera hacer llamadas a algún dispositivo de impresión, se tiene lo siguiente

Una de las características esenciales de las funciones de impresión de Windows es que soportan la independencia de dispositivo. Esto es, que en vez utilizar comandos para enviar la salida a una impresora especifica, la aplicación hace llamadas a funciones gráficas de alto nivel desde el GDI. Por ejemplo, para imprimir una imagen de mapa de bits, una aplicación llama a la función ***BitBlt***, proporcionando las coordenadas de la imagen. Esta imagen previamente ya debe tener un handle asignado, así como también debe existir un handle hacia el controlador del dispositivo de impresión. Estos handles reciben al nombre de **DC's** (Device Context). Esta llamada es después convertida a comandos de dispositivo por el controlador de impresora. Este controlador es visto por la aplicación como una librería de enlace dinámico ó **DLL** (Dynamic Link Library) que soporta las llamadas **DDI** (Device Driver Interface) de Windows. Un controlador de dispositivo genera comandos de dispositivo RAW cuando procesa las llamadas de funciones DDI hechas por la máquina gráfica. Los comandos son procesados por la impresora cuando esta imprime la imagen.

Una aplicación requiere un **DC(Device Context)** justo antes de poder comenzar el dibujo en el área cliente de una ventana. Necesita un DC especial de impresora antes de poder comenzar a enviar datos a la impresora. Un DC de impresora es similar a un DC de monitor, en el sentido de que es una estructura interna de datos que define un conjunto de objetos gráficos, así como sus atributos asociados, y especifica los modos gráficos que afectan la salida. Dentro de los objetos gráficos se incluye una Pluma, un Pincel y una Fuente (Pen, Brush y Font) para el dibujo de lineas, coloreado y rellenado, y para la salida de texto respectivamente.

Debido a que los DC de Monitor y de Impresora no son propiedad del componente de administración de la ventana, sus DC no pueden obtenerse mediante la llamada a la función ***GetDC***. En vez de ello la aplicación llama a ***CreateDC ó PrintDlg***. Cuando una aplicación llama a ***CreateDC***, debe proporcionar el nombre del controlador y el nombre del puerto. Estos datos se encuentran en los archivos de inicio de Windows INI, datos que pueden ser obtenidos de estos archivos mediante la llamada a la función "EnumPrinters". Cuando una aplicación llama a ***PrintDlg*** y especifica el valor **PD_RETURNDC** en el miembro "flags" de la estructura ***PRINTDLG***, Windows automáticamente retorna un "handle" identificando el DC de impresora seleccionado por el usuario.

A continuación mostramos un ejemplo de aplicación, donde el menú de archivo contiene 2 opciones. *Print* y *Print Setup*. Mediante la elección de alguna de

las dos el usuario puede configurar la impresora. Cuando el usuario elige ***Print Setup***, su correspondiente cuadro de diálogo aparece, donde se puede seleccionar la orientación del papel, tamaño, etc. Cuando se elige sólo ***Print***, se puede seleccionar el rango de páginas, calidad de impresión, número de copias, etc.

Como ya se había mencionado antes, ambos cuadros de diálogo son mostrados mediante la iniciación de los miembros de la estructura ***PRINTDLG*** y mediante la llamada a la función ***PrintDlg***. Esta última función puede ser utilizada para obtener un DC de impresora, especificando la bandera PD_RETURNDC en "flags". A continuación se muestra un fragmento de código fuente que muestra como iniciar los miembros de la estructura y mostrar el cuadro e diálogo.

```
/* Inicializa los miembros de PRINTDLG. */

pd.lStructSize = sizeof(PRINTDLG);
pd.hDevMode = (HANDLE) NULL;
pd.hDevNames = (HANDLE) NULL;
pd.Flags = PD_RETURNDC;
pd.hwndOwner = hwnd;
pd.hDC = (HDC) NULL;
pd.nFromPage = 1;
pd.nToPage = 1;
pd.nMinPage = 0;
pd.nMaxPage = 0;
pd.nCopies = 1;
pd.hInstance = (HANDLE) NULL;
pd.lCustData = 0L;
pd.lpfnPrintHook = (LPPRINTHOOKPROC) NULL;
pd.lpfnSetupHook = (LPSETUPHOOKPROC) NULL;
pd.lpPrintTemplateName = (LPSTR) NULL;
pd.lpSetupTemplateName = (LPSTR)  NULL;
pd.hPrintTemplate = (HANDLE) NULL;
pd.hSetupTemplate = (HANDLE) NULL;

/* Despliega el Cuadro de Diálogo PRINT. */

PrintDlg(&pd);
```

1.2.e) Funciones Escape

En adición y para soportar los Escapes, Win32 proporciona una nueva función Escape Extendida llamada ***ExtEscape***. Esta función permite a las aplicaciones accesar las capacidades de un dispositivo en particular no disponibles directamente a través del GDI.

Una vez que la aplicación realiza las iniciaciones de las variables pertinentes, se registra la función ***AbortProc*** y se despliega el cuadro de diálogo no modal "Cancelar". Para comenzar la tarea de impresión se hace una llamada a la función ***StartDoc***. Después de que una aplicación comienza la tarea de impresión, puede definir páginas individuales en un documento mediante la llamada a las ***StartPage*** y ***EndPage***. Después de que la aplicación ha definido la última página, esta cierra el documento y finaliza la tarea de impresión con la llamada a la función ***EndDoc***. El siguiente ejemplo muestra el código requerido para imprimir una cadena de texto y una imagen de mapa de bits. La cadena de texto centrada al margen superior, identifica la ruta y nombre de archivo que contiene la imagen de mapa de bits. La imagen de mapa de bits, centrada vertical y horizontalmente en la página es dibujada a manera que la misma proporción usada para dibujar la imagen en la ventana de la aplicación sea mantenida.

Ejemplo:

```
/**Inicializa los miembros de la estructura DOCINFO*/

di.cbSize = sizeof(DOCINFO);
di.lpszDocName = "Bitmap Printing Test";
di.lpszOutput = (LPTSTR) NULL;
di.lpszDataType = (LPTSTR) NULL;
di.fwType = 0;

/* Comienza la tarea de impresión con la llamada a la función StartDoc */

nError = StartDoc(pd.hDC, &di);
if (nError == SP_ERROR) {
        errhandler("StartDoc", hwnd);
goto Error;
}

/* Informa a el controlador que la aplicación esta por comenzar el envio de datos */

nError = StartPage(pd.hDC);
if (nError <= 0) {
        errhandler("StartPage", hwnd);
goto Error;
}

/* Obtiene información del número de pixeles por pulgada lógica en las direcciones horizontal y
vertical para el monitor para el mapa de bits que es creado */
```

```
fLogPelsX1 = (float) GetDeviceCaps(pd.hDC, LOGPIXELSX);
fLogPelsY1 = (float) GetDeviceCaps(pd.hDC, LOGPIXELSY);

/* Obtiene información del número de pixeles por pulgada lógica en las direcciones horizontal y
vertical para el impresor en le mapa de bits que será creado */

fLogPelsX2 = (float) GetDeviceCaps(pd.hDC, LOGPIXELSX);
fLogPelsY2 = (float) GetDeviceCaps(pd.hDC, LOGPIXELSY);

/* Determina los factores de escalamiento requeridos para imprimir el mapa de bits y retener sus
proporciones originales */

if (fLogPelsX1 > fLogPelsX2)
        fScaleX = (fLogPelsX1 / fLogPelsX2);
else
    fScaleX = (fLogPelsX2 / fLogPelsX1);

if (fLogPelsY1 > fLogPelsY2)
        fScaleY = (fLogPelsY1 / fLogPelsY2);
else
    fScaleY = (fLogPelsY2 / fLogPelsY1);

/* Calcula la coordenada de la esquina superior izquierda del mapa de bits centrado */

cWidthPels = GetDeviceCaps(pd.hDC, HORZRES);
xLeft = ((cWidthPels / 2)-((int)(((float) bmih.biWidth)*fScaleX))/2);
cHeightPels = GetDeviceCaps(pd.hDC, VERTRES);
yTop = ((cHeightPels / 2)-((int)(((float) bmih.biHeight)*fScaleY))/2);

/* Crea un DC en memoria que es compatible con la impresora y selecciona el mapa de bits hacia
este DC */

hdcMem = CreateCompatibleDC(pd.hDC);

if (!SelectObject(hdcMem, hbm))
        errhandler("SelectObject Failed", hwnd);

/* Utiliza la función StretchBlt para escalarel mapa de bits y mantener sus proporciones originales
(esto es, si el mapa de bits aparece cuadrado en el área cliente de la aplicación, debe aparecer
cuadrado en la página de la impresora). */

if (!StretchBlt(pd.hDC, xLeft, yTop,(int) ((float) bmih.biWidth * fScaleX),(int) ((float)
bmih.biHeight * fScaleY), hdcMem, 0, 0, bmih.biWidth, bmih.biHeight, SRCCOPY))
        errhandler("StretchBlt Failed", hwnd);
```

```
/* Elimina el DC de memoria */
DeleteDC(hdcMem);

/* Obtiene el ancho de la cadena que especifica la ruta absoluta y el nombre de archivo para el
archivo el mapa de bits */

GetTextExtentPoint32(pd.hDC, ofn.lpstrFile, ofn.nFileExtension + 3, &szMetric);

/* Calcula el punto de comienzo para las operaciones de salida de texto. La cadena será centrada
horizontalmente y posicionada tres lineas abajo, desde el tope de la página */

xLeft = ((cWidthPels / 2) - (szMetric.cx / 2));
yTop = (szMetric.cy * 3);

/*Imprime la ruta y nombre de archivo para el mapa de bits centrado al tope de la página*/

TextOut(pd.hDC, xLeft, yTop, ofn.lpstrFile, ofn.nFileExtension + 3);

/* Determina si el usuario ha presionado el botón CANCEL en el cuadro de diálogo
AbortPrintJob. Si el boton ha sido presionado llama a la función AbortDoc */

nError = EndPage(pd.hDC);

if (nError <= 0) {
        errhandler("EndPage", hwnd);
goto Error;
}

/* Informa al controlador que el documento ha finalizado */

nError = EndDoc(pd.hDC);
if (nError <= 0)
        errhandler("EndDoc", hwnd);

Error:
/* Habilita la ventana de la aplicación */
EnableWindow(hwnd, TRUE);

/* Remueve el cuadro de diálogo AbortPrintJob */
DestroyWindow(hdlgCancel);

/* Elimina el DC de impresora */
DeleteDC(pd.hDC);
```

Debido a que los pixeles en la pantalla tipicamente tienen diferentes dimensiones que los dots de una impresora, es necesario escalar las imagenes de mapa de bits para obtener el efecto WYSIWYG. Este se obtiene mediante vectores de escalamiento horizontal y vertical, que son aplicados a los valores de ancho y alto de la función ***StretchBlt***. En la aplicación de ejemplo, los factores de escalamiento fueron obtenidos del contador de pixel lógico vertical y horizontal para ambos dispositivo. Una vez que estos factores, fueron utilizados para ajustar el ancho y alto del mapa de bits.

Para centrar el mapa de bits en la página, la aplicación debe calcular el ancho y alto del mapa de bits escalado. Estos valores fueron divididos por dos y entonces restados de la mitad del ancho y alto de la página. El resultado define las coordenadas de la esquina superior izquierda del mapa de bits.

Para centrar el texto al tope superior de la página, la aplicación llama a la función ***GetTextExtentPoint32*** para obtener el ancho y alto de la cadena especificando la ruta y nombre de archivos. Una vez que esos valores fueron obtenidos, la aplicación usó la altura para posicionar la cadena tres líneas bajo la página y el ancho para posicionar la cadena horizontalmente centrada en la página.

Las funciones posibles que pueden ser utilizadas para impresión, así como de manejo del "spooler de impresión" son las siguientes:

Funciones de Impresión

AbortDoc
DeviceCapabilities
EndDoc
EndPage
Escape
ExtEscape
SetAbortProc
StartDoc
StartPage

Funciones de acceso al Spooler de Impresión

AbortPrinter
AbortProc
AddForm
AddJob

AddMonitor
AddPort
AddPrinter
AddPrinterConnection
AddPrinterDriver
AddPrintProcessor
AddPrintProvidor
AdvancedDocumentProperties
ClosePrinter
ConfigurePort
ConnectToPrinterDlg
DeleteForm
DeleteMonitor
DeletePort
DeletePrinter
DeletePrinterConnection
DeletePrinterData
DeletePrinterDriver
DeletePrintProcessor
DeletePrintProvidor
DocumentProperties
EndDocPrinter
EndPagePrinter
EnumForms
EnumJobs
EnumMonitors
EnumPorts
EnumPrinterData
EnumPrinterDrivers
EnumPrinters
EnumPrintProcessorDataTypes
EnumPrintProcessors
FindClosePrinterChangeNotification
FindFirstPrinterChangeNotification
FindNextPrinterChangeNotification
FreePrinterNotifyInfo
GetForm
GetJob
GetPrinter
GetPrinterData
GetPrinterDriver
GetPrinterDriverDirectory
GetPrintProcessorDirectory
OpenPrinter
PrinterMessageBox

PrinterProperties
ReadPrinter
ResetPrinter
ScheduleJob
SetForm
SetJob
SetPort
SetPrinter
SetPrinterData
StartDocPrinter
StartPagePrinter
WaitForPrinterChange
WritePrinter

LAS SIGUIENTES ESTRUCTURAS SON USADAS CON EL SPOOLER DE IMPRESION

ADDJOB_INFO_1
DATATYPES_INFO_1
DEVMODE
DOC_INFO_1
DOC_INFO_2
DOCINFO
DRIVER_INFO_1
DRIVER_INFO_2
DRIVER_INFO_3
FORM_INFO_1
JOB_INFO_1
JOB_INFO_2
JOB_INFO_3
MONITOR_INFO_1
MONITOR_INFO_2
PORT_INFO_1
PORT_INFO_2
PORT_INFO_3
PRINTER_DEFAULTS
PRINTER_INFO_1
PRINTER_INFO_2
PRINTER_INFO_3
PRINTER_INFO_4
PRINTER_INFO_5
PRINTER_INFO_6
PRINTER_NOTIFY_OPTIONS
PRINTER_NOTIFY_OPTIONS_TYPE
PRINTER_NOTIFY_INFO
PRINTER_NOTIFY_INFO_DATA
PRINTPROCESSOR_INFO_1
PROVIDOR_INFO_1

Capitulo 2

Funcionamiento y control en el proceso de impresión

2.1 Método y Formato de impresión de Windows 2000

En este capitulo trataremos de realizar un breve análisis acerca del procedimiento de Impresión de un documento bajo el esquema de que Windows 2000 es el servidor de Impresión y la solicitud de impresión es realizada desde el mismo equipo o desde algún otro equipo remoto Windows 2000 ó Windows Me.

Primero. El usuario de alguna computadora cliente elige imprimir un documento. Esta petición es solicitada al Interfaz de Dispositivo Gráfico ó Graphic Device Interface (GDI). El GDI llama a su vez al Controlador de Impresión asociado al Dispositivo de Impresión. Utilizando la Información del Controlador, el GDI traduce la tarea de impresión al lenguaje del dispositivo de Impresión e invoca al lado cliente del administrador de la cola de Impresión (Winspool.drv).

Segundo. Si el cliente tiene en operación Windows, el lado cliente del administrador de la cola de impresión (***Winspool.drv***), llama mediante una Llamada a Procedimiento Remoto ó Remote Procedure Call (RPC) a la versión servidor del administrador de la cola de impresión (***Spoolss.exe***), quien a su vez llama a la API del encaminador (***Spoolss.dll***). Este encaminador sondea a los proveedores de impresión remota existentes (***Win32spl.dll***), quienes a su vez hacen una RPC a ***Spoolss.exe*** en el servidor de Impresión, quien recibe la tarea de impresión vía red.

Si el cliente tiene en operación otro sistema operativo distinto a Windows o si el cliente creó el puerto local de impresora y reedirigió la impresión al servidor de red (\\servidor\impresora), entonces la tarea de impresión va directo al NETBios de la computadora cliente, que utilizando el administrador de protocolos envía la tarea vía red al administrador de protocolos del servidor de impresión quien la entrega a los servicios de Windows del servidor de impresión.

Tercero. En el servidor de impresión las tareas de los clientes Windows son del tipo Enhanced MetaFiles (EMF), mientras que las tareas de los clientes no Windows asignan diferentes formatos de datos. Estos formatos de datos le dicen al

administrador de la cola de impresión si se debe modificar la tarea de impresión y de que forma. Estos tipos de formatos de datos son asignados a las tareas de impresión por algunos servicios de impresión. Algunos servicios de impresión dejan los formatos de datos en blanco los cuales adoptan el tipo predeterminado de datos, según la impresora creada. Por ejemplo, si la impresora es PCL, el tipo de datos es EMF, mientras que si es Adobe PostScript, el tipo de datos es RAW.

Cuarto. El encaminador envía la tarea de impresión al proveedor de impresión local para ser encolado. A su vez, el proveedor de impresión sondea a los procesadores de impresión para ver cual puede reconocer y tratar el formato de los datos.

Quinto. La tarea es leída por el Monitor de Impresión la cual puede realizar alguna de las siguientes acciones:

a) Si el Dispositivo de impresión es bidireccional, la tarea es traducida al lenguaje del monitor, quien dialoga con la impresora para enviar la tarea al puerto del monitor.
b) Si el Dispositivo no es bidireccional, la tarea es enviada al puerto del monitor, quien envía la tarea al dispositivo de impresión.

Sexto. El dispositivo de impresión recibe la tarea de impresión y cumple su función. IMPRIME...

Como se podrá observar, hemos hablado de algunos términos un tanto técnicos, como es el caso de tarea de impresión. Es necesario hacer un poco de énfasis en el término y explicar el concepto con mayor claridad. Las tareas de Impresión Son Código fuente que contiene datos y ordenes para procesar estos datos. La aplicación cliente crea las tareas de impresión con los datos, gráficos, mapas de bits, y le anexa información del controlador, con lo que se crea el código fuente de un documento, que es conocido como la tarea de impresión.

2.1.a) Formato EMF (Enhanced Metafile)

Como ya se ha mencionado antes, las tareas de impresión para los clientes de Windows, son Metarchivos mejorados ó Enhanced Metafiles (EMF), lo que significa que antes de ser enviadas a la cola de impresión, realiza las traducciones necesarias. Una vez creado el archivo EMF el control de la computadora cliente es devuelto al usuario, mientras que el EMF en cuestión se interpreta en otro THREAD en segundo plano, que a su vez forma parte del administrador de la cola de impresión. Estos archivos son mucho más compactos y compatibles, puesto que pueden ser impresos en cualquier dispositivo de impresión.

Los archivos de cola de impresión EMF son utilizados para reducir enormemente la duración del tiempo entre el inicio de una petición de impresión de una aplicación y cuando el control es retornado a la aplicación por el sistema operativo. Esto se realiza mediante las llamadas a las funciones GDI que producen el objeto gráfico de la aplicación a la impresora especificada, dentro de un archivo con formato EMF, llamado un archivo de cola de impresión (Print Spool File). El archivo de la cola de impresión puede ser construido rápidamente mediante esta forma y retornar el control a la aplicación. Después, utilizando el proceso de tareas en ***Background***, el spooler de Impresión de Windows, desempeña la tarea que más tiempo consume de las llamadas a la ejecución del GDI. Una segunda ventaja del uso de archivos EMF es la independencia de dispositivo de los archivos EMF.

El Tamaño de un archivo EMF es un arreglo de registros de tamaño variable que codifican las llamadas a GDI necesarias para reproducir la figura ó imagen, cuando el archivo EMF sea reproducido. Los archivos de impresión EMF codifican información gráfica, en la medida de mantener la independencia de dispositivo. En Windows, los archivos EMF son encolados localmente y desencolados (despooled) por el servidor. Un archivo EMF es creado por cada tarea de impresión (print job). El GDI coloca información de fuentes en el archivo EMF si el servidor noposee esta información de fuentes.

2.1.b) Formato RAW

A diferencia del formato anterior, este tipo de datos le indica al administrador de la cola de impresión, que no realice ningún cambio en la tarea de impresión.

Este formato indica que la tarea de impresión, ya ha sido completamente construida por el GDI/DDI y no necesita mayor procesamiento. Los flujos de datos RAW pueden ser impresos directamente o pueden ser colocados en archivos de cola (spool files).

La mayoría de los clientes de impresión envían tareas RAW, que son dependientes del dispositivo, es decir, que el formato es destinado para un tipo de dispositivo en particular y no para algún otro.

2.1.c) Formato Texto

Este formato le comunica al administrador de la cola de impresión que la tarea de impresión esta formada por texto ANSI. El administrador de la cola, usa la información del controlador de dispositivo para crear la tarea de impresión, que contenga las características, formatos y orientaciones que incorpora el dispositivo de impresión de fabrica y que se encuentran disponibles en el cuadro de diálogo ***Propiedades del Documento***. Cuando una aplicación crea una tarea de impresión con un conjunto de caracteres ANSI diferente al que utiliza el servidor de impresión, algunos caracteres pueden ser impresos de manera incorrecta. La mayoría de los conjuntos de caracteres son idénticos para los valores entre 0 y 127, de manera que el problema sólo se da en los caracteres extendidos que van desde 128 a 255.

2.1.d) Formato PostScript

Este tipo de formato pertenece a los clientes Macintosh que utilizan el formato PostScript por defecto y, el dispositivo de impresión no es un dispositivo PostScript. Para lo cual, el administrador de la cola de impresión interpreta el código PostScript y crea un mapa de bits de la página, para que el GDI a través del Controlador de Impresión traduzcan este mapa de bits a lenguaje entendible para el Dispositivo de Impresión.

2.2 Componentes del Subsistema de Impresión de Windows

2.2.a) Spooler de Impresión

Antes de tratar los aspectos característicos del Procesador y el Monitor de Impresión, debemos señalar algunos aspectos acerca del Spooler de impresión. El Spooler de Impresión de Windows 2000 es un componente del subsistema de impresión que habilita tareas de impresión para ser encoladas hacia impresoras locales o remotas. Se le llama subsistema de impresión puesto que engloba varios componentes en sí, entre los que esta el spooler de Impresión.

Entre las responsabilidades del spooler están:

a) Encuentra la localización del el controlador de impresión correcto y cargarlo.
b) Aceptar cadenas de datos preparadas por el GDI o por el Controlador de Impresión para ser enviadas a una impresora en particular.
c) Asegurarse que las instrucciones del controlador van a la primera impresora física disponible en una cola lógica de impresión. El spooler soporta impresión por red.
d) Encolar los datos a un archivo si la impresora no está disponible y, si es necesario convierte las llamadas a funciones de alto nivel a formato EMF antes de bajar la tarea de impresión a disco.
e) Libera datos a la impresora para impresión, ya sea directamente o por retroalimentar archivos encolados previamente.
f) Administra las bases de datos de impresora, como son las bases de datos de formas, los ambientes dinámicos de impresoras, controladores de impresión, monitores, proveedores de impresión, puertos y tareas de impresión.
g) Mantener los registros desde los cuales deriva cuantos de estos componentes están en el subsistema de impresión en algún momento y cual es su nombre único.

El spooler de impresión comienza a trabajar cuando se inicia el sistema, y deja de funcionar hasta que el sistema operativo es dado de baja (apagado ó descargado).

2.2.b) Componentes del Spooler de Impresión

Servidor de Procesos de Spooler (Spooler Server Process (SSP)): Da entrada al subsistema spooler para peticiones locales y remotas. El archivo ejecutable por defecto es ***spoolss.dll*** y, por ser proveído por Microsoft, no puede ser substituido.

Enrutador de Peticiones de Impresión (Print Request Router (PRR)): Enruta las peticiones de impresión del flujo de datos al proveedor de impresión. El archivo ejecutable por defecto es ***spolss.dll*** y, por ser proveído por Microsoft, no puede ser substituido.

Proveedor de Impresión Local (Local Print Provider): Es el responsable de la reproducción ó ejecución de las tareas de impresión encoladas (spooled), también es responsable de desencolar las tareas de impresión (despooling), y de administrar la cola de impresión. El archivo ejecutable por defecto es ***localspl.dll*** y, por ser proveído por Microsoft, no puede ser substituido.

Proveedor de Impresión Remota (Network Print Provider): es el responsable de trasladar las peticiones de impresión desde las llamadas a funciones de impresión Win32 a los formatos específicos de transmisión de la red en cuestión. El archivo ejecutable por defecto es ***win32spl.dll*** y, aunque si puede ser desarrollado y substituido por algún otro, típicamente no es necesario este desarrollo.

Procesadores de Impresión (Print Processors): Prácticamente realizan el proceso de desencolado ó despooling y que como ya se ha mencionado antes, maneja tres tipos de formatos que son RAW, EMF y TEXTO. El Procesador de Impresión y el Controlador de Impresora trabajan de manera conjunta, de manera que el proveedor del Dispositivo de Impresión puede desarrollar algún otro procesador de impresión. El archivo ejecutable por defecto es ***winprint.dll***.

Monitores de Puerto (Port Monitors): Es el responsable de la comunicación entre el spooler y el dispositivo de impresión. Típicamente son comunicaciones basadas en controladores de E/S (Entrada y Salida), pero también se comunica con algunas otras interfaces como Sockets Windows, SCSI, etc. El archivo ejecutable por defecto es ***localmon.dll***. Para los dispositivos de impresión que utilizan puertos de E/S (Seriales ó Paralelos) pueden utilizar sin problemas este archivo. Para el caso de dispositivos que manejen otros puertos como Ethernet o SCSI es necesario el desarrollo de un nuevo Monitor de Puerto.

Monitores de Lenguaje (Language Monitors): Le permite al spooler configurar y monitorear el estado de una impresora bidireccional. Sólo existe un archivo por defecto para las impresoras bidireccionales que entiendan el lenguaje PJL. Para los demás dispositivos no existe ningún archivo por defecto, pero si el proveedor del dispositivo de impresión lo desea puede desarrollar uno para su dispositivo en particular.

2.2.c) Procesador y monitor de impresión

Una vez que el proveedor de impresión determina que la tarea puede ser impresa, este llama al procesador de impresión, el cual es un DLL que interpreta los formatos de datos de la tarea de impresión. Trabaja en conjunción con el controlador de impresión para desencolar (***despooling***) los datos durante la ejecución del archivo encolado. Después el Procesador de Impresión retorna los datos procesados y convertidos al "spooler", quien los pasa al Monitor de Impresión. Los archivos encargados para ello son ***winprint.dll*** y ***sfmpsprt.dll*** para Windows y Macintosh respectivamente.

Las rutinas específicas para el diseño de un nuevo Procesador de Impresión, pueden ser encontradas en el archivo de cabecera "winsplp.h", donde se encuentran los prototipos de las funciones y estructuras que el Procesador de Impresión debe soportar. Entre las funciones que deben ser exportadas, tenemos:

EnumPrintProcessorDataTypes: Enumera los formatos de datos soportados por el Procesador de Impresión.
OpenPrintProcesor: Abre el Procesador de Impresión para imprimir.
PrintDocumentOnPrintProcessor: Imprime un documento en el Procesador de impresión.
ControlPrintProcessor: Proporciona el control sobre la impresión de un documento.
ClosePrintProcessor: Cierra el procesador de impresión.

Windows NT soporta dos tipos de monitores de impresión: Los monitores de puerto y los monitores de lenguaje.

2.2.d) Monitores de puerto y de lenguaje

Como ya se ha mencionado, este es el responsable del canal de comunicación entre el spooler y el dispositivo de impresión. Para habilitar la impresión a un dispositivo físico, Windows NT requiere que la impresora sea conectada a uno de los puertos que son controlador por los monitores de puertos, siendo el puerto el destino que seguirán los datos enviados a la impresora. Windows proporciona los siguientes cinco monitores de puertos:

a) ***Localmon.dll***. Monitor local que soporta los puertos seriales y paralelos bidireccionales

b) ***Sfmmon.dll***. Monitor para Macintosh que soporta Tarjeta de red con protocolo Apple Talk.

c) ***Decpsmon.dll***. Monitor para dispositivos DEC que soporta tarjeta de red con protocolo DECnet o TCP/IP.

d) ***Hpmon.dll***. Monitor para dispositivos HP JetDirect DLC que soporta tarjeta de red DLC

e) ***Lprmon.dll***. Monitor por el cual un dispositivo LPD es accesado usando Tarjeta de red con protocolo TCP/IP.

Un monitor de lenguaje permite al spooler configurar y monitorear el estado de una impresora bidireccional. De esta manera el monitor de lenguaje puede solicitar la configuración y estado de la impresora, así como también la impresora puede enviar información no solicitada al spooler cuando cierto evento ocurre en la impresora.

Los monitores de lenguaje se asocian con los controladores de impresión cuando la función tipo Win32 cuyo nombre es ***AddPrinterDriver*** es llamada con el nombre del monitor de lenguaje especificado en el campo ***pMonitorname*** de la estructura ***DRIVER_INFO_3***. Este parámetro es especificado en el archivo de instalación ***INF*** usando la palabra clave ***LanguageMonitor***. Existe un archivo de información de instalación por defecto llamado ***ntprint.inf*** que los proveedores (OEMs) pueden proporcionar en el instalador.

2.3) Archivo de referencia en formato OemSetup.Inf

En el formato INF de los archivos de instalación para Windows 2000 y Windows Me, todo el procedimiento de instalación está incluido en el programa de instalación, de esta manera, el archivo INF actúa entonces como un recurso, conteniendo el formato e información de archivos.

Un archivo INF es construido a manera de secciones con un nombre. Para ser utilizadas por el instalador del sistema operativo, cada sección debe contener uno ó más elementos. Existen aproximadamente 20 tipos de secciones que pueden ser usadas por un archivo INF. Cada tipo de sección tiene un propósito en particular; por ejemplo, para instalar un servicio y agregar entradas al registro, el archivo INF debe seguir las siguientes reglas generales.

- La sección debe comenzar con el nombre de esta, encerrado en paréntesis cuadrados.
- Los valores pueden ser expresados como cadenas reemplazables usando la forma %strkey%. Para utilizar un carácter % en la cadena, debe usarse %%. La forma strkey debe ser definida en la sección ***STRINGS*** del archivo INF.
- Cada archivo INF debe contener una sección VERSION identificando como un archivo compatible Windows NT/2000 o Windows 9X/Me.

Tres secciones que forman parte de un archivo INF se muestran a continuación.

La sección VERSION contiene cinco elementos, la sección DSTINATIONDIRS contiene un elemento y la sección CONTROLFLAGS contiene tres elementos. El resto de las posibles secciones son:

a) Add Registry
b) ClassInstall32
c) ControlFlags
d) Copy Files
e) Delete Files
f) DestinationDirs
g) Device
h) EventLog Install
i) HW
j) Ini File to Registry
k) Install
l) Manufacturer
M) Service Install
n) Services
o) SourceDisksFiles

p) SourceDisksNames
q) Strings
r) Version

2.3.a) Sección AddRegistry

La Sección ***AddRegistry*** adiciona subclaves o nombres al registro, opcionalmente instala valores. Su formato es:

```
[add-registry-section]
reg-root-string, [subkey], [value-name], [flags], [value]
[reg-root-string, [subkey], [value-name], [flags], [value]]
.
.
.
```

reg-root-string

donde ***reg-root-string*** puede ser alguno de los siguientes valores:

HKCR que significa **HKEY_CLASSES_ROOT**.
HKCU que significa **HKEY_CURRENT_USER**.
HKLM que significa **HKEY_LOCAL_MACHINE**.
HKU que significa **HKEY_USERS**.
HKR relacionada a la clave pasada hacia la sección **SetupInstallFromInfSection**.

Subkey

Este valor es opcional. Identifica la subclave a establecer. Tiene la forma clave1, clave2, clave3...

Value-name

Opcional. Identifica el valor nombre para la subclave. Para las cadenas, sí el valor de la izquierda esta vacío, se establece entonces un valor por defecto.

Flag

Opcional. Establece el valor del tipo de datos, así como la el elemento de acción "AddReg". El valor de la bandera es un mapa de bits donde la palabra (WORD) de menor significado contiene banderas el valor del tipo de datos y el elemento de acción "AddReg". La palabra de mayor significado contiene valores que identifican el tipo de datos de registro:

FLG_ADDREG_BINVALUETYPE
(0x00000001)
El valor significa datos tipo "raw".
FLG_ADDREG_NOCLOBBER
(0x00000002)

El valor significa no escribir sobre la clave de registro si esta ya existe
FLG_ADDREG_DELVAL
(0x00000004)
Eliminar el valor del registro.
FLG_ADDREG_APPEND
(0x00000008)
Anexar un valor al valor existente. Esta bandera sólo es soportada por valores REG_MULTI_SZ .
FLG_ADDREG_TYPE_MASK
(0xFFFF0000 | FLG_ADDREG_BINVALUETYPE)
Mascara (Mask).
FLG_ADDREG_TYPE_SZ (0x00000000)
Tipo de datos de Registro REG_SZ.
FLG_ADDREG_TYPE_MULTI_SZ
(0x00010000)
Tipo de datos de Registro REG_MULTI_SZ.
FLG_ADDREG_TYPE_EXPAND_SZ
(0x00020000)
Tipo de datos de Registro REG_EXPAND_SZ.
FLG_ADDREG_TYPE_BINARY
(0x00000000 | FLG_ADDREG_BINVALUETYPE)
Tipo de datos de Registro REG_BINARY.
FLG_ADDREG_TYPE_DWORD
(0x00010000 | FLG_ADDREG_BINVALUETYPE)
Tipo de datos de Registro REG_DWORD.
FLG_ADDREG_TYPE_NONE
(0x00020000 | FLG_ADDREG_BINVALUETYPE)
Tipo de datos de Registro REG_NONE.

Para representar un número con un tipo de datos distinto al predefinido por REG_*, el desarrollador puede especificar el tipo número en la bandera del la palabra mas alta y especificar el tipo binario en la palabra mas baja. Es posible mostrar introducir los datos binarios un byte por campo. Por ejemplo: para colocar 16 bits de datos con el nuevo tipo de datos 0x38, es necesario tener un elemento AddReg como:

```
HKR,,MYValue,0x00380001,0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
```

Esta técnica puede ser utilizada con cualquier tipo e dato excepto REG_EXPAND_SZ, REG_MULTI_SZ, REG_NONE y REG_SZ.

Value

Opcional. El valor puede ser una cadena, o un número en notación hexadecimal. Al menos dos campos son requeridos, sin embargo, uno puede ser nulo. Para utilizar esta forma se utilizaran al menos una coma. En el siguiente ejemplo, la sección Add Registry tiene dos elementos que se llaman:

Sermouse_EventLog_AddReg, adicionan dos nombre al registro EvenMessageFile y TypeSupported, además de establecer sus respectivos valores.

```
HKR,,EventMessageFile,0x00020000,"%%SystemRoot%%\System32\IoLogMsg.dll;%%Sys
temRoot%%\System32\drivers\sermouse.sys"
HKR,,TypesSupported,0x00010001,7
```

2.3.b) Sección ClassInstall32

La sección ***ClassInstall32***, instala una nueva clase para un dispositivo en la sección de clases del registro. El formato es el siguiente:

```
[ClassInstall32]
AddReg=add-registry-section[,add-registry-section]...
Copyfiles=file-list-section[,file-list-section]...
Delfiles=file-list section[,file-list-section]...
DelReg=del-registry-section[,del-registry-section]...
Renfiles=file-list-section[,file-list-section]...
UpdateInis=update-ini-section[,update-ini-section]...
UpdateIniFields=update-inifields-section[,update-inifields-section]...
```

No todos los elementos son necesarios o requeridos. Típicamente esta sección utilizara el elemento "AddReg" para agregar una descripción de clase y un icono de clase al registro si la clase asociada es ***Unknown***. Cada dispositivo instalado en Windows tiene una clase asociada con él en forma paralela, si la clase es desconocida. Cada dispositivo instalado tiene también un instalador asociado con él.

El instalador procesa la sección ***ClassInstall32*** si uno de los dispositivos definidos en un archivo INF será instalado y su clase no ha sido construida en Windows. Los siguientes nombres de clases de hardware están ya construidos en Windows:

Display
Keyboard
Modem
Mouse
Ports
Printer
SCSI Adapter
Tape Drive
Unknown

2.3.c) Sección ControlFlags

La sección ***ControlFlags*** controla como un dispositivo es manejado por el instalador del dispositivo. El nuevo "Wizard" de dispositivo construye su propia lista de dispositivos instalables mediante una búsqueda de archivos INF. El instalador de dispositivos extrae información de dispositivos de archivos INF y la despliega.

```
[ControlFlags]
ExcludeFromSelect=device-id[device-id]...
```

El elemento "ExcludeFromSelect" lista el ID de los dispositivos que serán excluidos de la lista del instalador de dispositivos. Si se utiliza un asterisco en lugar de **"device-id"**, el valor **"ExcludeFronSelect"** es establecido para todos los dispositivos listados en el archivo INF.

2.3.d) Sección CopyFiles

La Sección ***CopyFiles*** lista los archivos a ser copiados desde el disco origen, al directorio destino. Los directorios asociados con cada archivo están definidos en otras secciones. El nombre de la sección ***file-list-section*** debe aparecer en el elemento ***CopyFiles*** de la sección ***Install***. El desarrollador puede copiar un archivo único con el elemento "CopyFiles" en la sección "Install", sin necesidad de construir una sección ***CopyFiles***.

```
[file-list-section]
destination-file-name[,source-file-name][,temporary-file-name][,flag]
[destination-file-name[,source-file-name][,temporary-file-name]][,flag]
.
.
.
```

destination-file-name

En esta opción debe ir el archivo destino. Si no se coloca nada en "source-file-name", este archivo es también tomado para este elemento.

Source-file-name

Nombre del archivo destino.

Temporary-file-name

Nombre del archivo temporal para la operación de copia. El instalador copia el archivo fuente, pero le da un nombre temporal. La siguiente vez que el sistema operativo arranca, renombra el archivo temporal al archivo destino.

Flag

Las banderas son opcionales. Pueden ser utilizadas para controlar como los archivos son copiados. Los valores están en base 10, o base 16 si se le antepone el prefijo 0x.

COPYFLG_WARN_IF_SKIP (0x00000001)
Envía un aviso si el usuario elige no copiar un archivo.

COPYFLG_NOSKIP (0x00000002)
No permite al usuario saltarse la copia del archivo.

COPYFLG_NOVERSIONCHECK (0x00000004)
Ignora versiones anteriores del archivo y sobre-escribe en el directorio destino.

COPYFLG_FORCE_FILE_IN_USE (0x00000008)
Fuerza el comportamiento del archivo en uso (file-in-use)

COPYFLG_NO_OVERWRITE (0x00000010)
No sobre-escribe en el directorio destino con el archivo fuente.

COPYFLG_NO_VERSION_DIALOG (0x00000020)
No Sobre-escribe en el directorio destino con el archivo fuente si el archivo existente es más reciente que el archivo fuente..

COPYFLG_REPLACEONLY (0x00000040)
Copia el archivo fuente al directorio destino solo si el archivo ya existe en el directorio destino.

NOTA: Todos los nombres utilizados en esta sección deben estar definidos en la sección ***SourceDiskFiles*** y en los identificadores de directorio que aparecen en ***SourceDiskDFiles***, sección que debe ser definida en ***SourceDiskNames***.

2.3.e) Sección DeleteFiles.

La sección ***DeleteFiles*** Lista los archivos a ser eliminados. El nombre de esta sección debe aparecer en el elemento ***DelFiles*** de la sección ***Install***.

2.3.f) Sección DestinationDirs.

La sección ***DestinationDirs*** define los directorios destino para las operaciones especificadas en las secciones FileList (***CopyFiles, RenameFiles o DeleteFiles***). Opcionalmente se puede especificar un directorio destino por defecto para estas secciones.

```
[DestinationDirs]
file-list-section=drid[,subdir]
.
.
.
[DefaultDestDir=drid[,subdir]]
```

file-list-section

Contiene el nombre de las secciones CopyFiles, RenameFiles o DeleteFiles.

Drid

Este es un identificador de directorio, que puede ser alguno de los siguientes valores:

-01	Path Absoluto (32 bits)
0xffff	Path Absoluto (16 bits)
01	El directorio desde el cual el archivo INF fue instalado
10	Directorio Windows
11	Directorio Sistema %windir%\system32 en Windows
12	Directorio de Controladores %windir%\system32\drivers en Windows
17	Directorio del archivo INF
18	Directorio de ayuda
20	Directorio de Fuentes
21	Directorio de los visores
24	Directorio de Aplicaciones
25	Directorio Compartido
30	Directorio Raíz del disco de arranque
50	directorio sistema %windir%
51	Directorio Spool
52	Directorio de los controladores Spool
53	Directorio de Usuario (Profile)
54	Directorio donde el "ntldr" y "osloader" están localizados.

Subdir

Nombre del directorio dentro del directorio nombrado por ***dir*** del directorio destino.

2.3.g) Sección Device.

La sección ***Device*** proporciona la descripción del dispositivo, identifica la sección ***install*** e identifica el identificador de dispositivo para este. Opcionalmente la sección ***Device*** puede especificar uno o más identificadores de dispositivo de dispositivos compatibles. Puede haber mas de un elemento en esta sección, dependiendo de cuantos dispositivos serán instalados de un mismo fabricante.

```
[device-section]
```

```
device-description=install-section-name,device-id[,compatible-device-id]...
[device-description=install-section-name,device-id[,compatible-device-id]...]
```

device-description

Descripción del dispositivo a instalar

2.3.h) Sección EventLogInstall.

Install-section-name

Nombre de la sección ***Install*** para este dispositivo. El nombre de la sección Install, será adicionado con una extensión denotando el sistema operativo especifico ó la plataforma.

Device-id

Identificador para este dispositivo. Este puede ser utilizado en la sección "ControlFlags" para controlar la forma en que el dispositivo es manejado por el instalador.

Compatible-device-id

Identificador de un dispositivo compatible. Mas de un identificador puede ser proporcionado, pero cada uno debe estar precedido por una coma.

```
[LogiMfg]
%*pnp0f08.DeviceDesc%=Ser_Inst,*PNP0F08          ; Logi serial mouse
```

El ejemplo anterior es la típica sección ***Device*** que contiene un sólo elemento. En este caso, el nombre de la sección ***Install*** es "Ser_inst" y el ***device-id*** es "*PNP0F08".

Para cada controlador instalado utilizando el archivo INF, el instalador usa la información de "Manufacturer" y "Device" para generar una descripción del controlador, Nombre del Fabricante, ID de Dispositivo y una Lista de Entradas de Compatibilidad en el Registro.

2.3.i) Sección HW

La sección ***HW*** es un caso especial de la sección ***Install***. Esta es la única sección del archivo INF que permite que entradas sean realizadas a la rama ***Enum*** del registro

```
[install-section-name.HW]
AddReg=add-registry-section[,add-registry-section]...
```

```
DelReg=del-registry-section[,del-registry-section]...
```

2.3.j) Sección IniFileToRegistry

La sección ***IniFileToRegistry*** mueve líneas o secciones desde el archivo INI al registro, creando o reemplazando las entradas de registro bajo la clave especificada en el registro. El nombre de esta sección debe aparecer como elemento ***Ini2Reg*** en la sección ***Install*** de l archivo INF

2.3.k) Sección Install

La sección ***Install*** identifica las secciones en el archivo INF que contienen descripciones del dispositivo e instrucciones para la instalación de archivos y la información necesaria por los controladores de dispositivo. El nombre de esta sección debe listarse en la sección ***Device*** la cual es asociada con la sección ***Manufacturer***.

```
[install-section-name]
LogConfig=log-config-section-name[,log-config-section-name]...
Copyfiles=file-list-section-name[,file-list-section-name]...
Renfiles=file-list-section-name[,file-list-section-name]...
Delfiles=file-list-section-name[,file-list-section-name]...
UpdateInis=update-ini-section-name[,update-ini-section-name]...
UpdateIniFields=update-inifields-section-name[,update-inifields-section-
name]...
AddReg=add-registry-section-name[,add-registry-section-name]...
DelReg=del-registry-section-name[,del-registry-section-name]...
Ini2Reg=ini-to-registry-section-name[,ini-to-registry-section-name]...
```

Cabe señalar que no todos los elementos mostrados son necesarios o requeridos en la sección ***Install***. Los nombres deben consistir de caracteres imprimibles. Mas de un nombre de sección puede ser listado en un elemento; cada nombre adicional debe estar precedido por una coma. La única excepción a esto es el elemento ***CopyFiles*** el cual no necesita especificar una sección si sólo un archivo será copiado. Es posible utilizar la notación especial de ***CopyFiles*** para copiar un sólo archivo directamente desde la línea ***CopyFiles***. Esto es colocando como prefijo el simbolo @. El Directorio destino será el elemento ***DefaultDestDir*** definido en la sección ***DestinationDirs***.

Con adicionarle una extensión al nombre de la sección ***Install***, es posible tener varias secciones para las distintas plataformas y sistemas operativo si es necesario. Las siguientes extensiones pueden ser utilizadas:

.win Windows 9x/Millenium
.nt Windows NT/2000 todas las plataformas
.ntx86 Windows NT solo x86
.ntmips Windows NT/2000 solo MIPS
.ntalpha Windows NT/2000 solo ALPHA
.ntppc Windows NT/2000 solo PPC

Las extensiones no son sensitivas al caso.

```
[Ser_Inst]
CopyFiles=Ser_CopyFiles, mouclass_CopyFiles

[Ser_CopyFiles]
sermouse.sys

[mouclass_CopyFiles]
mouclass.sys
```

2.3.l) Sección Manufacturer

La sección ***Manufacturer*** identifica al fabricante del dispositivo que será instalado usando el archivo INF, además de listar una sección ***Device*** que describe el dispositivo:

```
[Manufacturer]
manufacturer-name | %strings-key%=device-section-name
```

manufacturer-name
Nombre de le fabricante. Este nombre puede ser una combinación de caracteres imprimibles, pero sólo debe identificar al fabricante.
Strings-key
Nombre de una cadena definida en la sección ***Strings***.
Device-section-name
Nombre de la sección ***Device***. Puede componerse de una combinación de caracteres imprimibles, pero solamente debe identificar a otra sección del archivo INF.

2.3.m) Sección ServiceInstall

La sección ***Service Install*** instala el servicio listado en la entrada ***AddService*** de la sección ***Services***.

```
[install-section-name_ServiceInstallSection]
DisplayName=[Un nombre amigable]
ServiceType=tipo de driver (permitido por la función SDK CreateService)
StartType= SERVICE_BOOT_START (0x0)
           SERVICE_SYSTEM_START (0x1)
           SERVICE_AUTO_START (0x2)
           SERVICE_DEMAND_START (0x3)
           SERVICE_DISABLED (0x4)
ErrorControl=     CRITICAL (0x3)
                  SEVERE (0x2)
                  NORMAL (0x1)
                  IGNORE (0x0)
ServiceBinary=Ruta hacia el archivo binario de servicio
LoadOrderGroup=[grupo de orden de cargado del cual el controlador es
miembro]
Dependencies=+depend-on-group-name[[,depend-on-service-name]...]
StartName=[driver-object-name]
```

A continuación se muestra un ejemplo de una sección típica ***Service Install***.

```
[mouclass_Service_Inst]
DisplayName     = %mouclass.SvcDesc%
ServiceType     = 1                    ; SERVICE_KERNEL_DRIVER
StartType       = 1                    ; SERVICE_SYSTEM_START
ErrorControl    = 1                    ; SERVICE_ERROR_NORMAL
ServiceBinary   = %12%\mouclass.sys
LoadOrderGroup = Pointer Class
```

2.3.n) Sección Services

La sección ***Services*** permite agregar servicios a la máquina. Estos servicios pueden ser eliminados por la inclusión de la entrada ***DelService*** en la sección ***Services***.

```
[install-section-name.Services]
AddService=ServiceName,flag,service-install-section-name[,event-log-install-
section-name]
DelService=ServiceName
```

2.3.o) Sección SourceDisksFiles

La sección ***SourceDisksFiles*** nombra los archivos fuente utilizados durante la instalación e identifica los discos fuente que contienen los archivos. En la medida de permitir la distribución multiplataforma, es posible construir varios tipos de

secciones colocándoles un punto y una extensión de acuerdo a la plataforma. Por ejemplo ***SourceDisksFiles.mips*** ó ***SourceDisksFiles.x86***.

```
[SourceDisksFiles]
filename=disk-number[,subdir][,size]
.
.
.
```

El ***filename*** identifica el archivo en el disco fuente. El ***disk-number*** representa un numero ordinal que referencia al disco que contiene al archivo. Este número debe ser definido en la sección ***SourceDisksNames*** y debe ser mayor o igual a 1. ***Subdir*** representa un parámetro opcional que se refiere a un subdirectorio en el disco fuente donde reside el archivo. ***Size*** es opcional y representa el tamaño del archivo descomprimido.

2.3.p) Sección SourceDisksNames

La sección ***SourcesDisksNames*** identifica y nombra el disco que contiene los archivos fuente para las operaciones de copiado y renombrado. Al igual que la sección anterior, se permite el uso de multiplataformas.

```
[SourceDisksNames]
disk-ordinal="disk-description",tag-file,unused[,path]
.
.
.
```

En el siguiente ejemplo, el archivo write.exe es el mismo para todas las plataformas y esta localizado en el directorio \common en el CD-ROM. El valor 1de ***disk-number*** dirige las funciones de instalación al directorio \common sin importar de cual plataforma se están instalando archivos. El archivo cmd.exe es un archivo especifico de plataforma que se distribuye para todas las plataformas. El valor 2 de ***disk-number*** dirige las funciones de instalación al directorio correcto para cada plataforma. El archivo halnecmp.dll es específico de la plataforma MIPS:

```
[SourceDisksNames]
1 = "Windows NT CD-ROM", file.tag,,\common

[SourceDisksNames.mips]
2 = "Windows NT CD-ROM", file.tag,, \mips

[SourceDisksNames.x86]
2 = "Windows NT CD-ROM", file.tag,, \i386
```

```
[SourceDisksNames.ppc]
2 = "Windows NT CD-ROM", file.tag,, \ppc

[SourceDisksFiles]
write.exe = 1
cmd.exe = 2

[SourceDisksFiles.mips]
halnecmp.dll = 2
```

2.3.q) Sección Strings

La sección ***Strings*** define una o más claves de cadenas. Una clave de cadena es un nombre que representa una cadena de caracteres imprimibles. Aunque normalmente esta sección es la última sección de un archivo INF, las cadenas definidas en esta sección pueden ser utilizadas en cualquier parte del archivo. Cuando estas claves de cadenas se utilizan, deben estar cerradas al inicio y fin por el carácter (%).

```
[Strings]
strings-key=Cadena compuesta de letras y digitos imprimibles. Debe estar
encerrada entre comillas dobles
.
.
.
```

2.3.r) Sección Version

La sección ***Version*** es la cabecera estándar para todos los archivos INF Windows NT (Indistintamente para Windows NT o 2000) y Windows 95 (Indistintamente para Windows 9X o Millenium Me):

```
[Version]
Signature="signature-name"
Class=class-name
ClassGUID=GUID
Provider=INF-creator
LayoutFile=filename.inf[,filename.inf]...
```

Signature name

Este valor sólo puede ser $Windows NT$, $Chicago$, o $Windows95$. Si este valor no es alguna de estas cadenas, el archivo no es aceptado como una archivo INF para las clases y dispositivos reconocidos por Windows NT

Class-name

Define el nombre de clase en el registro para algún dispositivo instalado desde este archivo INF.

GUID

Opcional. Define el GUID (Graphic User Identifier) en el registro para algún dispositivo instalado desde le archivo INF. Los siguientes GUID son definidos por Windows

Adapter	4d36e964L-e325-11ce-bfc1-08002be10318
CD-ROM	4d36e965L-e325-11ce-bfc1-08002be10318
Computer	4d36e966L-e325-11ce-bfc1-08002be10318
Disk Drive	4d36e967L-e325-11ce-bfc1-08002be10318
Display	4d36e968L-e325-11ce-bfc1-08002be10318
FDC	4d36e969L-e325-11ce-bfc1-08002be10318
HDC	4d36e96aL-e325-11ce-bfc1-08002be10318
Keyboard	4d36e96bL-e325-11ce-bfc1-08002be10318
Media	4d36e96cL-e325-11ce-bfc1-08002be10318
Modem	4d36e96dL-e325-11ce-bfc1-08002be10318
Monitor	4d36e96eL-e325-11ce-bfc1-08002be10318
Mouse	4d36e96fL-e325-11ce-bfc1-08002be10318
MTD	4d36e970L-e325-11ce-bfc1-08002be10318
Multifunction	4d36e971L-e325-11ce-bfc1-08002be10318
Network	4d36e972L-e325-11ce-bfc1-08002be10318
Network Client	4d36e973L-e325-11ce-bfc1-08002be10318
Network Service	4d36e974L-e325-11ce-bfc1-08002be10318
Network Transport	4d36e975L-e325-11ce-bfc1-08002be10318
No Driver	4d36e976L-e325-11ce-bfc1-08002be10318
PCMCIA	4d36e977L-e325-11ce-bfc1-08002be10318
Ports	4d36e978L-e325-11ce-bfc1-08002be10318
Printer	4d36e979L-e325-11ce-bfc1-08002be10318
Printer Upgrade	4d36e97aL-e325-11ce-bfc1-08002be10318
SCSI Adapter	4d36e97bL-e325-11ce-bfc1-08002be10318
Sound	4d36e97cL-e325-11ce-bfc1-08002be10318
System	4d36e97dL-e325-11ce-bfc1-08002be10318
Tape Drive	6d807884L-7d21-11cf-801c-08002be10318
Unknown	4d36e97eL-e325-11ce-bfc1-08002be10318

Inf-creator

Identifica el creador del archivo INF. Normalmente este es el nombre de alguna organización que crea el archivo INF.

Filename.inf

Se requiere para algún caso que un archivo INF que contenga plantillas de información (information layout) sobre archivos y discos, y sea requerida para

instalar el controlador. Si este valor no es especificado, la sección ***SourceDisksNames*** y ***SourceDisksFiles*** deben ser incluidas en este archivo INF. El siguiente ejemplo muestra un típico archivo INF utilizando la sección ***Version***.

```
[Version]
Signature="$Windows NT$"
Class=Mouse
ClassGUID={4D36E96F-E325-11CE-BFC1-08002BE10318}
Provider=%Provider%
LayoutFile=layout.inf

[Strings]
Provider="Corporation X"
```

Existen muchas funciones definidas para la manipulación de los archivos INF desde los programas de instalación de las distintas plataformas. Para tener mayor referencia sobre estas funciones, el desarrollador debe referirse a la documentación del SDK Win 32, ya que no es tema de este trabajo puesto que sólo nos referimos a las funciones y código que hace uso la construcción, instalación y utilización del controlador en las distintas plataformas Windows.

2.4) Archivo Inf de instalación del controlador

A continuación hacemos un breve análisis a nuestro archivo de instalación llamado OEMSETUP.INF, que es el utilizado para instalar nuestro controlador ejemplo

Como ya se mencionó en capítulos anteriores, requerimos de varios archivos para la instalación del controlador. Estos archivos son:

a) Archivo del controlador PEDROGRH.DLL.
b) Archivo del GPD PEDROGRH.GPD
c) Archivo INF OEMSETUP.INF

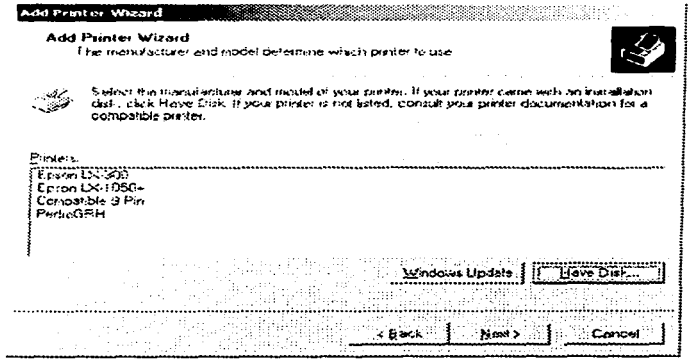
El primero es nuestro controlador o "minidriver" desarrollado para nuestro dispositivo en particular, llamado PEDROGRH.DLL. En nuestro caso, estamos considerando que no utilizamos mas archivos nuestros a parte de PEDROGRH.DLL, es decir, no utilizamos archivos adicionales de ayuda tipo HLP.

A continuación hacemos un breve análisis a nuestro archivo de instalación llamado OEMSETUP.INF, que es el utilizado para instalar nuestro controlador ejemplo:

```
[Version]
Signature="$Windows NT$"
Provider=%PEDRO%
LayoutFile=OemSetup.inf
ClassGUID={4D36E979-E325-11CE-BFC1-08002BE10318}
Class=Printer

[ClassInstall32.NT]
AddReg=printer_class_addreg

[Manufacturer]
"Pedro Gabriel Ramírez Hernández"    = PEDROGRH
```

```
;
; Model Specifications Section
;
 [PEDROGRH]
"Epson LX-300" = eplx300.GPD,EpsonEpson_LX-3005681,Epson_LX-300
"Epson LX-1050+" = epx1050p.GPD,EpsonEpson_LX-1050+,Epson_LX-1050+
"Compatible 9 Pin" = epcomp9.GPD,Epson_Compatible_9_Pin
"PedroGRH" = pedrogrh.GPD,PedroGRH, PedroGRH-A, PedroGRH-8000
;
; Installer section(s) referenced above.
;
[pedrogrh.GPD]
CopyFiles=@pedrogrh.dll,@pedrogrh.GPD
DataSection=UNIDRV_DATA
Include=NTPRINT.INF
Needs=UNIDRV.OEM

[eplx300.GPD]
CopyFiles=@ep9res.dll,@eplx300.GPD
DataSection=UNIDRV_DATA
Include=NTPRINT.INF
Needs=UNIDRV.OEM

[epx1050p.GPD]
CopyFiles=@ep9res.dll,@epx1050p.GPD
DataSection=UNIDRV_DATA
Include=NTPRINT.INF
Needs=UNIDRV.OEM

[epcomp9.GPD]
CopyFiles=@ep9res.dll,@epcomp9.GPD
DataSection=UNIDRV_DATA
Include=NTPRINT.INF
Needs=UNIDRV.OEM

[DestinationDirs]
DefaultDestDir=66000

[SourceDisksNames.x86]
```

```
4 = %sp%,,,\i386

[SourceDisksNames.alpha]
4 = %sp%,,,\alpha

[SourceDisksNames.ppc]
4 = %sp%,,,\ppc

[SourceDisksFiles]
ep9res.dll = 1
pedrogrh.GPD = 1
eplx300.GPD = 1
epx1050p.GPD = 1
epcomp9.GPD = 1;
;
; Localizable Strings
;
[Strings]
PrinterClassName="Printers"
MS=Microsoft
PEDRO="Pedro Software"
disk1="PedroGRH Printer Driver Setup Disk"
```



EXPLICACION DE APARTADOS

VERSION: En este archivo observamos que se cumplen las características antes mencionadas de los archivos INF. En la sección Versión, tenemos sólo que destacar que en la entrada "Signature", tenemos la opción $Windows NT$, así como en la entrada LayoutFile, tenemos el propio "OemSetup.Inf", ya que será el archivo plantilla para nuestra instalación

MANUFACTURER: Este apartado contiene a la empresa ó razón social que desarrolló el dispositivo hardware de impresión, así como el controlador software. Además la cadena de caracteres que sea colocada ahí, será el nombre de la siguiente sección, la cual es particular para cada tipo, ya que es creada por el desarrollador.

Este apartado, como se mencionaba en el párrafo anterior, aunque todos los archivos "oemsetup.inf" lo deben llevar, el nombre es particular para cada desarrollo. En nuestro caso, la única sentencia que tiene este apartado es:

"Pedro Gabriel Ramírez Hernández" = PEDROGRH

Que es la cadena que aparecerá durante la fase de instalación al seleccionar el dispositivo a instalar. A esta cadena se le asigna otro apartado, que es el que le indica las partes de las cuales se va a componer la instalación del controlador.

PEDROGRH.GPD: Esta parte muestra las partes de la instalación, como son:

```
CopyFiles=@pedrogrh.dll,@pedrogrh.GPD
DataSection=UNIDRV_DATA
Include=NTPRINT.INF
Needs=UNIDRV.OEM
```

La primera sección indica los componentes que serán copiados durante la instalación. En nuestro caso en particular, como nuestro controlador se compone únicamente del archivo PEDROGRH.DLL, este es el único archivo que será copiado del disco de instalación y por lo tanto, se le antepone el carácter @ para indicarle al instalador que es un único archivo. Además, en esta primera sección esta incluido un apartado llamado RASDD, que indica los archivos que componen al subsistema de impresión para dispositivos tipo raster.

DESTINATIONDIR: Este apartado o sección contiene la entrada DefaultDestDir, que nos indica mediante un valor numérico el directorio por defecto a donde se copiaran los archivos fuente. Para nuestro caso particular, el valor 66000 llama al valor ***SetupSetDirectoryId*** para colocar el valor del directorio destino en tiempo de ejecución.

```
[DestinationDirs]
DefaultDestDir=66000
```

En la sección ***SourceDisksNames*** tenemos varios elementos que hemos venido explicando con anterioridad. Como podemos observar tenemos que hay varios apartados con el mismo nombre pero con distinta extensión. La extensión depende de la plataforma para la cual existe soporte. En nuestro ejemplo tenemos la plataforma x86, Alpha y PPC.

```
[SourceDisksNames.x86]
4 = %sp%,,,\i386
```

```
[SourceDisksNames.alpha]
4 = %sp%,,,\alpha

[SourceDisksNames.ppc]
4 = %sp%,,,\ppc
```

VALORES

Identificador = descripción, archivo tag, no usado, subdirectorio

En la sección ***SourceDisksFiles*** tenemos los cinco archivos necesarios para la instalación de nuestro controlador, así como sus identificadores, y sus directorios fuente y destino, identificados por valores numéricos.

```
[SourceDisksFiles]
ep9res.dll = 1
pedrogrh.GPD = 1
eplx300.GPD = 1
epx1050p.GPD = 1
epcomp9.GPD = 1;
```

STRINGS

La sección ***Strings*** no requiere mayor explicación ya que sus elementos son sólo asignaciones e identificadores a cadenas reales que son las que el sistema de instalación substituirá en tiempo de instalación todos los valores de estas cadenas donde encuentre las asignaciones del lado izquierdo, encerradas entre signos de porcentaje. Se recomienda analizar detenidamente el archivo INF, ya que el archivo por si mismo, no requiere mayor explicación.

Capitulo 3

Versión piloto de un controlador de impresión que cumpla con los estándares mínimos utilizando el DDK (Device Driver Kit).

3.1 Particularidades de la herramienta Minidriver Development Tool (MDT) incluida en el Device Driver Kit para Windows 2000

La Herramienta de desarrollo Minidev.exe (Minidriver Development Tool) permite a los Programadores de Controladores de Dispositivo, fácilmente crear un ***Minidriver***, para algún dispositivo de impresión de tipo "Raster". Esto permite a los Datos tipo GPC estar organizados y definidos dentro de un Minidriver ya sea para una sola impresora ó para un grupo similar de impresoras raster. El MDT automáticamente combina el Archivo de Descripción de Impresora con el Código Plantilla Interfaz del RasDD para crear un Minidriver lo que provee de manera intrínseca los recursos que el RasDD requiere para ejecutar un dispositivo de Impresión; además de eliminar la necesidad de escribir un Complejo Controlador para cada tipo de Impresora que a su vez forma parte de un mismo tipo de Dispositivos de Impresión.

El MDT conjunta los datos GPD y otros archivos relacionados para crear un ***Minidriver*** para Windows NT, y esta en posibilidad de proveer código para los dispositivos de impresión:

Con el objeto de ahorrar tiempo de desarrollo, es posible comenzar con los datos GPC que son proporcionados por el DDK, ya que se adapta a las características generales de la mayoría de los Dispositivos de Impresión tipo Raster y sólo personalizar algunos parámetros para algún dispositivo específico.

Básicamente un UNIDRV consiste de uno o mas archivos GPD, un archivo de recursos RC, en el que se definen los textos, iconos y mapas requeridos por el minidriver. Adicionalmente se requieren archivos GTT y UFM, los cuales definen

las características de las fuentes residentes. Existen dos maneras de crear un minidriver.

1) Si se tiene un anterior "driver" para Win16 o Win32, este es fácilmente convertible con la herramienta MDT
2) Se puede basar en los "minidrivers" proporcionados por el DDK

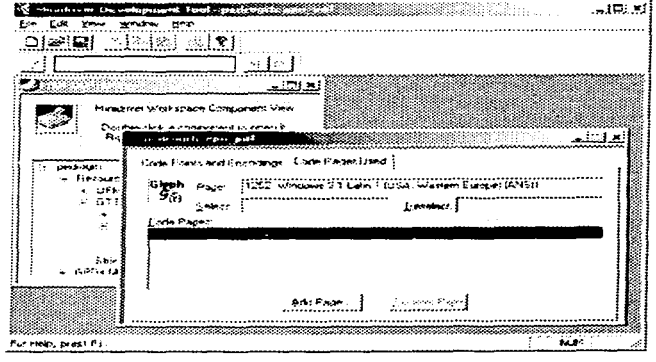
Si el dispositivo soporta fuentes de dispositivo, entonces se deberán proporcionar archivos GTT y UFM

La tabla de recursos GPD en MDT, requiere hacer una colección de información acerca del dispositivo en cuestión para ser usado en el ***Minidriver***.

Algunos temas importantes a considerar antes de tratar la creación de la interface de usuario del mismo Controlador son:

3.1.a) GTT o CTT (Glyph Translation Table) Tabla de Traslación de Caracteres

Usada para remapear caracteres desde los caracteres nativos del dispositivo y el conjunto de símbolos al conjunto ANSI de Windows. Cada fuente de dispositivo debe ser representada por un archivo GTT, es cual es un archivo binario construido utilizando las estructuras GTT en lenguaje C



3.1.b) Archivo de Descripción Genérico de Impresora GPD (Generic Printer Description)

Para habilitar o deshabilitar las características de un dispositivo de Impresión, el MDT recibe y provee a la vez comandos de los llamados ESC (Escape) para hacer estas funciones a través del Minidriver. Por ejemplo, Para nuestra Impresora

en cuestión, el comando para seleccionar la calidad de impresión en modo estándar, es:

ESC "x" n

que traducido a comandos hexadecimales es:

1B 78 n

n puede tomar el valor de 0 ó el valor de 1, para la calidad ***Draft ó Near Letter Quality*** respectivamente, lo que daría como resultado alguno de los siguientes dos comandos posibles:

ESC 78 0 ó 1B 78 0

ESC 78 1 ó 1B 78 1

En el MDT, el comando anterior es representado por \x1B\x78\x00 ó \x1B\x78\x01. Como se podrá observar, en el "MDT" los comandos ESC en formato ASCII para el control de funciones son convertidos a sus equivalentes en hexadecimal.

. Los Comandos también pueden contener parámetros proporcionados por un "RasDD" en tiempo de ejecución. Por ejemplo, en nuestro caso de Impresora, el comando de alimentación de papel (One n/216 inch line feed) es:

ESC "J" n ó

1B 4A n

Lo que este comando realiza es avanzar (alimentar) el papel un n/216 de pulgada, donde ***n*** puede ser desde 1 hasta 255, sin mover la posición del cursor a la izquierda ó derecha respectivamente. El comando respectivo para el MDT sería:

\x1B\x4A\xn

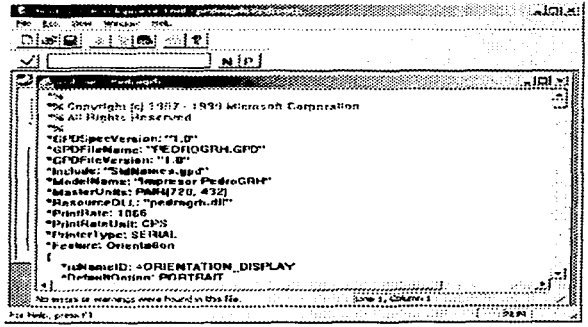
Los archivos GPD son utilizados para la creación de minidrivers, el cual puede contener uno a varios archivos GPD. Estos archivos están escritos en un lenguaje GPD, el cual contiene entradas para proveer la siguiente información:

Atributos de la impresora
Comandos de la impresora
Características que describen las capacidades de la impresora
Opciones de impresión
Descripción de las fuentes de impresora
Sentencias condicionales para describir dependencias entre atributos de impresora y configuración de la misma.
Las entradas de los archivos GPD tienen el siguiente formato:

*EntryName: EntryValue {GPD_FileEntry, GPD_FileEntry .....}

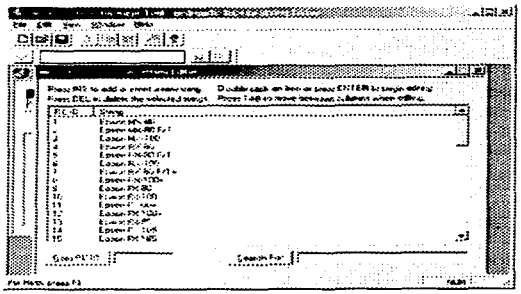
El EntryName es siempre una palabra clave predefinida que el "parser" de GPD del "Unidrv" reconoce por el asterisco, y debe ser uno de los tipos válidos GPD. El siguiente ejemplo muestra como trabajan los EntryNames:

```
*Feature: Orientation
{4
     *Name: "Orientation"
     *DefaultOption: Portrait
     *Option: Portrait
     {
          *Name: "Portrait"
          *Command: CmdSelect
          {
               *Order: DOC_SETUP.7
               *Cmd: "<1B>&l0O"
          }
     }
     *Option: Landscape_CC90
     {
          *Name: "Landscape"
          *Command: CmdSelect
          {
               *Order: DOC_SETUP.7
               *Cmd: "<1B>&l1O"
          }
     }
}
```



3.1.c) Cadenas

Los datos se refieren a la tabla de cadenas de recursos por índices. Las cadenas de recursos están especificadas en un número de cajas de diálogo del MDT. Por ejemplo, la cadena indica los modelos de impresoras que el controlador soporta. En la Interfaz de usuario, las cadenas definen como dar formato a la resolución para la impresora y como desplegar información para el usuario. Otras cadenas definen información como los nombres de las fuentes de papel. La cadena indica el lugar específico del actual ID en el archivo RC. Las cadenas se encuentran en un formato que puede ser traducido a otro lenguaje si es necesario.



3.1.d) Unidades Maestras y Sistema Coordenado

Un dispositivo RasDD utiliza el Sistema Coordenado horizontal y Vertical Sencillo(X, Y), donde las unidades en este sistema son llamadas Unidades Maestras ó Master Units. Estas Unidades Maestras son el Mínimo Común Múltiplo de varias resoluciones soportadas por el Impresor. El RasDD puede sumar, multiplicar o dividir un valor por la Unidad Maestra para calcular los argumentos de los comandos de impresora que usan estas coordenadas.

La mayoría de impresoras soportan una variedad de resoluciones verticales y horizontales, por ejemplo, el comando para la alimentación inmediata de línea trabaja a una resolución de 1/288 de pulgada, además de soportar la resolución:

horizontal de 1/80, 1/160, 1/320 de pulgada
vertical de 1/288 y 1/96

El unidriver provee un sencillo sistema coordenado llamado Unidades Maestras, las cuales son expresadas en pares (x, y), donde X es el mínimo común múltiplo de las resoluciones en horizontal y Y es el MCM de las resoluciones en vertical. El mínimo común múltiplo para

Vertical es: 320
Horizontal es: 576

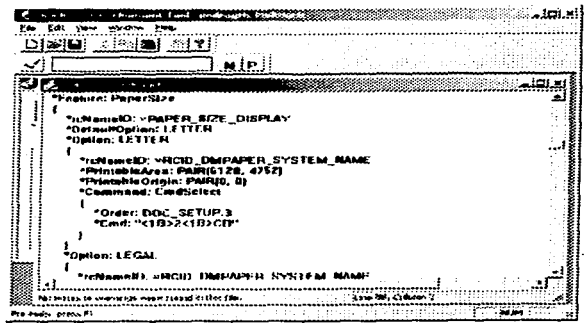
Para especificar unidades maestras, se utiliza el atributo:

*MasterUnits: PAIR(320, 576)

Generalmente las posiciones y tamaños se expresan también en unidades maestras, por ejemplo, para un tamaño de papel de 9 x 12 pulgadas, tenemos:

(9 x 320 = 2880)
(12 x 576 = 6912)

*MaxSize: PAIR(2880, 6912)



3.1.e) Directivas del PreProcesador

Los archivos GPD pueden contener directivas del preprocesador, las cuales son utilizadas para el "parseo" de las secciones del control condicional. La siguiente tabla describe las directivas que pueden ser utilizadas:

Directiva del Preprocesador	**Definición**
***Define:** *SymbolName*	
	Define un símbolo.
***Undefine:** *SymbolName*	
	Remueve un símbolo previamente definido.
***Ifdef:** *SymbolName*	
	Indica el comienzo de un bloque de entradas GPD. Si el símbolo esta definido, las entradas entre *Ifdef y *endif, serán procesadas.
***Elseifdef:** *SymbolName*	
	Si el símbolo esta definido, y el símbolo previo a ***Ifdef** o ***Elseifdef** no esta definido, Las entradas GPD de esta esta directiva y el siguiente ***Ifdef**, ***Elseifdef**, ***Else**, o ***Endif** serán procesadas por el "parser" GPD.
***Else:**	
	Si el símbolo especificado por el anterior ***Ifdef** o ***Elseifdef** es indefinido, Las entradas GPD entre esta directiva y el siguiente ***Ifdef** o ***Endif** son procesadas por el parser GPD.
***Endif:**	
	Indica el fin de un bloque de entradas GPD.
***Include:** "*FileName*"	
	Especifica el nombre de un archivo GPD adicional.
***SetPPPrefix:** *PrefixString*	
	Cambia la cadena prefijada.

Las directivas condicionales pueden ser anidadas como sigue:

***Ifdef:** *Symbol1*
Sección de archivo *GPD*
***Elseifdef:** *Symbol2*
Sección de archivo *GPD*
***Elseifdef:** *Symbo3*
Sección de archivo *GPD*
***Elseifdef:** *Symbol4*
Sección de archivo *GPD*
...
***Else:**
Sección de archivo *GPD*
***Endif:**

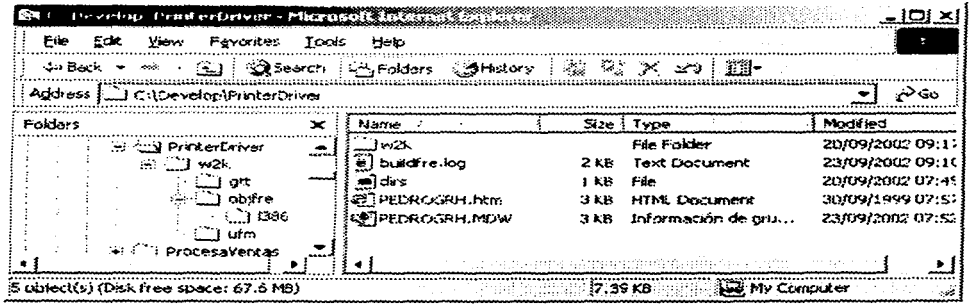
Simbolo	**Donde se define**	**Definición**
WINNT_50	GPD preprocesador para Windows 2000	Ambiente Windows 2000.
WINNT_40	GPD preprocesador para Windows NT 4.0 y Windows 2000.	Ambiente Windows NT 4.0.
PARSER_VER_1.0	GPD preprocesador para Windows NT 4.0 y Windows 2000.	Parser GPD versión 1.0

3.2 Creación de un Archivo de Datos de Controlador utilizando la herramienta MDT de Windows 2000

En este capitulo trataremos la creación de un archivo de datos a partir de algunos existentes.

A continuación realizaremos un ejemplo, donde listaremos una serie de pasos para crear un minidriver utilizando los ejemplos suministrados por el DDK como plantilla. Para este caso utilizaremos los archivos pertenecientes al dispositivo ***epson de 9 agujas (epson9)***, por asemejarse lo más posible sus comandos escape a los comandos escape de nuestro dispositivo impresor para el cual desarrollaremos el controlador. Este dispositivo es un ***Star Micronix NX1040***,

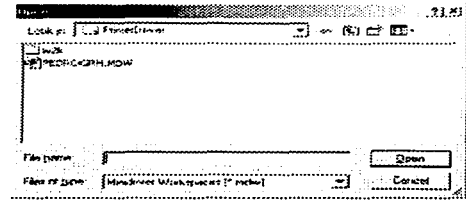
1- Se crea un directorio destino para los nuevos archivos del minidriver



2.- Se crea en nuestro directorio el Midriver Develop Wizard (MDW) a partir de uno de los ejemplos de Epson:

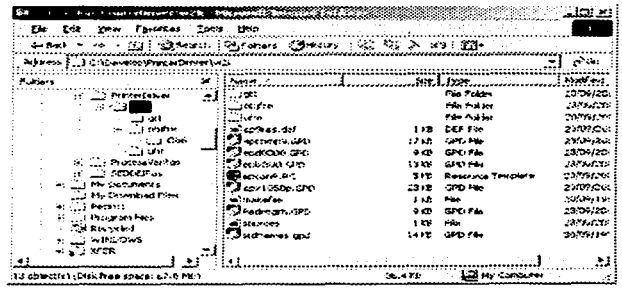
3.- Se abre desde el MDT , lo que crea las siguientes carpetas para su desarrollo

- ...\MiDire\w2k
- ...\MiDire\w2k\gtt
- ...\MiDire\w2k\ufm



4.- Después se crean los siguientes archivos:

- ...\MiDire\w2k\ep9res.def
- ...\MiDire\w2k\epson9.rc
- ...\MiDire\w2k\makefile
- ...\MiDire\w2k\sources
- ...\MiDire\w2k\epd8000.gpd
- ...\MiDire\w2k\eplx300.gpd
- ...\MiDire\w2k\epx1050.gpd
- ...\MiDire\w2k\pedrogrh.gpd
- ...\MiDire\w2k\stdnames.gpd



NOTA: Para nuestro caso el nombre del controlador que desarrollaremos será PEDROGRH.

5- Se personaliza los datos de la ventana de GPD del controlador.

3.3 Análisis del código GPD que interviene en la realización del Controlador

En Esta sección explicaremos mas detalladamente el funcionamiento del archivo PEDROGRH.GPD creado específicamente para nuestra impresora

```
*GPDSpecVersion: "1.0"
*GPDFileName: "PEDROGRH.GPD"
*GPDFileVersion: "1.0"
*ModelName: "Impresor PedroGRH"
*ResourceDLL: "pedrogrh.dll"
```

Estas entradas nos especifican la versión del lenguaje GPD, la versión del archivo, así como el nombre lógico y físico del archivo GPD. El Módelo viene indicado como un string que aparecerá como los modelos disponibles durante la instalación. *ResourceDLL nos indica el nombre físico del archivo de salida o ejecutable

```
*Include: "StdNames.gpd"
```

El archivo "StdNames" contiene las opciones estándar de entradas posibles que son reconocidas como por el lenguaje GPD. La siguiente tabla muestra las opciones permitidas:

Feature Name	Nombres estándares	Permite opciones personalizadas ?
Collate	ON, OFF	No.
ColorMode	No existen opciones estándar.	Si. Opciones como "Handling Color" y Controlling Image Quality"
Duplex	HORIZONTAL, VERTICAL, NONE	No.
Halftone	HT_PATSIZE_2x2 HT_PATSIZE_2x2_M HT_PATSIZE_4x4 HT_PATSIZE_4x4_M HT_PATSIZE_6x6 HT_PATSIZE_6x6_M HT_PATSIZE_8x8 HT_PATSIZE_8x8_M HT_PATSIZE_10x10 HT_PATSIZE_10x10_M HT_PATSIZE_12x12 HT_PATSIZE_12x12_M HT_PATSIZE_14x14 HT_PATSIZE_14x14_M HT_PATSIZE_16x16 HT_PATSIZE_16x16_M HT_PATSIZE_SUPERCELL	Si.

	HT_PATSIZE_SUPERCELL_M HT_PATSIZE_AUTO	
InputBin	AUTO CASSETTE ENVFEED ENVMANUAL FORMSOURCE LARGECAPACITY LARGEFMT LOWER MANUAL MIDDLE SMALLFMT TRACTOR UPPER	Si.
MediaType	GLOSSY STANDARD TRANSPARENCY	Si.
Memory	No hay opciones estandard.	Si.
Orientation	PORTRAIT, LANDSCAPE_CC90, LANDSCAPE_CC270	No.
PageProtect	ON, OFF	No.
PaperSize	10X11 10X14 11X17 12X11 15X11 9X11 A_PLUS A2 A3 A3_EXTRA A3_EXTRA_TRANSVERSE A3_ROTATED A3_TRANSVERSE A4 A4_EXTRA A4_PLUS A4_ROTATED A4_TRANSVERSE A4SMALL A5 A5_EXTRA A5_ROTATED A5_TRANSVERSE A6 A6_ROTATED B_PLUS B4 B4_JIS_ROTATED B5 B5_EXTRA B5_JIS_ROTATED B5_TRANSVERSE B6_JIS B6_JIS_ROTATED	Si. Los nombres personalizados no deben de exceder la longitud especificada por CCHFORMNAME en *wingdi.h*.

	CSHEET CUSTOMSIZE DBL_JAPANESE_POSTCARD DBL_JAPANESE_POSTCARD_ROTATED DSHEET ENV_10 ENV_11 ENV_12 ENV_14 ENV_9 ENV_B4 ENV_B5 ENV_B6 ENV_C3 ENV_C4 ENV_C5 ENV_C6 ENV_C65 ENV_DL ENV_INVITE ENV_ITALY ENV_MONARCH ENV_PERSONAL ESHEET EXECUTIVE FANFOLD_LGL_GERMAN FANFOLD_STD_GERMAN FANFOLD_US FOLIO ISO_B4 JAPANESE_POSTCARD JAPANESE_POSTCARD_ROTATED JENV_CHOU3 JENV_CHOU3_ROTATED JENV_CHOU4 JENV_CHOU4_ROTATED JENV_KAKU2 JENV_KAKU2_ROTATED JENV_KAKU3 JENV_KAKU3_ROTATED JENV_YOU4 JENV_YOU4_ROTATED LEDGER LEGAL LEGAL_EXTRA LETTER LETTER_EXTRA LETTER_EXTRA_TRANSVERSE LETTER_PLUS LETTER_ROTATED LETTER_TRANSVERSE LETTERSMALL NOTE P16K P16K_ROTATED P32K	

	P32K_ROTATED P32KBIG P32KBIG_ROTATED PENV_1 PENV_1_ROTATED PENV_10 PENV_10_ROTATED PENV_2 PENV_2_ROTATED PENV_3 PENV_3_ROTATED PENV_4 PENV_4_ROTATED PENV_5 PENV_5_ROTATED PENV_6 PENV_6_ROTATED PENV_7 PENV_7_ROTATED PENV_8 PENV_8_ROTATED PENV_9 PENV_9_ROTATED QUARTO STATEMENT TABLOID TABLOID_EXTRA	
Resolution	No hay opciones estándar.	Si.

***MasterUnits: PAIR(720, 432)**

En esta entrada, establecemos las unidades maestras como coordenadas (x, y), donde el mínimo común múltiplo es:

Horizontal: 720 (Acepta avances de 1/90, 1/180, 1/360 de pulgada)
Vertical: 432 (Acepta avances de 1/72, 1/216 de pulgada)

***PrintRate: 1066**
***PrintRateUnit: CPS**
***PrinterType: SERIAL**

En estas entradas establecemos la velocidad de impresión a 1066 caracteres por segundo. Cabe señalar que también pueden darse en PPM, LPM, IPM (páginas por minuto, líneas por minuto o pulgadas por minuto). El tipo por defecto es serial

***Feature: Orientation**
{
***rcNameID: =ORIENTATION_DISPLAY**
***DefaultOption: PORTRAIT**

```
    *Option: PORTRAIT
    {
      *rcNameID: =PORTRAIT_DISPLAY
    }
    *Option: LANDSCAPE_CC270
    {
      *rcNameID: =LANDSCAPE_DISPLAY
    }
}
```

En esta entrada establecemos si la impresora puede manejar las distintas orientaciones posibles. Para Windows solo podemos tener:

PORTRAIT
LANDSCAPE_CC90
LANDSCAPE_CC270

```
*Feature: InputBin
{
   *rcNameID: =PAPER_SOURCE_DISPLAY
   *DefaultOption: Option1
   *Option: Option1
   {
      *rcNameID: 262
      *Command: CmdSelect
      {
         *Order: DOC_SETUP.2
         *Cmd: "<1B1904>"
      }
   }
   *Option: Option2
   {
      *rcNameID: 261
      *Command: CmdSelect
      {
         *Order: DOC_SETUP.2
         *Cmd: "<1B1900>"
      }
   }
}
```

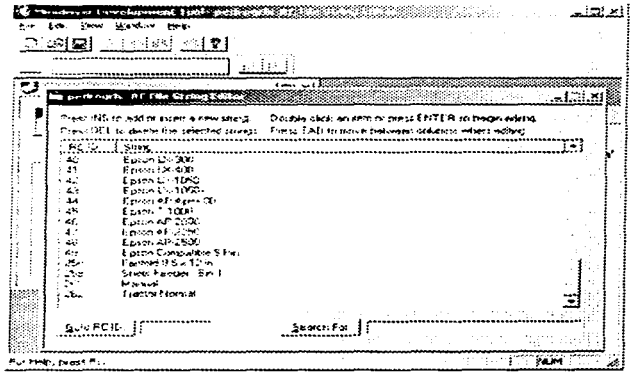
En esta entrada se establecen los valores de la bandeja de entrada. La opción 1 establece la alimentación por tractor, mientas que la opción 2 establece la alimentación manual.

La entrada ****RcnameId*** establece como se llamara este recurso en la ventana de dialogo de las propiedades de la impresora: Tractor y Manual respectivamente.

La entrada ***Cmd:** "<1B1900>"* selecciona el comando para establecer alimentación por tractor. Para nuestra impresora, éste comando en hexadecimal es:

1B ESC (Escape)
19 EM (End of Medium)
00

Los valores de "strings" se pueden visualizar en la siguiente figura:



Controlando la calidad de la imagen por GPD

Para el tema de las entradas de del archivo GPD para el control de la calidad de la imagen, la interfase de usuario del UNIDRV proporciona un conjunto de tres controles tipo "radio buttons", con las opciones "draft", "better" y "best", para la calidad de la imagen en el trabajo de impresión. Las distintas opciones mantienen una relación entre velocidad de impresión y resolución de la imagen. Estas opciones se definen se definen mediante tres entradas del lenguaje GPD:

***DraftQualitySettings**
***BetterQualitySettings**
***BestQualitySettings**

Cada una de esas entradas esta asociada con uno de los controles mencionados, y cada entrada acepta una lista de opciones

El formato para este control es:.

*XXXX**QualitySettings: LIST(***FeatureName.OptionName*, *FeatureName.OptionName*, *FeatureName.OptionName*, ...**)**

donde cada "*FeatureName*" es un nombre asociado con la entrada ***Feature,** y "*OptionName*" es un nombre asociado con alguna de las características de la entrada ***Option**. Una lista vacía causa que los controles de "radio Button" se coloquen en "grayed". Una entrada adicional representa la calidad de la imagen por defecto, donde el formato es como sigue:

***DefaultQuality:** *DefaultQuality*

donde *DefaultQuality* puede ser cualquiera de **DRAFTQUALITY**, **BETTERQUALITY**, o **BESTQUALITY**.

Estas entradas GPD puede a su vez estar asociadas con alguna opción de características **ColorMode** y **MediaType**. Típicamente, estas entradas se encuentran dentro de sentencias de lógica condicional, de acuerdo a lo mostrado en el siguiente ejemplo:

```
*switch: Colormode {
    *case: Mono {
    *switch: MediaType {
        *case: CLAYCOATED {
            *DraftQualitySettings:  LIST(Option List)
            *BetterQualitySettings:  LIST(Option List)
            *BestQualitySettings:  LIST(Option List)
            *DefaultQuality:  BESTQUALITY }
        *case: GLOSSY {
            *DraftQualitySettings:  LIST(Option List)
            *BetterQualitySettings:  LIST(Option List)
            *BestQualitySettings:  LIST(Option List)
            *DefaultQuality:  BETTERQUALITY
        *default:
            *DraftQualitySettings:  LIST(Option List)
            *BetterQualitySettings:  LIST(Option List)
            *BestQualitySettings:  LIST(Option List)
            *DefaultQuality:  DRAFTQUALITY }}}
    *default: {
    *switch: MediaType {
        *case: CLAYCOATED {
            *DraftQualitySettings:  LIST(Option List)
            *BetterQualitySettings:  LIST(Option List)
            *BestQualitySettings:  LIST(Option List)
            *DefaultQuality:  BESTQUALITY }
```

```
        *case: GLOSSY {
            *DraftQualitySettings:  LIST(Option List)
            *BetterQualitySettings:  LIST(Option List)
            *BestQualitySettings:  LIST(Option List)
            *DefaultQuality:  BETTERQUALITY }
        *default: {
            *DraftQualitySettings:  LIST(Option List)
            *BetterQualitySettings:  LIST(Option List)
            *BestQualitySettings:  LIST(Option List)
            *DefaultQuality:  DRAFTQUALITY }}}
}
```

Cuando se utilizan entradas GPD para la configuración de la calidad de la imágen, las siguientes reglas deben ser observadas:

- Utilizar siempre las 4 entradas. LA especificación de una lista vacia esta permitida y causa la deshabilitacion de los controles tipo "radio".
- Todas las 4 entradas deben ser utilizadas para todos Modos.
- Las listas de opción, en las entradas de calidad, no deben violar algunas "constraints" que se hubiesen especificado.

```
*Feature: Resolution
{
  *rcNameID: =RESOLUTION_DISPLAY
  *DefaultOption: Option1
  *Option: Option1
  {
    *Name: "120 x 144 " =DOTS_PER_INCH
    *DPI: PAIR(120, 144)
    *TextDPI: PAIR(120, 144)
    *PinsPerLogPass: 16
    *PinsPerPhysPass: 8
    EXTERN_GLOBAL: *StripBlanks: LIST(LEADING,TRAILING)
    *SpotDiameter: 140
    *Command: CmdSendBlockData { *Cmd : "<1B>L" %I{NumOfDataBytes} }
  }
  *Option: Option2
  {
    *Name: "240 x 144 " =DOTS_PER_INCH
    *DPI: PAIR(240, 144)
    *TextDPI: PAIR(240, 144)
    *PinsPerLogPass: 16
    *PinsPerPhysPass: 8
    EXTERN_GLOBAL: *StripBlanks: LIST(LEADING,TRAILING)
    *SpotDiameter: 170
    *Command: CmdSendBlockData { *Cmd : "<1B>Z" %I{NumOfDataBytes} }
  }
  *Option: Option3
  {
    *Name: "120 x 72 " =DOTS_PER_INCH
```

```
        *DPI: PAIR(120, 72)
        *TextDPI: PAIR(120, 72)
        *PinsPerLogPass: 8
        *PinsPerPhysPass: 8
        EXTERN_GLOBAL: *StripBlanks: LIST(LEADING,TRAILING)
        *SpotDiameter: 100
        *Command: CmdSendBlockData { *Cmd : "<1B>L" %I{NumOfDataBytes} }
    }
}
```

Para la entrada ***Resolution***, tenemos tres opciones, donde la nombramos como un "string" llamado por el atributo ***Name*** conteniendo explícitamente el nombre de tal resolución. Para nuestra impresora tenemos tres resoluciones posibles:

244 x 144	DPI (dots por pulgada)
240 x 144	DPI (dots por pulgada)
120 x 72	DPI (dots por pulgada)

Para la entrada ***Resolution***, podemos tener los siguientes atributos, los cuales son explicados como sigue:

Nombre del atributo	**Parámetro del atributo**	**Comentarios**
***DPI**	PAR de valores numéricos representando los valores X y Y de la impresora, en dots por pulgadas.	Esta entrada es requerida. Los valores establecidos deben ser los mismos ***TextDPI**, o en su defecto deben ser equivalentes a su división entre potencias de dos. Por ejemplo, si ***TextDPI** es PAIR(300, 300), entonces los valores ***DPI** pueden ser los pares PAIR(300, 300), PAIR(150, 150), o PAIR(75, 75), pero no PAIR(100, 100).
***MinStripBlankPixels**	Valor numérico representando el mínimo número de bytes blancos que el UNIDRV puede encontrar en una línea de rastreo antes de cerrar la línea.	Opcional. Si no se especifica, su valor es cero. Es relevante sólo si ***StripBlanks** especifica ENCLOSED.
***PinsPerLogPass**	Valor numérico que representa las líneas de rastreo impresas en una pasada lógica de la cabeza de impresión.	Opcional. Si no se especifica su valor es 1. Se requiere si la impresora desarrolla entrelazado, requiriendo múltiples pasos de la cabeza de impresión para imprimir las líneas de rastreo.
***PinsPerPhysPass**	Valor numérico que representa en número de líneas de rastreo impresas según se mueve la cabeza de impresión a través de la página. Debe ser 1 o un múltiplo de 8.	Opcional.
***RequireUniDir?**	TRUE o FALSE, Especificando si la resolución requiere impresión	Opcional. Si no se especifica su valor es FALSE.

	unidireccional habilitada.	
***SpotDiameter**	Valor numérico representando el diámetro del "spot" como un porcentaje del tamaño del pixel, según la resolución especificada en ***DPI**.	Requerido. Ejemplo: 100 significa que el "spot diameter" es igual al tamaño del pixel. 200 significa que el "spot diameter" es dos veces el tamaño del pixel. 50 significa que el spot diameter es de medio pixel.
***TextDPI**	PAR de valores numéricos representados por X y Y para la resolución de la impresión en de texto en dots por pulgadas.	Requerido. Esta resolución es usada para el dibujo de fuentes y gráficos vectoriales.

```
*Feature: PaperSize
{
   *rcNameID: =PAPER_SIZE_DISPLAY
   *DefaultOption: LETTER
   *Option: LETTER
   {
      *rcNameID: =RCID_DMPAPER_SYSTEM_NAME
      *PrintableArea: PAIR(6120, 4752)
      *PrintableOrigin: PAIR(0, 0)
      *Command: CmdSelect
      {
         *Order: DOC_SETUP.3
         *Cmd: "<1B>2<1B>CB"
      }
   }
   *Option: LEGAL
   {
      *rcNameID: =RCID_DMPAPER_SYSTEM_NAME
      *PrintableArea: PAIR(6120, 6048)
      *PrintableOrigin: PAIR(0, 0)
      *Command: CmdSelect
      {
         *Order: DOC_SETUP.3
         *Cmd: "<1B>2<1B>CT"
      }
   }
   *Option: A4
   {
      *rcNameID: =RCID_DMPAPER_SYSTEM_NAME
      *PrintableArea: PAIR(5954, 5050)
      *PrintableOrigin: PAIR(0, 0)
      *switch: Resolution
      {
         *case: Option1
         {
            *PrintableArea: PAIR(5952, 5049)
         }
         *case: Option2
         {
            *PrintableArea: PAIR(5952, 5049)
```

```
        }
        *case: Option3
        {
          *PrintableArea: PAIR(5952, 5046)
        }
      }
      *Command: CmdSelect
      {
        *Order: DOC_SETUP.3
        *Cmd: "<1B>2<1B>CF"
      }
    }
    *Option: A3
    {
      *rcNameID: =RCID_DMPAPER_SYSTEM_NAME
      *PrintableArea: PAIR(8417, 7145)
      *PrintableOrigin: PAIR(0, 0)
      *switch: Resolution
      {
        *case: Option1
        {
          *PrintableArea: PAIR(8412, 7143)
        }
        *case: Option2
        {
          *PrintableArea: PAIR(8415, 7143)
        }
        *case: Option3
        {
          *PrintableArea: PAIR(8412, 7140)
        }
      }
      *Command: CmdSelect
      {
        *Order: DOC_SETUP.3
        *Cmd: "<1B>2<1B>Cc"
      }
    }
    *Option: A5
    {
      *rcNameID: =RCID_DMPAPER_SYSTEM_NAME
      *PrintableArea: PAIR(4198, 3573)
      *PrintableOrigin: PAIR(0, 0)
      *switch: Resolution
      {
        *case: Option1
        {
          *PrintableArea: PAIR(4194, 3573)
        }
        *case: Option2
        {
          *PrintableArea: PAIR(4197, 3573)
        }
        *case: Option3
```

```
        {
          *PrintableArea: PAIR(4194, 3570)
        }
      }
      *Command: CmdSelect
      {
        *Order: DOC_SETUP.3
        *Cmd: "<1B>2<1B>C2"
      }
    }
  *Option: CUSTOMSIZE
    {
      *rcNameID: =USER_DEFINED_SIZE_DISPLAY
      *MinSize: PAIR(720, 432)
      *MaxSize: PAIR(10080, 9504)
      *MaxPrintableWidth: 9792
      *MinLeftMargin: 0
      *CenterPrintable?: FALSE
      *Command: CmdSelect
      {
        *Order: DOC_SETUP.3
        *Cmd: "<1B>2"
      }
    }
}
```

Esta entrada permite definir los atributos asociados con el tamaño de las distintas páginas que nuestra impresora pueda soportar. Para nuestro caso, la impresora puede soportar:

LETTER
LEGAL
A4
A3
A5
CUSTOMSIZE

Los atributos asociados posibles se definen en la siguiente tabla:

Nombre del atributo	Parámetro del atributo	Comentarios
***BottomMargin**	Valor numérico que representa el mínimo margen permisible en unidades maestras para X. Este valor es relativo a la parte inferior de la página física.	Opcional, si no es especificado, su valor por defecto es cero.
***CenterPrintable?**	TRUE o FALSE, indicando si el valor especificado por ***MaxPrintableWidth** será centrado.	Opcional. Si no se especifica, el área imprimible es alineada al margen derecho.

***CursorOrigin**	Par (PAIR) de valores numéricos representando la posición origen del cursor.	Opcional. Si no se especifica su valor será PAIR (0, 0).
***CustCursorOriginX**	Usado para crear un valor personalizado de origen de cursor para ***CursorOrigin**.	Opcional. Se utiliza solo con la opción CUSTOMSIZE.
***CustCursorOriginY**	Usado para crear un valor personalizado de origen de cursor para ***CursorOrigin**.	Opcional. Se utiliza solo con la opción CUSTOMSIZE
***CustPrintableOriginX**		Opcional.
***CustPrintableOriginY**		Opcional.
***CustPrintableSizeX**	Usado para crear un valor personalizado de origen de cursor para ***PrintableArea**.	Opcional. Se utiliza solo con la opción CUSTOMSIZE
***CustPrintableSizeY**	Usado para crear un valor personalizado de origen de cursor para ***PrintableArea**.	Opcional. Se utiliza solo con la opción CUSTOMSIZE
***MaxSize**	Par (PAIR) de valores numéricos que representan el tamaño máximo permisible donde los valores son X y Y representados	Requerido. Para la opción CUSTOMSIZE Se asume la orientación de portada.
***MaxPrintableWidth**	Valor numérico que representa la máxima área imprimible para X. Es asociado con la opción CUSTOMSIZE.	Requerido. Para la opción CUSTOMSIZE Se asume la orientación de portada.
***MinLeftMargin**	Valor numérico que representa la mínima área imprimible para X. Es asociado con la opción CUSTOMSIZE.	Requerido. Para la opción CUSTOMSIZE Se asume la orientación de portada.
***MinSize**	Par (PAIR) de valores numéricos que representan el tamaño mínimo permisible donde los valores son X y Y representados	Requerido. Para la opción CUSTOMSIZE Se asume la orientación de portada.
***PageDimensions**	Par de valores numéricos (PAIR) que representan el largo y ancho de la páginas en unidades maestras.	Utilizado solo para proveedores de un tamaño de papel personalizado.
***PageProtectMem**	Valor numérico que representa el monto de la memoria de la impresora en Kb requerida para proteger una página.	Requerido si la opción ***PageProtect** es especificada.
***PrintableArea**	Par de valores numéricos representando la longitud de los planos X y Y, en unidades maestras de un área imprimible.	Requerido para todas las opciones.
***PrintableOrigin**	Par de valores numéricos representando el origen de un área imprimible, relativo a la esquina superior izquierda de la página. En unidades maestras.	Requerido para todas las opciones.
***RotateSize?**	TRUE o FALSE, Indicando si el Unidrv puede rotar las dimensiones de la página. Típicamente para Sobres (Envelope).	Opcional. Si no se especifica es FALSE

*TopMargin	Valor numérico representando el mínimo margen superior permisible.	Requerido. Si no se especifica, su valor es cero. Para la opción CUSTOMSIZE Se asume la orientación de portada.

Manejo del Halftoning

El Halftoning tradicional de tipo análogo usa precisamente una pantalla halftoning, compuesta de células de tamaños iguales con espacios iguales entre ellas de centro a centro. El espaciamiento entre células fijas acomoda el grosor de la tinta, toda vez que el tamaño de un punto dentro de una célula puede variar para producir la sensación de tono continuo. Para simular el tamaño variable del punto, una combinación de racimos de pixeles simulan la pantalla halftoning.

El Controlador envía al GDI las especificaciones relacionadas al dispositivo que el GDI necesita para desarrollar el halftoning por medio de la estructura GDIINFO retornada por la función ***DrvEnablePDEV***. El Controlador especifica el tamaño de la plantilla mediante por el miembro ulHTPatternSize, el cual define el formato de salida preferido por el halftoning. Para este caso, el GDI proporciona numerosos tamaños de plantillas predefinidas desde 2 x 2, hasta 16 x 16. Para cada tamaño de plantilla estándar, existe una versión modificada, la cual es definida por el sufijo _M en el nombre de la plantilla estándar. Por ejemplo, el nombre definido para el patrón de 6 x 6 es HT_PATSIZE_6x6, mientras que su nombre modificado es HT_PATSIZE_6x6_M. Estas versiones modificadas dan mucha mas resolución de color, pero pueden producir efectos colaterales de ruido vertical u horizontal.

El intercambio entre el tamaño de la plantilla patrón ***(resolución espacial),*** y la resolución del color es determinado por el tamaño de la plantilla patrón. Un patrón de degradado (halftoning) más grande produce mejor resolución de color, mientras que una más pequeña resulta en mejor resolución espacial. El proceso de determinar la mejor relación es muchas veces una situación de prueba y error.

Si el formato de color establecido para el número de bits x pixel, usado para el degradado de la imagen "rendering" en ***DrvBPP** es mayor a los bits por pixel soportados por la impresora **(DevBPP x *DevNumOfPlanes)**, entonces, se deben proporcionar capacidades de degradado personalizadas

para proporcionar capacidades de "halftoning", se debe hacer lo siguiente:

- Proporcionar un "**rendering plug-in**" que implemente el método "**IPrintOemUni::ImageProcessing**", el cual se puede implementar mediante la metodología de clases de VC++
- Incluir la característica **Halftone** en la entrada ***Feature** en nuestro archivo GPD, por cada método personalizado de "halftoning".
- Incluir la característica **ColorMode** en la entrada ***Feature** en nuestro archivo GPD. Para cada opción de formato de color, se debe incluir un atributo ***IPCallbackID**, si se desea implementar el método **IPrintOemUni::ImageProcessing**.

El siguiente ejemplo define dos formatos de color y cuatro métodos de "halftoning":

```
*Feature: ColorMode
{
    *Option: ColorFormat1
    {
        *Name: "Color Format 1"
        *DevBPP: 1
        *DevNumofPlanes: 4
        *ColorPlaneOrder: LIST (CYAN, MAGENTA, YELLOW, BLACK)
        *DrvBPP: 4
        *Constraints: LIST (Halftone.CustomHalftoneMethod1,
+                           Halftone.CustomHalftoneMethod2)
    }
    *Option: ColorFormat2
    {
        *Name: "Color Format 2"
        *DevBPP: 24
        *DevNumofPlanes: 1
        *DrvBPP: 8
        *IPCallbackID: 100
        *Constraints: LIST (Halftone.StandardHalftoneMethod1,
+                           Halftone.StandardHalftoneMethod2)
    }
}
*Feature: Halftone
{
    *Option: StandardHalftoneMethod1
    {
        *Name: "Standard Halftone Method 1"
    }
    *Option: StandardHalftoneMethod2
    {
        *Name: "Standard Halftone Method 2"
    }
    *Option: CustomHalftoneMethod1
    {
        *Name: "Custom Halftone Method 1"
    }
    *Option: CustomHalftoneMethod2
    {
        *Name: "Custom Halftone Method 2"
    }
}
```

En este ejemplo, las dos opciones de **ColorMode** (ColorFormat1 y ColorFormat2) representan formatos de colores que el UNIDRV puede administrar. Para ColorFormat2, un atributo ***IPCallbackID** se ha especificado. Si el Usuario selecciona ColorFormat2 como el formato de color, entonces el UNIDRV llama al método COM ***IPrintOemUni::ImageProcessing*** el cual puede retornar el mapa de bits modificado al UNIDRV, o enviarlo directamente al spooler de impresión

```
STDMETHOD
  (ImageProcessing) (
    THIS_
    PDEVOBJ  pdevobj,
    PBYTE  pSrcBitmap,
    PBITMAPINFOHEADER  pBitmapInfoHeader,
    PBYTE  pColorTable,
    DWORD  dwCallbackID,
    PIPPARAMS  pIPParams,
    OUT PBYTE  *ppbResult
    ) PURE;
```

Su función es administrar el halftoning. Uno de los parámetros del método es un apuntador a una cadena representando el nombre del actual método seleccionado.

```
*Feature: Halftone
{
  *rcNameID: =HALFTONING_DISPLAY
  *DefaultOption: HT_PATSIZE_AUTO
  *Option: HT_PATSIZE_AUTO
  {
    *rcNameID: =HT_AUTO_SELECT_DISPLAY
  }
*Ifdef: WINNT_50
  *Option: HT_PATSIZE_SUPERCELL_M
  {
    *rcNameID: =HT_SUPERCELL_DISPLAY
  }
*Endif:
  *Option: HT_PATSIZE_6x6_M
  {
    *rcNameID: =HT_DITHER6X6_DISPLAY
  }
  *Option: HT_PATSIZE_8x8_M
  {
    *rcNameID: =HT_DITHER8X8_DISPLAY
  }
}
```

Esta entrada define los atributos para la construcción gráfica mediante "halfoning", donde los valores posibles son:

Nombre del atributo	Parámetro del atributo	Comentarios
***HTCallbackID**	Valor numérico positivo, pasado al conector de rendering (plug-in's).	Requerido si el método **IPrintOemUni::HalftonePattern** es proporcionado.
***HTNumPatterns**	Valor numérico representando el numero de patrones para medios tonos..	Opcional. Puede ser 1 o 3, donde 3 implica patrones separados por rojo, verde y azul. Si no es especificado, su valor es 1.
***HTPatternSize**	Par de valores numéricos (PAIR) representando el ancho y largo en pixeles del patron especificado en ***rcHTPatternID**.	Requerido si ***rcHTPatternID** es especificado. El tamaño máximo del patron es PAIR (256, 256). El ancho y largo deben ser divisibles entre 4, para que se puedan alocar como DWORD.
***rcHTPatternID**	Identificador de recurso para un patrón RC_HTPATTERN representando el patrón de datos de los medios tonos.	Requerido si un patrón de medios tonos es proporcionado en un recurso DLL.

Comandos de Impresora generados dinámicamente

Por cada especificación de un comando de impresora en un GPD, se pueden utilizar dos métodos
:

Colocar la cadena del comando directamente en el archivo GPD. Esto es, cuando se escoge este método, el UNIDRV envía el comando al spooler de impresión en los momentos apropiados.

- Proporcionar una rutina o función de llamada. Esto es, el UNIDRV llama a la función cuando es momento de enviar el comando, y la función es responsable de enviar dicho comando al spooler de impresión.

Para colocar una cadena de comando en un archivo GPD, es necesario hacerlo mediante el atributo ***Cmd**, en la entrada ***Command**, mientras que para proporcionar código que dinámicamente genere la cadena de comando, se debe hacer lo siguiente:

- Proporcionar un plug-in que implemente el método **IprintOemUni::CommandCallback.**
- Incluir un atributo ***CallbackID**, y opcionalmente el atributo ***Params**, en la entrada ***Command** del archivo GPD.

Cuando el UNIDRV esta listo para enviar el comando a la impresora, checa la base de datos del **Minidriver**, para determinar si el comando ha sido especificado con el atributo ***Cmd** o ***CallbackID**.

Para especificar una entrada de comando, se utiliza el formato:

***Command:** *CommandName* {*CommandAttributes*}

donde *CommandName* es uno de los nombres de comando predefinidos, y *CommandAttributes* son los atributos propios. Por ejemplo:

```
*Command: CmdStartPage
{
   *Order: PAGE_SETUP.100
   *Cmd: "<0D>"
}
```

La siguiente tabla muestra los atributos de comando en orden alfabético, y sus respectivos parámetros.

Nombre del Atributo	Parámetro	Comentarios
***CallbackID**	Valor numérico positivo, que le pasa al método plug-in's **IPrintOemUni::CommandCallback** como un argumento dCmdCbID.	Requerido si se generan dinámicamente. No válido si se utiliza ***Cmd**.
***Cmd**	Cadena de texto conteniendo el comando escape.	Requerido si no se especifica ***CallbackID**.
***NoPageEject?**	TRUE o FALSE, Indicando si la ejecución del comando causa la eyección de la página. Utilizado sólo si ***Order** especifica la sección DOC_SETUP y si la impresión DUPLEX esta habilitada.	Opcional, si no se especifica, su valor es FALSE, significando que la ejecución del comando causa la eyección de la página.
***Order**	Número que representa el orden de la sección.	Válido solo con configuración de comandos.
***Params**	Lista de variables estándar que son pasadas al método plug-in's **IPrintOemUni::CommandCallback** en la estructura ESTRAPARAM.	Válido sólo si se especifica ***CallbackID**.

```
*Command: CmdStartDoc
{
   *Order: DOC_SETUP.1
   *Cmd: "<1B>@<0D1B>t<011B>6<1B>R<001B>x<011B>P"
}
*Command: CmdStartPage
```

```
{
   *Order: PAGE_SETUP.1
   *Cmd: "<0D>"
}
*Command: CmdEndJob
{
   *Order: JOB_FINISH.1
   *Cmd: "<0D>"
}
*RotateCoordinate?: FALSE
*RotateRaster?: FALSE
*RotateFont?: FALSE
*switch: Orientation
{
   *case: PORTRAIT
   {
      *TextCaps:
LIST(TC_OP_CHARACTER,TC_EA_DOUBLE,TC_IA_ABLE,TC_UA_ABLE,TC_RA_ABLE)
   }
   *case: LANDSCAPE_CC270
   {
      *TextCaps: LIST(TC_RA_ABLE)
   }
}
*CursorXAfterCR: AT_CURSOR_X_ORIGIN
*YMoveAttributes: LIST(SEND_CR_FIRST)
*MaxLineSpacing: 255
*XMoveThreshold: *
*YMoveThreshold: *
*XMoveUnit: 120
*YMoveUnit: 216
*Command: CmdXMoveRelRight { *Cmd : "<1B>\" %I{(DestXRel / 6) } }
*Command: CmdYMoveRelDown { *Cmd : "<1B>J" %c[0,255]{max_repeat((DestYRel / 2) )) }
*Command: CmdSetLineSpacing { *Cmd : "<1B>3" %c[0,255]{((LinefeedSpacing / 2) } }
*Command: CmdCR { *Cmd : "<0D>" }
*Command: CmdLF { *Cmd : "<0A>" }
*Command: CmdFF { *Cmd : "<0C>" }
*Command: CmdBackSpace { *Cmd : "<08>" }
*Command: CmdUniDirectionOn { *Cmd : "<1B>U<01>" }
*Command: CmdUniDirectionOff { *Cmd : "<1B>U<00>" }
*EjectPageWithFF?: TRUE
*switch: PaperSize
{
   *case: CUSTOMSIZE
   {
      *EjectPageWithFF?: FALSE
      *switch: InputBin
      {
      }
   }
}
*OutputDataFormat: V_BYTE
*OptimizeLeftBound?: FALSE
*CursorXAfterSendBlockData: AT_GRXDATA_END
```

```
*CursorYAfterSendBlockData: NO_MOVE
*DefaultFont: 1
*DefaultCTT: 1
*switch: Orientation
{
   *case: PORTRAIT
   {
      *DeviceFonts: LIST(1,2,4,5,6,7,8,9,15,17,18,19,21,22,23,24,
+                25,26,32,34,45,46,48,49,50,51)
   }
   *case: LANDSCAPE_CC270
   {
      *DeviceFonts: LIST()
   }
}
*Command: CmdBoldOn { *Cmd : "<1B>E" }
*Command: CmdBoldOff { *Cmd : "<1B>F" }
*Command: CmdItalicOn { *Cmd : "<1B>4" }
*Command: CmdItalicOff { *Cmd : "<1B>5" }
*Command: CmdUnderlineOn { *Cmd : "<1B>-<01>" }
*Command: CmdUnderlineOff { *Cmd : "<1B>-<00>" }
```

3.4 Análisis del código en lenguaje C que interviene en la realización del Controlador

Observamos que en nuestro proyecto particular, en nuestro directorio de trabajo, en este caso... \ENEP_A01, el único archivo de código fuente, aparte de los archivos de recursos, es uno cuyo nombre es CODE.C. Su ubicación es... \MINI\ENEP_A01\CODE.C, que sólo hace referencia a una línea:

```
#include      "..\modinit.c"
```

Que incluye el archivo MODINIT.C, cuya ubicación es... \MINI\MODINIT.C, que hace referencia a las siguientes lineas de código:

```
/********Module Header ************************************
 * modinit.c
 *      The generic minidriver module.  This is the DLL initialisation
 *      function,  being called at load time.  We remember our handle,
 *      in case it might be useful.
 *
 * HISTORY:
 *  17:37 on Fri 05 Apr 1991    -by-   Lindsay Harris   [lindsayh]
 *      Created it.
 *
 *  Copyright (C) 1992  Microsoft Corporation.
 *
*****************************************************************/

#include      <windows.h>


/*************** Function Header ******************************
 * bInitProc()
 *      DLL initialization procedure.  Save the module handle and
 *      initialise some machine dependent "constants".
 *
 * Returns:
 *   This function returns TRUE.
 *
```

```
 * HISTORY:
 *  15:59 on Wed 08 Jan 1992    -by-    Lindsay Harris   [lindsayh]
 *      Taken from rasdd's enabldrv.c
 *
*************************************************************/


BOOL bInitProc( hmod, Reason, pctx )
PVOID    hmod;
ULONG    Reason;
PCONTEXT pctx;
{
   UNREFERENCED_PARAMETER( hmod );
   UNREFERENCED_PARAMETER( Reason );
   UNREFERENCED_PARAMETER( pctx );

   return  TRUE;
}

#ifdef _GET_FUNC_ADDR

/*
 *    If the minidriver contains code called by RasDD, then we will need
 * to be initialised: we need some function addresses in RasDD. But these
 * cannot be statically linked since we do not know the path to rasdd.
 * Hence,  we export the following function,  which RasDD will call first.
 * This gives us the address of the RasDD functions available to us.
 */


/******************* Function Header *****************************
 * bSetFuncAddr
 *      Called by RasDD to pass in addresses needed by us to call into
 *      the available functions in Rasdd.
 *
 * RETURNS:
 *      TRUE if data is understable,  else FALSE.
 *
 * HISTORY:
```

```
 *  13:50 on Wed 20 May 1992  -by-   Lindsay Harris   [lindsayh]
 *     First version
 *
*****************************************************************/

BOOL
bSetFuncAddr( pntmd_init )
NTMD_INIT   *pntmd_init;
{
   /*  Check that the data format is the type we understand.  */

   if( pntmd_init->wSize < sizeof( NTMD_INIT ) ||
      pntmd_init->wVersion < NTMD_INIT_VER )
   {
      return   FALSE;        /* Can't afford to monkey around */
   }

   /*  Data is GOOD,  so copy it to our global data  */

   ntmdInit = *pntmd_init;

   return  TRUE;
}

#endif
```

Como podemos observar, incluye a la librería de cabecera del SDK y de cualquier lenguaje para Win32, llamada "WINDOWS.H", que como el lector que haya desarrollado código para Windows sabrá, es necesaria para cualquier aplicación en Win16 y Win32. También posee las rutinas:

BOOL bInitProc(PVOID hmod, ULONG Reason, PCONTEXT pctx)

Que es el procedimiento de inicialización de cualquier DLL para Windows. Graba el handle del módulo e inicializa alguna constantes dependientes del dispositivo. Esta función siempre regresa verdadero. Para mayor información consultar la Guia del Microsoft Win32 SDK.

BOOL bSetFuncAddr(NTMD_INIT *pntmd_init)

Sí el ***minidriver*** contiene código llamado por el RasDD, entonces necesitaremos sea inicializado: Necesitamos algunas direcciones de funciones en el RasDD. Pero estas no pueden ser enlazadas estáticamente, ya que no conocemos la ruta que ocupara el RasDD, de manera que entonces exportamos esta función, que el RasDD llamará primero, lo que entonces no da la dirección de las funciones del RasDD disponibles para nosotros. Básicamente es llamada por el RasDD para pasarle las direcciones necesarias por nosotros para llamar a las funciones disponibles en el RasDD. Retorna Verdadero si los datos son entendibles, sino, Falso. Para mayor información, ver el Microsoft Win32 SDK

También se tienen algunos archivos, principalmente de cabeceras, como:

MAPFILE.H
NTMINDRV.H
MINDRVRC.H
WINDDIUI.H
WINFONT.H
WINSPLP.H

Entre otra varias funciones...

3.5 Creación y Compilación de un archivo ejecutable que servirá como Controlador de dispositivo

Para el proceso de preparación intervienen varios archivos como el archivo DIRS, MAKEFILE, SOURCES, SETENV.BAT, los cuales se tratarán a continuación. La construcción se realiza mediante las llamadas a los programas BUILD y NMAKE

3.5.a) Archivo Dirs y MakeFile

El archivo DIRS debe encontrarse directamente desde la raíz de nuestro directorio de trabajo, es decir, en el directorio ***D:\NTDDK***. Este archivo contiene sólo la directiva DIRS = ruta_de_trabajo, donde ruta_de_trabajo es la ruta donde se encuentran los archivos fuente desde donde se compilarán para generar el ejecutable del controlador.

3.5.b) Lista de Macros utilizadas en el archivo Sources

La construcción de código de controladores de dispositivo se realiza a través de la utilidad de compilación BUILD, proporcionada con el DDK. Esta utilidad hace referencia a un archivo llamado SOURCES, ubicado en el directorio de código fuente de nuestro controlador. Este archivo contiene una serie de definiciones MACRO, que son reconocidas por BUILD. La siguiente lista contiene las macros que normalmente contiene un archivo SOURCES:

TARGETNAME
: Especifica el nombre de la librería que se construye. Se excluye la extensión.

TARGETPATH
: Especifica el nombre del directorio que es destino de los elementos creados por BUILD (EXE, DLL, LIB) Los archivos OBJ son creados en el directorio \obj.

TARGETTYPE
: Especifica el tipo de producto que se construye. Típicamente es LIBRARY o DYNLINK para los DLL, Se pueden especificar otros valores.

INCLUDES
: Lista de rutas para los archivos INCLUDE durante la compilación.

SOURCES
: Lista de archivos fuente (excepto el archivo que contiene el ***main()***) con sus extensiones. Los elementos son separados por espacios o TABS.

UMTYPE
: Especifica el tipo de producto que se construye:

Windows	Especifica Win32 (modo usuario)
NT	Especifica modo kernel.
console	Especifica una aplicación de consola de 32 bits

UMTEST
: Lista de archivos fuente conteniendo a la función ***main()***. Estos son especificados sin extensión y son separados por asteriscos.

UMAPPL
: Lo mismo que UMTEST, con la diferencia de que si se usa UMTEST, se necesita dar la lista de los nombres de los archivos por ser construidos desde la línea de comandos a BUILD.EXE. Si se usa UMAPPL, no es necesario especificar nada en la línea de comandos, y BUILD.EXE automáticamente construirá los ejecutables.

UMAPPLEXT

Especifica la extensión del nombre de archivo (COM, SCR, etc.) si se desean archivos de imagen distintos a la extensión EXE.

TARGETEXT

Especifica la extensión de nombre de archivo, tal como CPL, si es que se desea otro distinto a la extensión DLL.

UMLIBS

Contiene una lista de los nombres de las librerías para ser enlazadas a los archivos especificados por UMTEST o UMAPPL. La librería generada por SOURCES debe estar incluida aquí. Los elementos están separadas por espacios o TABS. Los nombres de rutas deben ser absolutos.

Los nombres de rutas UMLIBS necesitan ser definidos de manera especial. La razón es que BUILD proporciona un mecanismo para la construcción de productos destino para diversas plataformas. La ruta destino debe estar como sigue:

%TARGETPATH%\<cpu_type>

Donde TARGETPATH esta definido en el archivo SOURCES. Y CPU_TYPE especifica el CPU para el que será construido. Por ejemplo, para un TARGETPATH que deseemos sea OBJ y un hardware i386 y MIPS, el producto por construir debe quedar direccionado de la siguiente manera:

obj\i386 ***// build products for i386***
obj\mips ***// build products for MIPS***

Debido a esta convención, las entradas UMLIBS deben especificar nombres de librerías en la siguiente forma:

<targetpath>*\<library_name>

Donde TARGETPATH es idéntico al definido en el archivo SOURCES y LIBRARY_NAME es el nombre completo de la librería a ser enlazada a la aplicación. El * es automáticamente reemplazado por el destino del CPU actual durante el proceso de construcción.
Es posible construir las aplicaciones en una unidad lógica distinta a la que contiene el árbol DDK mediante el uso de la variable de ambiente BASEDIR. El DDK la define a ser \DDK_ROOT por defecto. Si se tiene el árbol del directorio fuente en una unidad diferente de los fuentes del DDK, es necesario preparar la variable de

ambiente BASEDIR a las librerías que se necesitan enlazar. Un a forma de hacer esto se muestra a continuación:

$(BASEDIR)\lib*\base.lib

En adición a los archivos fuentes, un archivo llamado MAKEFILE es requerido en cada directorio que contenga código fuente a ser construido. Este archivo no debe variar de un directorio fuente a otro y nunca debe ser alterado.
Para construir un ejecutable en particular debe cambiar el directorio contenido en el archivo SOURCES e invocar a BUILD. BUILD automáticamente construye la librería especificada en SOURCES, pero sólo construye los ejecutables sólo si se le ha especificado explícitamente. (A menos que se use UMAPPL).

Por ejemplo, si un archivo SOURCES tiene definida una librería llamada MILIB (TARGETNAME) y un ejecutable (UMTEST) llamado y MIEJEC. Ambos productos deben ser construidos para i386 con la siguiente invocación:

build -386 miejec

Si MIEJEC no es especificado, solo MILIB será construido por defecto.
De otra manera, si se ha definido una librería llamada MILIB (TARGETNAME y un ejecutable (UMAPPL) llamado MIEJEC. Ambos productos deben ser construidos para i386 con la siguiente invocación:

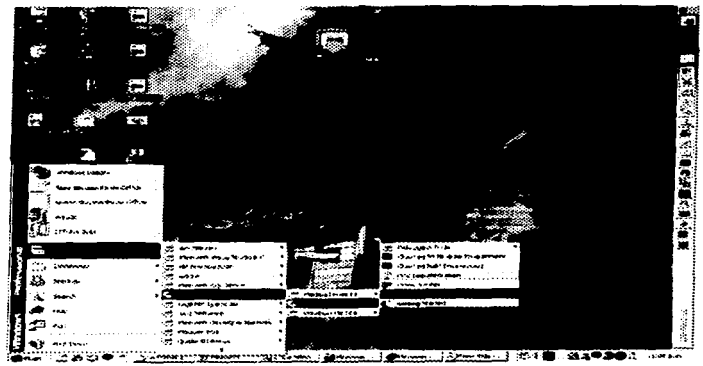
build -386

Para un programa simple que consista de un solo archivo fuente, la librería no es requerida. En esta situación el archivo SOURCES se define como sigue:

- Colocar TARGETNAME y TARGETPATH como si se fuese a generar la librería.
- Colocar TARGETTYPE a UMAPPL_NOLIB.
- Colocar SOURCES a nada.
- No incluir la librería TARGETNAME en UMLIBS.
- Invocar a BUILD, especificando el archivo listado en UMTEST.
- BUILD generará el archivo ejecutable.

3.5.c) Construcción del ambiente de desarrollo mediante el archivo por lotes Setenv.bat

Para la construcción de este ambiente se pueden seguir dos formas diferentes. Una forma manual mediante la introducción manual de las variables de ambiente en la consola del sistema (ventana de DOS), y otra es mediante la ejecución de archivos de procesamiento por lotes, donde estén incluidas estas variables. Se realizan desde la consola del sistema debido a que las variables de entorno son acciones que sin necesidad de ejecutar elementos gráficos, realizan la operación de instalar definiciones de ambiente con mucho mayor eficiencia. Es importante hacer notar que aún en los actuales entornos gráficos, para la creación de software controlador de dispositivos se sigue utilizando la programación un tanto de bajo nivel en lenguaje ensamblador para las entradas de las rutinas, así como medio y alto nivel en lenguaje C++ para las rutinas y funciones GDI restantes. Cabe señalar que la construcción del ambiente de desarrollo debe ser construido antes de construir los ejecutables y las librerías del controlador.



Al instalar correctamente el SDK y DDK, se crean subgrupos de trabajo dentro del grupo *Accesorios* accesibles desde el botón de INICIO del escritorio de Windows. En particular, dentro del subgrupo ***Windows 2000 DDK***, existen 3 elementos de archivo por lotes listados como sigue:

- Checked 64 bit Build Environment
- Checked Build Environment
- Free Build Environment

Los tres archivos por lotes elementos ejecutan el archivo por lotes llamado SETENV.BAT que se encuentra colocado en el respectivo directorio \DDK\BIN\SETENV.BAT. El segundo elemento ejecuta la línea:

SETENV D:\DDK CHECKED

Y el tercer elemento ejecuta: SET_ENV D:\DDK FREE.

Este archivo realiza una serie de establecimientos de varias variables de ambiente mediante asignaciones, tanto directas como mediante MACROS. En este archivo por lotes, la sustitución de nombres por valores, se realiza mediante el formato %VARIABLE%, donde la variable encerrada entre signos de porcentaje (%) es la variable por asignar.

Como se puede observar, dentro del archivo SETENV.BAT se realizan una serie de inicializaciones mediante saltos condicionales hacia alguno de los apartados mencionados. Los nombres de los apartados tienen antepuestos el símbolo de dos puntos ":". Los siguientes son los apartados que aparecen en el archivo por lotes SETENV.BAT:

- **Setbasedir**
- **Setenv**
- **Alpha**
- **Ppc**
- **Mips**
- **I386**
- **Envtest**
- **Free**
- **Checked**
- **Done**
- **Noddkdrive**
- **Cpuerror**
- **No_mstools**
- **Usage**
- **End**

Haciendo una análisis de este archivo por lotes, tenemos la siguiente partición de este archivo. Para un mejor estudio, hemos numerado las líneas:

```
1- @echo off

2- if "%1"=="" goto usage
3- rem This will put the SDK headers & libs first in the search path.

4- if "%MSTOOLS%"=="" goto no_mstools
5- call %MSTOOLS%\setenv %MSTOOLS%

6- if "%BASEDIR%"=="" goto setbasedir
7- if NOT "%BASEDIR%"=="%1" goto setbasedir

8- if "%DDKBUILDENV%"=="" goto setenv
9- if NOT "%DDKBUILDENV%"=="%2" goto envtest
10- goto done
```

En la línea 1, tenemos que sólo se apaga la impresión de la ejecución de las instrucciones en el monitor. La línea 2 brinca hacia el apartado USAGE si se ha tecleado sólo el nombre SETENV y ENTER, es decir, si no se ha tecleado ningún parámetro adicional a la llamada de archivo por lotes. La línea 4 brinca hacia el apartado NO_MSTOOLS, si la variable MSTOOLS no está definida. Esta variable la inicia el sistema si previamente se ha instalado el Win32 SDK. La línea 5 de manera recursiva manda a llamar al archivo por lotes SETENV.BAT colocado en el directorio C:\MSTOOLS con la variable de ambiente %MSTOOLS%. La línea 6 brinca hacia el apartado SETBASEDIR si la variable de entorno BASEDIR no se encuentra definida. La línea 7 brinca a el apartado SETBASEDIR, si el valor de la variable BASEDIR no es igual al valor de ***%1***, que es el primer parámetro tecleado después de la llamada a SETENV.BAT. La línea 8 brinca a el apartado SETENV si la variable DDKBUILDENV no se encuentra definida. La línea 9 brinca a el apartado ENVTEST, si el valor de DDKBUILDENV no es igual al valor de ***%2***, donde el valor de ***%2*** es el segundo parámetro tecleado al mandar llamar a SETENV.BAT. El primer parámetro es el mismo nombre del la llamada. Este valor puede ser FREE ó CHECKED. Y la línea 10 finaliza la ejecución del archivo por lotes. Ejemplo, si se manda llamar a: D:\DDK\BIN\SETENV D:\DDK FREE. El Valor D:***DDK*** es ***%1***, y ***FREE*** es ***%2***.

Cuando se inicia el sistema, las variables que ya se encuentran definidas de acuerdo al sistema, al SDK y al DDK, son:

BASEDIR = D:\DDK

MSTOOLS = D:\MSTOOLS
TMP = C:\TMP
PROCESSOR_ARCHITECTURE = x86
CPU = i386
DDKDRIVE = D:

De acuerdo a estas variables definidas, es como la ejecución de este archivo sigue una ruta definida por las variables de entorno. Cabe señalar que estos archivos SETENV.BAT no deben ser modificados, ya que previamente se encuentran preparados para soportar la mayoría de condiciones. El objeto de ejecutar este archivo es para definir las variables de ambiente restantes, de acuerdo al entorno que se quiera preparar, ya sea FREE o CHECKED, para que al momento de ejecutar el programa BUILD, ya se encuentren definidas las variables necesarias. La ejecución de este archivo debe ser antes de la compilación y enlace de los archivos, y puede ser antes o después de la creación de los archivos del minidriver en el UNITOOL. Las variables que se definen en este archivo, entre otras son:

NTMAKEENV = %BASEDIR%\inc
BUILD_MAKE_PROGRAM = nmake.exe
BUILD_DEFAULT = -ei -nmake -i
BUILD_DEFAULT_TARGETS = 386 (Depende de los parámetros)
Path = %path%;%BASEDIR%\bin
Cpu = i386
NTBUILDENV = free | checked (Depende de los parámetros tecleados)
C_DEFINES = (Depende de los parámetros)
NTDBGFILES = (Depende de los parámetros)
NTDEBUG = (Depende de los parámetros)
NTDEBUGTYPE = (Depende de los parámetros)
MSC_OPTIMIZATION = (Depende de los parámetros)

Entre paréntesis aparece que depende de los parámetros, haciendo referencia a que depende de la arquitectura del procesador para el cual se desarrolla el controlador, así como si se llama a SETENV.BAT con el parámetro FREE o CHECKED. Las variables establecidas por este archivo por lotes desaparecen al reiniciar el equipo de cómputo.

3.5.d) Utilización de Build para la construcción de los archivos ejecutables

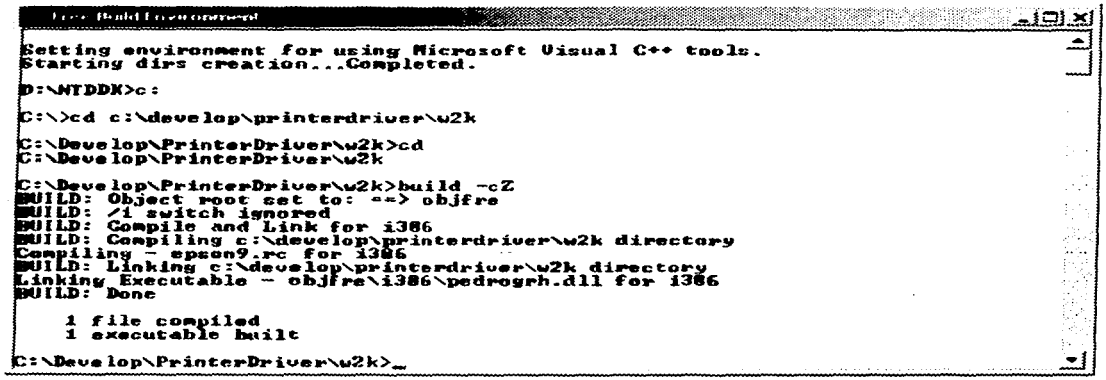
El archivo BUILD.EXE es un programa proporcionado con el Win 32 DDK, y se encuentra en ***D:\DDK\BIN\BUILD.EXE***. Este programa utiliza un conjunto de reglas y archivos de proyectos que le especifican como debe ser creador el controlador. Estos archivos son DIRS que especifica que directorios en que subarboles contienen el código fuente que será construido, SOURCES que especifica que archivos son requeridos para construir el controlador. Si existe código fuente en distintos directorios, puede ser necesario crear un archivo DIRS por cada directorio. MAKEFILE y SETENV.BAT.

Este programa BUILD.EXE manda llamar a la utilidad NMAKE por cada archivo fuente listado en SOURCES. NMAKE usa a MAKEFILE para generar la dependencia y la lista de comandos. El archivo MAKEFILE.DEF, define las banderas de construcción para el compilador y enlazador y simplifica la creación de proyectos para plataformas independientes de dispositivo. Una vez que todos los archivos se tienen listos, lo que sigue es ejecutar BUILD.EXE, lo cual comenzará la compilación de los distintos fuentes, se evaluarán las MACROS en MAKEFILE.DEF, se crearán y enlazaran los distintos objetos. Si todo está correcto, esto deberá generar él o los programas ejecutables de nuestro controlador.

El Programa BUILD, también genera archivos históricos de errores, los cuales se utilizan en distintos estados de la construcción

Build.log Contiene un histórico de los comandos invocados vía NMAKE.
Build.wrn Una lista de avisos (warnings) generados durante la construcción
Build.err Una lista de errores generados durante la construcción.

Algunos avisos del compilador y enlazador son filtrados por la utilidad BUILD. Estos mensajes filtrados no aparecen en las estadísticas reportados por BUILD, pero se encuentran en los archivos BUILD.LOG y BUILD.WRN.



Inmediatamente después de ejecutar el comando SETENV.BAT desde el menú DDK Free Build Environment, Podemos observar que después queda abierta la ventana de consola, que a su vez queda en el directorio ***D: \NTDDK***. Deberemos cambiarnos a la ruta donde se encuentran nuestros archivos por compilar: ***C:\Develop\PrinterDriver\w2k*** donde se deberá ejecutar el comando BUILD, como se muestra en la siguiente figura donde se muestra la ventana de consola completa, donde se ha ejecutado el comando **BUILD -cZ**, y podemos observar que se ha compilado y enlazado para la plataforma i386, además se han enlazado algunas librerías particulares del DDK, así como del Microsoft Visual C++, y se ha realizado una búsqueda de archivos RC por compilar, mismos que ha compilado y enlazado bajo el directorio ***C:\Develop\PrinterDriver\w2k\ objfre\i386*** Cabe hacer mención que el archivo destino por crear ENEP_A01.DLL no debe existir, de lo contrario, el ejecutable no se creará de nuevo. Una vez realizado esto, los programas ejecutables resultantes son construidos en el directorio ***C:\Develop\PrinterDriver\w2k\ objfre\i386,*** Si la construcción es exitosa, se construirá un archivo DLL, un archivo RES y un archivo INF.

3.5.e) Caso Particular del Controlador PEDROGRH para Archivo DIRS

Para nuestro caso en particular, uno de los archivos DIRS contiene sólo la directiva

```
DIRS= \
   w2k
```

Otro ejemplo sería:

```
DIRS= \
   network \
   print \
   smartcrd \
   vdd \
   video \
   preview
```

3.5.f) *Caso Particular del Controlador PEDROGRH para Archivo MAKEFILE*

Este archivo se encuentra en el mismo directorio donde se encuentran los archivos fuentes, no tiene extensión y sólo recibe el nombre de MAKEFILE. Este archivo sólo ejecuta la directiva

!INCLUDE $(NTMAKEENV)\makefile.def

Cabe señalar que este archivo no debe ser modificado. En esta línea lo que se hace es incluir el archivo llamado MAKEFILE.DEF, que se encuentra en C: \NTDDK\INC\MAKEFILE.DEF, ya que la variable NTMAKEENV = D: \NTDDK. El archivo MAKEFILE.DEF es el archivo estándar para la creación de los componentes del proyecto de Windows. Incluye a los siguientes archivos:

.\sources. El desarrollador proporciona este archivo donde se define TARGETNAME, TARGETPATH, TARGETTYPE y SOURCES como macros para el compilador y enlazador.

obj_objects.mac Construido por BUILD.EXE a partir de \sources.

3.5.g) *Caso Particular del Controlador PEDROGRH para Archivo SOURCES*

```
!IF 0
Copyright (C) 1997 - 1999 Microsoft Corporation
!ENDIF

TARGETNAME=pedrogrh
TARGETPATH=obj
TARGETTYPE=DYNLINK
TARGETLIBS=

DLLBASE=0x20000000
INCLUDES=D:\NTDDK\src\print\inc
RESOURCE_ONLY_DLL=1

SOURCES= \
      epson9.rc
UMTYPE=windows
MISCFILES= \
        epcomp9.GPD \
        epd8000.GPD \
        pedrogrh.GPD \
        eplx300.GPD \
        epx1050p.GPD
```

Que para nuestro caso en particular conforma la totalidad del archivo SOURCES

Capitulo 4

Particularidades de Funcionamiento del Controlador de Impresión.

4.1 Comandos Escape y Soporte de Caracteres ASCII

En este caso, y como se mencionaba con anterioridad, cada dispositivo de impresión tiene su propia familia de Comandos Escape, puesto que son estos los que controlan el funcionamiento a nivel hardware. Para el caso de nuestro ejemplo en particular estamos utilizando una Impresora de Matriz de Puntos Star NX1040, la cual tiene dos modos de emulación:

Modo Standard/Epson
Modo IBM

En el modo Standard/Epson este dispositivo emula las funciones de la Epson LX-810 o EX800 para el caso de impresión a color. En el modo IBM emula a la IBM Proprinter III. Los comandos adicionales son incluidos como un superconjunto de estas emulaciones.

En este dispositivo la emulación es cambiada por el movimiento del EDS (Electronic Dip Switch) A-1. Cuando este en ON, significa la selección del Modo Standard/Epson, mientras que en OFF significa Modo IBM.

De manera adicional, cuando el EDS A-2 esta en ON, la impresora automáticamente cambia su emulación a un control por software

TABLA DE COMANDOS ESCAPE PARA STAR NX-1040

- Comandos de Control de Impresión
- Comandos de Control de Fuentes
- Comandos de Conjunto de Caracteres
- Comandos de Tamaño y Densidad de Caracteres

A continuación mostraremos una lista de tablas de algunos de los más importantes comandos de impresión, de control, de posicionamiento, selección etc. De manera que no se listan todos los posibles comandos por ser demasiados. Si el lector está interesado en conocer la totalidad de comandos puede referirse al Manual

de Usuario de la Impresora Star NX 1040 o al Manual Epson LX-810 para B/N o EX800 para el caso de impresión a color. También puede referirse al Manual de IBM Proprinter III para el caso de impresión a color.

En la siguiente lista el apartado Modo se refiere al modo en el cual el comando es reconocido, y los tres siguientes apartados se refieren a sus equivalentes es ASCII, Decimal y Hexadecimal.

4.1.a) Comandos de control

Inicialización de la Impresora

Modo	ASCII	Decimal	Hexadecimal
Std	<Esc> "@" n	27 @	1B 40

Este comando permite reinicializar la impresora, limpiando el buffer de impresión y regresando la configuración a la de por defecto.

Selección de la Calidad de Impresión

Modo	ASCII	Decimal	Hexadecimal
Std	<Esc> "x" n	27 120 n	1B 78 01

Este comando permite cambiar la calidad de impresión de Draft que equivale a n=0, a Near Letter Quality, que equivale a n=1. Solo trabaja en Modo Estándar.

Modo	ASCII	Decimal	Hexadecimal
IBM	<Esc> "[" "d" <1> <0> n	27 91 100 1 0 n	1B 5B 64 01 00 n

Este comando permite cambiar la calidad de impresión para el Modo IBM de la siguiente manera:

N	Calidad de Impresión
0	Sin Cambio
1 – 127	Draft
128 – 254	Near Letter Quality
255	Retorno al Modo EDS

4.1.b) Comandos de control de Fuentes

Selección de una Fuente Near Letter Quality (NLQ)

Modo	ASCII	Decimal	Hexadecimal
Ambos	<Esc> "k" n	27 107 n	1B 6B n

Donde n puede seleccionar:

0 Courier
1 Sanserif
2 Courier
7 Orator with small capitals
8 Orator with lower case

Selección de Caracteres itálicos

Modo	ASCII	Decimal	Hexadecimal
Std	<Esc> "4"	27 52	1B 34

Finalización de caracteres itálicos

Modo	ASCII	Decimal	Hexadecimal
Std	<Esc> "5"	27 53	1B 35

Impresión Enfatizada

Modo	ASCII	Decimal	Hexadecimal
Ambos	<Esc> "E"	27 69	1B 45

Fin de Impresión Enfatizada

Modo	ASCII	Decimal	Hexadecimal
Ambos	<Esc> "F"	27 70	1B 46

Impresión en Doble Remarcado

Modo	ASCII	Decimal	Hexadecimal
Ambos	<Esc> "G"	27 71	1B 47

Fin de Impresión en Doble Remarcado

Modo	ASCII	Decimal	Hexadecimal
Ambos	<Esc> "G"	27 72	1B 48

4.1.c) Comandos de control de Conjunto de Caracteres

Selecciona el Conjunto de Caracteres Estándar

Modo	ASCII	Decimal	Hexadecimal
Ambos	<Esc> "t" <0>	27 116 0	1B 74 00

Selecciona el Conjunto de Caracteres IBM

Modo	ASCII	Decimal	Hexadecimal
Ambos	<Esc> "t" <1>	27 116 1	1B 74 01

Selecciona el Conjunto Internacional de Caracteres

Modo	ASCII	Decimal	Hexadecimal
Std	<Esc> "R" n	27 82 n	1B 52 n

N Puede ser alguno de los siguientes valores:

0 USA
1 France
2 Germany
3 England
4 DenMark I
5 Sweden
6 Italy
7 Spain I
8 Japan
9 Norway
10 Denmark II
11 Spain II
12 LatinAmerica
13 Korea
14 Irish
64 Legal.

Selecciona la Página de Códigos de IBM

Modo	ASCII	Decimal	Hexadecimal
IBM	<Esc> "[" "T" <4> <0> <0> <0> n1 n2	27 91 84 4 0 0 0 n1 n2	1B 5B 54 04 00 00 00 n1 n2

N1	n2	Página de Códigos
1	181	#437 USA

3	82	#850 Multi lingual
3	92	#860 Portuguese
3	93	#861 Icelandic
3	95	#863 Canadian French
3	97	#865 Nordic

4.1.d) Comandos de control de tamaño y densidad de caracteres

Densidad Pica (Pica Pitch)

Modo	ASCII	Decimal	Hexadecimal
Std	<Esc> "P"	27 80	1B 50
IBM	<DC2>	18	12

Densidad Elite (Elite Pitch)

Modo	ASCII	Decimal	Hexadecimal
Std	<Esc> "M"	27 77	1B 4D
IBM	<Esc> ":"	27 58	1B 3ª

Impresión Condensada

Modo	ASCII	Decimal	Hexadecimal
Ambos	<SI>	15	0F
Ambos	<Esc> <SI>	27 15	1B 0F

Cancelación de Impresión Condensada

Modo	ASCII	Decimal	Hexadecimal
Ambos	<DC2>	18	12

Selección de Fuente y Densidad

Modo	ASCII	Decimal	Hexadecimal
IBM	<Esc> "I" n	27 73 n	1B 49 n

N	Fuente y Densidad
0	10 CPI Draft
1	12 CPI Draft
2	10 CPI Near Letter Quality
3	10 CPI Near Letter Quality
4	10 CPI Draft Download
5	12 CPI Draft Download

6 10 CPI Draft Double Strike
7 10 CPI Near Letter Quality Download
11 10 CPI Italic Near Letter Quality
15 10 CPI Italic Near Letter Quality Download

Selección de Modo e Impresión Maestro

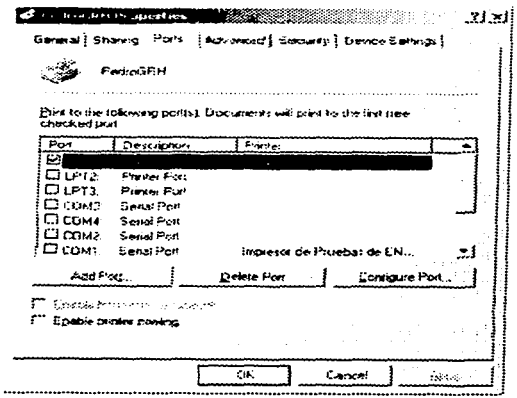
Modo	ASCII	Decimal	Hexadecimal
Std	<Esc> "!" n	27 33 n	1B 21 n

Función	n
Underline	128
Italic	64
Expanded	32
Double Strike	16
Emphatized	8
Condensed	4
Proportional	2

4.2 Funcionamiento del Controlador en Operación

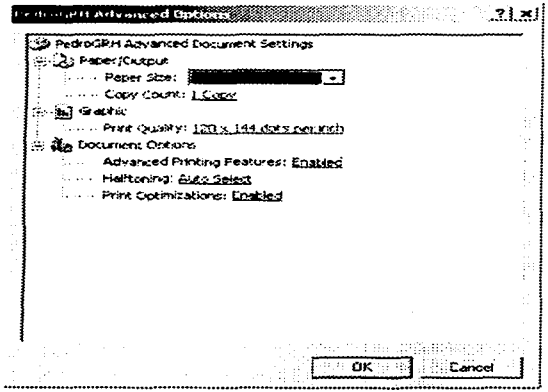
En esta sección mostramos algunas imágenes del funcionamiento del controlador de impresión bajo Windows NT, y como editor el Microsoft Word.

Primeramente tenemos el cuadro de diálogo *Imprimir*, donde se muestra el nombre del impresor que se utilizará, que en este caso es el Impresor de Pruebas de ENEP Aragón, así como también su cuadro de diálogo *Propiedades*, donde se muestran sus respectivas propiedades de documento.



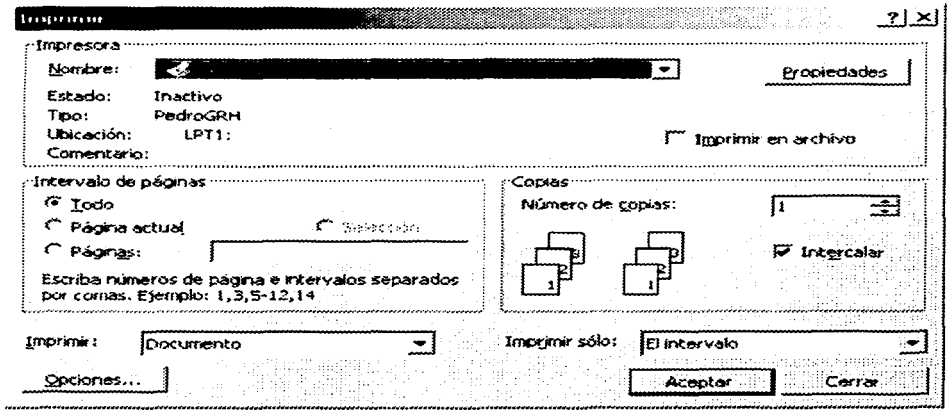
En la siguiente imagen podemos observar que el cuadro de diálogo Propiedades de Documento de Impresor de Pruebas de ENEP Aragón, es muy similar a cualquier cuadro de diálogo de propiedades de algún otro controlador de impresora de algún fabricante conocido en el mercado comercial.

Podemos observar, que el controlador de dispositivo de impresión ha sido satisfactoriamente instalado y se puede observar desde el Panel de Control en el apartado Impresoras.



Donde podemos observar, algunas características que son comunes a todos los controladores, como lo es su ventana de estado, llamada Impresor de Pruebas de ENEP Aragón, que a su vez posee un menú que es común a cualquier dispositivo de impresión tipo raster en Windows 2000

Como se puede observar, El elemento de menú Impresora/Propiedades, también sirve de entrada para el cuadro de diálogo Propiedades mostrado anteriormente, donde es posible configurar algunas características como la orientación del papel, el número de copias, el tamaño del papel, resolución, etc.



Capitulo 5

Conclusiones.

Conclusiones

Actualmente nos encontramos ante demasiados cambios dentro de la geografía informática. Cambios que propician de manera natural la labor de investigación y desarrolo, y nuestro país no puede ni debe escapar ante tales detonantes.

Es conocido por todos, que en cuanto a aspectos tecnológicos, existe una globalización informática, la cual se vierte de las potencias tecnológicas, hacia el resto del mundo, y hoy por hoy, el desarrollo de sistemas, como parte de esta globalización experimenta la creación de grupos de desarrollo, los cuales se han dado a la tarea de investigar mas alla de lo que los grandes corporativos presentan dentro de las líneas de negocios.

Lo vemos frecuentemente en los grupos mundiales de investigación en el tema, que si bien pueden trabajar para algún corporativo, tienen una insistencia en la investigación sin un proposito definido de lucro.

La creación de un Controlador de Dispositivo de Impresión, obedece a una de las miles de ramificaciones que esta industria ha generado. La investigación en cuanto a este rubro, se ha vertido hacia las compañias fabricantes de hardware del lejano oriente. Estas empresas, se han vuelto líderes en la fabricación masiva de los distintos dispositivos existentes, mas no en la investigación. Si bien es cierto que un porcentaje de esta se realiza en los paises en cuestión, el grueso de la investigación de innovación se realiza en Estados Unidos y Europa, donde como palpables ejemplos tenemos que los grandes corporativos de software, tienen sus sedes de investigación y desarrollo.

La labor de investigación de este controlador de dispositivo obedece a una necesidad local, de adaptar y reutilizar la funcionalidad de los impresores, o de otros dispositivos, a necesidades específicas. Por ejemplo, ciertos controladores que ya existen en la industria no manejan toda la funcionalidad del dispositivo, situación que hace necesaria la generación de un nuevo dispositivo. Esta

investigación sirve como apoyo para otros grupos tecnológicos. Por ejemplo, si existiera en nuestro país una industria de manufactura de dispositivos físicos, esta se debiera apoyar en grupos de investigación como los descritos, para adaptarse a las tendencias de mercado. Digo adaptarse, porque es conocido que los sistemas operativos de las distintas compañías, definen el rumbo y crecimiento de la industria del hardware.

Por poner un ejemplo, las necesidades de los sistemas desarrollados Microsoft, IBM, Apple Macintosh, establecen y dirigen las necesidades futuras, que tendrán que cumplir los fabricantes de componentes de hardware como Intel, Texas Instruments, AMD, asi como integradores de hardware como Compaq, Dell, etc.

Nuestro país ya esta lejos de participar activamente en el juego tecnológico antes descrito, pero esta a tiempo de integrarse en los grupos de investigación en cuanto a estrategias y filosofías se refiere, de integrarse a la investigación de la ingeniería del software, donde se ha colocado como lider de los paises latinoamericanos.

Es ahí donde entra el papel de las distintas universidades que lo han hecho posible. La creación de grupos de investigación y desarrollo en las distintas entidades académicas de nuestro país se ha convertido en una muy agradable actividad que reune profesionales del tema de distintos puntos geográficos.

Este trabajo, como resultado de una necesidad local, cumplió con las expectativas que le dieron vida y se logró adaptar a su objetivo. Espero que este trabajo aporte con algo a esta corriente de investigación y desarrollo, para poder lograr, cada vez una mejor calidad de vida y de pensamiento.

Glosario de Términos

Animación: Desplegado de una serie de imágenes gráficas, simulación de movimiento.

Applet: Una pequeña aplicación, generalmente de un sólo propósito que es iniciada por alguna otra aplicación, Por ejemplo, las diferentes opciones del Panel de Control son conocidas como applets.

Bitmap: Mapa de bits estándar.

Brush Origin (Pincel Orígen): La coordenada de un pixel en una superficie del dispositivo de impresión alineado al pixel superior izquierdo del patrón de relleno.

CMYK: El espacio de color que frecuentemente es implementado en las impresoras. El acronimo representa Cyan, Magenta, Yellow y Black.

Color Index (Indice de Colores): Un índice a un arreglo de RGB.

CPSUI (Common Property Sheet User Interface): Es un módulo proporcionado por el sistema que administra todos los aspectos de la hoja de propiedades de una aplicación, incluyendo las páginas definidas por controlador. Específicamente es responsable de la creación y destrucción de la hoja de propiedades de la aplicación más alguna página especifica de dispositivo que alguna aplicación pueda requerir.

CTT (Character Translation Table): Es utilizado en los minidrivers de Windows NT y traducido en Windows NT a formato RLE.

DDB (Device Dependent BitMap): Una estructura de datos describiendo una imagen de bit, especificando ancho y alto de una región particular en pixeles, el ancho de un arreglo que mapea entradas desde la paleta del dispositivo a pixeles, así como el formato de color del dispositivo en términos de planos de color y bits por pixel.

DDI (Device Driver Interface): Un conjunto de funciones implementadas por controladores gráficos y utilixados por el GDI para comunicarse con el controlador.

Despooling: La conversión de registros independientes del dispositivo a registros dependientes de dispositivo.

DIB (Device Independent Bitmap): La información en el mapa de bits es colocada en estructuras de datos conteniendo información como el formato de color, resolución y paleta de colores para el dispositivo en el cual la imagen aparece, un arreglo de bits que mapea tripletas de pixeles RGB, y un identificador de compresión de datos, lo cual indica si algún esquema de compresión de datos fue usado para reducir el tamaño de un arreglo de bits. Un mapa de bits GDI que puede estar construido precisamente en diferentes dispositivos porque contiene una tabla de color que pueda ser leída por el constructor del controlador de dispositivo. Estos son mapas de bits en formato estándar creados y soportados por el GDI a través de EngCreateBitmap.

Dithering: Una técnica usada para crear la ilusión de variación de sombras de grises en un monitor o impresor monocromático, o colores adicionales en un monitor o impresor de color. Dithering depende en el tratamiento de áreas de una imagen como grupos de dots que son coloreados en diferentes patrones. Similar a los medios tonos en fotografías.

DLL (Dynamic Link Library): Las DLL ó Librerías de enlace dinámico, son código ejecutable encapsulado, que se accesa y enlaza en tiempo de ejecución. Esto permite que los programas de aplicación, sean más pequeños, ya que es código particular, mientras que el código general (DLL), puede ser reutilizable por varias aplicaciones

EMF (Enhanced Metafiles): Los EMF es un tipo de archivo de encolado (spooling) usado por el spooler del Windows NT. Esos archivos spool reducen el tiempo gastado entre donde la aplicación hace un requerimiento de impresión y cuando el control es retornado. Esto es posible porque las llamadas a las funciones GDI producen los objetos de salida gráfica que son colocados directamente en el EMF. El tiempo consumido por la ejecución es recobrado mas tarde, en residente, cuando el archivo spool es reejecutado.

Formato Estándar de Mapa de Bits: Plano simple, pixel empacado (donde los datos para cada pixel es colocado de manera contigua) formato de mapa de bits creado y soportado por el GDI a través de ***EngCreateBitmap***. Cada línea de rastreo del mapa de bits es alineada en un elemento de 4 bytes. Los mapas de bits estandar codifican información a una razón de 1, 4, 8, 16 o 32 bits por pixel.

GDI (Graphic Device Interface): El GDI es dividido en dos piezas. El GDI Win 32 es un API modo usuario utilizado por las aplicaciones Win 32 que requieren soporte gráfico. GDI modo kernel hace una interface directamente con los controladores gráficos modo kernel. El GDI modo kernel exporta varios servicios y funciones que pueden ser llamadas por los controladores de dispositivo para desempeñar un host de operaciones de dibujo y gráficos relacionados. Esto elimina la necesidad para los controladores gráficos de implementar mucha funcionalidad de las funciones requeridas.

La librería Win 32 GDI que es directamente accesible a aplicaciones Win32 puede encontrarse en la librería GDI32.DLL. Este campo llama a las funciones listadas en wingdi.h y pasa la información proporcionada por la aplicación al GDI modo kernel por la vía del sistema de servicios Windows NT Ejecutive. La librería GDI modo kernel esta en WIN32K.SYS. El GDI modo kernel comunica con un controlador gráfico por el llamado de las funciones DDI listadas en winddi.h

Glyph: La figura actual (patrón de bits, perímetro, etc.) de una imagen carácter: Por ejemplo una 'a' itálica y una 'a' romana son dos diferentes glyphs representando el mismo carácter. En este contexto, glyph es un sinónimo para "imagen carácter" o simplemente "imagen".

GPC (Generic Printer Caracterization): Archivos de datos para el controlador de impresora RasDD.

GPD (Generic Printer Description): Archivos de datos para el controlador de impresora RasDD versión Windows NT 5.0.

GTT (Glyph Translation Table): Una glyph translation table especifica cadenas de selección requeridas para imprimir caracteres asociados con códigos de página (code points).

Graphic Engine (Máquina Gráfica): Ver la definición del ***GDI Modo Kernel***.

HAL (Hardware Abstraction layer): Un componente de "Windows NT Executive", que proporciona soporte especifico para la plataforma Kernel, I/O Manager, Depuradores modo kernel y controladores de dispositivos de bajo nivel para Windows NT. El HAL exporta rutinas que abstraen hardware especifico de

plataforma detallado acerca de caches, I/O buses, interrupciones, y provee una interface entre el hardware de la plataforma y el software del sistema. Por ejemplo, el HAL implementa una rutina para mapear cada bus de controlador de dispositivo relativo al vector de interrupciones de dispositivo a u n vector asignado con un prioritario sistema correspondiente hardware de plataforma especifica (DIRQL).

Hook: Una función llamable (callback) que maneja puertos de entrada y salida y accesos a tarjeta de memoria. Sinónimo de manejador (handler).

IEEE 1284 (Bi-directional Parallel Peripheral Interface): Interface estándar que agrega capacidades de comunicación bidireccional a dispositivos periféricos que son conectados al puerto paralelo de la computadora.

IOCTL (Input Output Control): Funciones que toman el control de I/O en un módulo de controlador de dispositivo.

IRP (I/O Request Packet): Un IRP es la estructura administradora básica de I/O usada para comunicarse con los controladores y permite a los controladores comunicarse con cada uno.

MDT (MiniDriver Development Tool): Herramienta de desarrollo que permite a los desarrolladores de controladores de impresoras raster de una manera más sencilla la elaboración de un minidriver. MDT es proporcionada con el DDK de W2K

Modo Kernel (Kernel Mode): El modo de procesador privilegiado en el cual Windows NT Executive corre. Un controlador que corre en modo kernel accesa a la memoria del sistema y al hardware.

Monitor de Lenguaje (Language Monitor): Componente del Spooler que permite al spooler configurar y monitorear el estado de una impresora bidireccional.

NTVDM o VDM: Máquina Virtual DOS Windows NT (Windows NT Virtual DOS Machine).

Palette (Paleta): Una tabla de colores.

PCD (Plotter Caracterization Data): Un DLL que proporciona información especifica del dispositivo para el controlador gráfico del plotter.

PDEV: Una representación lógica del dispositivo físico. Un PDEV es privado a un controlador y contiene toda la información y datos que representa el dispositivo físico asociado. Como una parte de la habilitación de un PDEV, un controlador proporciona información al GDI que describe el dispositivo requerido y sus capacidades. Información importante que el controlador brinda al GDI es un conjunto de banderas que permiten al GDI determinar que tipo de operaciones es pDEV y el dispositivo requerido soportan.

PFM (Printer Fonts Metrics): Archivo que describe las fuentes para un controlador.

PJL (Printer Job Language): Lenguaje que implementa comunicación entre un dispositivo impresor bidireccional y una computadora. PJL fue definido e implementado por Hewlett Packard. Para una mayor definición,contactar a HP.

Monitor de Puerto (Port Monitor): Componente del spooler que controla el puerto de I/O al cual la impresora está conectada.

PPD (PostScript Printer Description): Un archivo leído por el controlador genérico PostScript como arte de la inicialización del dispositivo. Los archivos PPD funcionan conforme a las especificaciones publicadas por la Corporación Adobe y por los fabricantes de impresoras PostScript.

Preimaging: El Proceso de construir una plantilla de película en un buffer de memoria antes de ser desplegada.

RasDD: Controlador de Dispositivo de impresoras Raster. Este controlador soporta la impresión de gráficos raster (mapas de bits).

Raster: Significa arreglo de pixeles o mapas de bits. La GDI proporciona dicersas funciones para este propósito (dibujar gráficos raster).

Realized Brush: Una estructura de datos conteniendo información y aceleradores que el controlador de monitor o impresora necesita con el fin de rellenar una área con un patrón.

Resolución: Todas las impresoras soportan uno o más resoluciones horizontales y verticales para salidas gráficas raster (mapa de bits). Una resolución es un valor

entero especificado en dots por pulgada (dpi: dots per inch), que indica el número de los distintos dots que la cabeza de la impresora puede desplazarse en x, y.

RLE bitmap (Run Length Encoded Bitmap): Los mapas de bits son comprimidos para reducir el espacio requerido en disco y memoria. Un formato de codificación es especificado en el miembro biCompression de la estructura BITMAPINFO.

ROP (Raster Operation): Las operaciones de bits aplicadas a los bits de datos de color para los pinceles replicados y los bits de datos de color para el rectángulo destino. Hay 256 ROPs en Windows.

Streaming (Vaciado): Proceso de transferir información desde algún dispositivo a un controlador de dispositivo.

Superficie: Las salidas de dibujo y texto requieren una superficie, asociado con el dispositivo requerido en el cual se dibuja. Tal como una superficie de dispositivo es un subconjunto de un arreglo de 2 xy 28 por 2 xy 28 pixeles. Esos pixeles están direccionados por pares de números signados de 28 bits. El pixel de la esquina superior izquierda tiene la coordenada (0,0). Las superficies son asociadas a un PDEV particular.

Tagged File Format: Formato de archivo en el cual los datos son etiquetados al utilizar cabeceras estándar que identifiquen longitud y tipos de información.

Unitool (Versión Anterior del MDT): Herramienta de desarrollo que permite a los desarrolladores de controladores de impresoras raster de una manera más sencilla la elaboración de un minidriver. Unitool y MDT es proporcionada con el DDK de Windows NT y W2K respectivamente.

Modo Usuario (User Mode): Es un modo no privilegiado del procesador en el cual el código de aplicación corre, incluyendo el código del subsistema protegido. Las aplicaciones en modo usuario no pueden accesar directamente a los datos del sistema, excepto por las llamadas a funciones, las cuales llaman a los recursos del sistema.

VDD (Virtual Device Driver): Un driver de dispositivo que opera en modo usuario y comunica con el correspondiente controlador de dispositivo en el modo kernel. Un VDD soporta sólo dispositivos hardware de proposito especial desde una aplicación MS-DOS. Un emulador de instrucciones que convierte instrucciones x86 a

instrucciones MIPS. Actúa como una plantilla entre aplicaciones MS-DOS y el hardware conectado a la máquina Windows NT.

Windows NT Virtual DOS Machine: Máquina Virtual DOS Windows NT. Un subsistema en ambiente protegido que emula al MS-DOS a al Windows de 16 bits en Windows NT. El VDM es creado cuando el usuario comienza una aplicación MS-DOS en Windows NT.

WOW(Windows On Win32): Un subsistema de ambiente protegido que corre en un proceso VDM. Proporciona las capacidades de ambiente de Windows 16 bits de correr algún número de aplicaciones 16 bits en Windows NT.

Xlate Object (translate Object): Tablas que asisten en la administración de paletas, para trasladar un índice de colores de una aplicación a un índice gráfico.

Aclaraciones

Todos los nombres de los sistemas operativos Windows y Windows NT , así como los programas y herramientas de desarrollo elaboradas por Microsoft para la creación de Controladores de Dispositivos como UNITOOL, SDK y DDK mencionados aquí, así como los componentes que los integran y caracterizan son propiedad intelectual registrada por Microsoft Corporation y su mención en esta obra es sólo con fines académicos sin ningún beneficio económico.

La base del material de apoyo e investigación y el resultado de la presente compilación, es o son productos de software de y para el Sistema Operativo Windows NT, cuya empresa creadora es Microsoft Corporation. También las herramientas de desarrollo utilizadas para lograr nuestro propósito son propiedad intelectual de Microsoft.

Microsoft tiene reservados los siguientes derechos:

Además tiene otros derechos como: Microsoft, Microsoft Press, MS, MS-DOS, Visual Basic, Visual C++, Windows, Win32, Win32s, y Windows NT son todos marcas registradas de Microsoft Corporation en los Estados Unidos y en otros paises.

PostScript es una Marca Registrada de Adobe Systems, Inc.

Macintosh y TrueType son marcas Registradas de Apple Computer, Inc.

Hewlett-Packard, HP, LaserJet, Openview, and PCL son Marcas Registradas de Hewlett-Packard Company.

Otros productos y nombres de compañías mencionadas son marcas registradas.

Bibliografía

a) Bibliografía Consultada

Microsoft Windows NT Server, Supplement 1
CopyRight 1996 Microsoft Corporation
ISBN: 1-57231-344-7
ISBN: (Obra Completa en Español) 84-481-1174-5

Microsoft Windows NT Server, Kit de Recursos, Suplemento 1
Derechos reservados 1997, respecto a la primera edición en español por McGRAW-HILL/INTREAMERICANA de ESPAÑA, S. A. U.
Traducción Javier Molinero Velasco – Universidad Politécnica de Madríd
ISBN: 84-481-1177-X

Microsoft Windows NT Server, Kit de Recursos, Guia de Recursos
Derechos reservados 1997, respecto a la primera edición en español por McGRAW-HILL/INTREAMERICANA de ESPAÑA, S. A. U.
Traducción Enrique Barja Sánchez – Universidad Politécnica de Madrid
ISBN: 84-481-1175-3

Developing Windows NT Device Drivers: A Programmer's Handbook
Edward N. Dekker, Joseph M. Newcomer
Abril 1999
Addison-Wesley Pub Co; ISBN: 0201695901

Inside Windows 2000, 3rd Edition
por David Solomon y Mark Russinovich
Microsoft Press, ISBN 0-7356-1021-5
Inside Windows NT - Second Edition(Microsoft Programming Series)
David A. Solomon
Segunda edición (Mayo de 1998)
Microsoft Press; ISBN: 1572316772

Programming the Microsoft Windows Driver Model
Walter Oney
Microsoft Press, 1999, ISBN 0-7356-0588-2

Windows 2000 Kernel Debugging
Steven McDowell
Primera edición (1/5/01) Prentice Hall Computer Books; ISBN: 0130406376

The Windows NT Device Driver Book: A Guide for Programmers
Art Baker
Prentice-Hall Inc, 1997, ISBN:0-13-184474-1

Windows NT Device Driver Development
Peter Viscarola and W. Anthony Mason
Primera edición (Noviembre 10, 1998)
Macmillan Technical Publishing; ISBN: 1578700582

Writing Secure Code
Michal Howard and David LeBlanc
Primera edición (Diciembre 15, 2001)
Microsoft Press; ISBN: 0735615888

b) Mesografía Consultada

http://www.microsoft.com/ddk/
http://www.microsoft.com/hwdev/driver/default.asp
http://www.microsoft.com/hwdev/tech/print/default.asp
http://www.microsoft.com/hwdev/tech/color/icmwp.asp
http://www.color.org/icc_specs2.html
http://www.w3.org/Graphics/Color/sRGB.html