Librería: winfis32.dll v04.27

Para uso con impresoras fiscales HASAR

Argentina , Venezuela, Panamá

Contenido

Librería winfis32.dll v04.27

Se trata de una librería de enlace dinámico para desarrollos visuales en entorno Windows® 32 / 64 bits, la cual puede emplearse de dos formas:

Combinada

- Para desarrollos basados en el uso del OCX Fiscal HASAR v010724.

El software de gestión para puntos de venta configura propiedades, e invoca métodos, pertenecientes al mencionado OCX. Es este objeto (y no el software de gestión para puntos de venta) quien finalmente utiliza las funciones disponibles en la librería *winfis32.dll*.

	(Válido sólo en Argentina) *- Para desarrollos basados en el uso del ejecutable wspooler.exe.* Existen en el mercado incontables desarrollos para MS-DOS® -que corren en ventanas MS-DOS® de Windows®-, los cuales no cuentan con capacidad para interactuar con el OCX, y tampoco cuentan con la posibilidad de declarar una interfaz para disponer de las funciones de la librería *winfis32.dll*. En estos casos, el desarrollador suele recurrir al uso del ejecutable *wspooler.exe*, programa que requiere de la librería *winfis32.dll* para poder operar. Es este programa (y no el software de gestión para puntos de venta) quien utiliza las funciones disponibles en la librería *winfis32.dll*.
Independiente	*- Para desarrollos basados en el uso de las funciones incluidas en la librería.* En este caso, el software de gestión para puntos de venta no hace uso del *OCX Fiscal HASAR v010724*, o el ejecutable *wspooler.exe*, prefe-renciando –el diseñador- la invocación directa de las funciones disponibles en la librería *winfis32.dll*.

Si bien es cierto que resulta más amigable el desarrollo de un proyecto basado en el uso del *OCX Fiscal Hasar v010724* (o posteriores –basados en clases multiplataforma-), no todos los lenguajes de desarrollo visual soportan el empleo de objetos basados en tecnología 'COM' –como es el caso del OCX mencionado-, o de objetos basados en clases multiplataforma; y también es sabido que otros lenguajes no son ciento por ciento compatibles con el uso de OCX como los descriptos. El desarrollador deberá consultar este aspecto en la documentación de su entorno de desarrollo.

Para quienes opten por el uso directo de las funciones de la librería *winfis32.dll*, sea porque están acostumbrados a la generación directa de strings de comandos y al análisis de strings de respuestas (diálogo con la impresora fiscal HASAR), o porque las funciones de esta librería son independientes del modelo de impresora fiscal HASAR a manejar, se describen a continuación las funciones incluidas en la librería *winfis32.dll*.

A partir de la versión 4.00 de la librería *winfis32.dll* es posible el comercio de strings (comandos y respuestas) tanto a través de un puerto serie local, como de ethernet (indicando dirección de IP y número de socket a considerar).

Funciones disponibles

Si bien en las descripciones de las funciones incluidas en la librería *winfis32.dll* –escrita en lenguaje C- se indica que para algunos argumentos y valores retornados, el tipo de dato declarado es **int**, se debe tener en cuenta que se trata de un entero representado en 32 bits (ocupa 4 bytes). Este hecho es importante a la hora de declarar la interfaz requerida en otros lenguajes que utilicen estas funciones. Por ejemplo, en Visual Basic los mismos datos deben ser declarados como Long.

Función: VersionDLLFiscal( )

int *VersionDLLFiscal* (void)

Argumentos:	No requiere.
Retorna:	El número correspondiente a la versión de la librería *winfis32.dll* en uso.

Esta función permite al software de gestión para puntos de venta determinar si está trabajando con la versión adecuada de la librería *winfis32.dll*. Tener presente que las funciones disponibles pueden ser modificadas o eliminadas, además de agregarse nuevas.

Función: OpenComFiscal( )

int *OpenComFiscal* (int Com, int Mode)

Argumentos:	**Com**	Número entero que indica el puerto serie que se desea abrir. Este argumento puede tomar valores entre 1 y 30.	
	Mode	ANSI = 1	Este valor indica que el comercio de strings entre la librería *winfis32.dll* y el software de gestión para puntos de venta se realizará en base a caracteres ANSI. En este caso, la librería *winfis32.dll* convertirá los caracteres ANSI a sus equivalentes en ASCII antes de enviar los strings de comandos a la impresora fiscal HASAR. En sentido inverso, las respuestas en caracteres ASCII que entrega la impresora fiscal Hasar serán convertidos a sus equivalentes en ANSI antes de entregar el string al software de gestión para puntos de venta.
		ASCII = 0	Este valor indica que el comercio de strings entre

		la librería *winfis32.dll* y el software de gestión para puntos de venta se realizará en base a caracteres ANSI. En este caso la librería *winfis32.dll* no realiza conversión alguna de los strings de comandos, ni de los correspondientes a respuestas de la impresora fiscal HASAR.
Retorna:	**> = 0**	Número (positivo) de 'handler' devuelto por el sistema operativo, para manejar el puerto serie indicado como el argumento **Com**. Este valor será requerido por el resto de las funciones de la librería *winfis32.dll* que dialoguen con la impresora fiscal HASAR, o deban hacer uso del puerto serie (cerrarlo). Este número indica que la apertura del puerto serie fue exitosa.
	< 0	Un número negativo indica que se ha producido un error. Más adelante se describen los valores posibles de error.

Esta función permite al software de gestión para puntos de venta la apertura del puerto serie, de la PC, donde se conectará la impresora fiscal HASAR.

Es conveniente que la apertura del puerto serie se realice cuando la aplicación es disparada para su ejecución, y no por la emisión de cada comprobante. Tener presente que el puerto serie está dedicado a la impresora fiscal HASAR.

Función: ReOpenComFiscal( )

int *ReOpenComFiscal* (int Com)

Argumentos:	**Com**	Número entero que indica el puerto serie que ya se encontraba abierto. Este argumento puede tomar valores entre 1 y 30.
Retorna:	**> = 0**	Número (positivo) de 'handler' devuelto por el sistema operativo, para manejar el puerto serie indicado como el argumento **Com**. Este valor será requerido por el resto de las funciones de la librería *winfis32.dll* que dialoguen con la impresora fiscal HASAR, o deban hacer uso del puerto serie (cerrarlo). Este número indica que la apertura del puerto serie fue exitosa.
	< 0	Un número negativo indica que se ha producido un error. Más adelante se describen los valores posibles de error.

Esta función permite al software de gestión para puntos de venta recuperarse del error que se produce al intentar abrir un puerto serie que ya estaba abierto.

Función: CloseComFiscal( )

void *CloseComFiscal* (int Handler)

Argumentos:	**Handler**	Valor devuelto por la función *OpenComFiscal( )* –normalmente-, o por la función *ReOpenComFiscal( )* –excepcionalmente-.
Retorna:	------	

Esta función permite al software de gestión para puntos de venta cerrar el puerto serie utilizado para comerciar strings de comandos y respuestas, con la impresora fiscal HASAR.

Es conveniente que el cierre del puerto serie se realice cuando se da por finalizada la ejecución de la aplicación, y no por la emisión de cada comprobante. Tener presente que el puerto serie está dedicado a la impresora fiscal HASAR.

Función: InitFiscal( )

int InitFiscal (int Handler)

Argumentos:	**Handler**	Valor devuelto por la función *OpenComFiscal( )* –normalmente-, o por la función *ReOpenComFiscal( )* –excepcionalmente-.
Retorna:	**= 0**	La función se ha ejecutado con éxito.
	< 0	Un número negativo indica que se ha producido un error. Más adelante se describen los valores posibles de error.

Esta función permite al software de gestión para puntos de venta la sincronización de la numeración de paquetes correspondientes al protocolo de comunicaciones para evitar la retransmisión de paquetes. La sincronización la logra la propia función mediante el envío de sucesivos comandos de pedidos de estado (*StatusRequest* –ver manual *publ….pdf*-) a la impresora fiscal HASAR.

Es recomendable que el software de gestión para puntos de venta invoque a esta función después de abrir con éxito el puerto serie.

Función: MandaPaqueteFiscal( )

int *MandaPaqueteFiscal* (int Handler, char *Buffer)

Argumentos:	**Handler**	Valor devuelto por la función *OpenComFiscal( )* –normalmen

		te-, o por la función *ReOpenComFiscal( )* –excepcionalmente-.
	Buffer	String de comando destinado a la impresora fiscal HASAR.
Retorna:	**= 0**	La función se ha ejecutado con éxito.
	< 0	Un número negativo indica que se ha producido un error. Más adelante se describen los valores posibles de error.

Esta función permite al software de gestión para puntos de venta enviar un string de comando a la impresora fiscal HASAR, a través del puerto serie local. Dicho string debe respetar la sintaxis descripta en el manual de comandos correspondiente (*publ….pdf* –el nombre del archivo depende el modelo de equipo en uso-).

El formato del string de comando, básicamente, puede describirse como:

ID[∟*Campo1*[∟*Campo2*[∟…..[∟*CampoN*]…..]]]

ID	Caracter de identificación del comando.
∟	Separador de campos de información: ASCII 28 (decimal).
Campo…	La cantidad de campos, formato y contenido se describen en el manual *publ….pdf*.
[…]	Indica que la presencia del campo depende del comando en cuestión.

Función: UltimaRespuesta( )

int UltimaRespuesta (int Handler, char *Buffer)

Argumentos:	**Handler**	Valor devuelto por la función *OpenComFiscal( )* –normalmente-, o por la función *ReOpenComFiscal( )* –excepcionalmente-.
	Buffer	String de respuesta proveniente de la impresora fiscal HASAR. La variable donde se alojará el string de respuesta debe ser dimensionada de manera de poder contener la respuesta más larga conocida. Generalmente 512 bytes son suficientes.
Retorna:	**= 0**	La función se ha ejecutado con éxito.
	< 0	Un número negativo indica que se ha producido un error. Más adelante se describen los valores posibles de error.

Esta función permite al software de gestión para puntos de venta obtener el string de respuesta de la impresora fiscal HASAR, a través del puerto serie local, y correspondiente al último comando enviado mediante la función *MandaPaqueteFiscal( )*. Dicho string respeta

la sintaxis descripta en el manual de comandos correspondiente (*publ….pdf* –el nombre del archivo depende el modelo de equipo en uso-).

El formato del string de respuesta, básicamente, puede describirse como:

ID∟StatPrn∟StatFis[∟*Campo1*[∟*Campo2*[∟…..[∟*CampoN*]…..]]]

StatPrn	Campo conteniendo el estado en que se encuentra la impresora propiamente dicha. Este campo se halla presente en todas las respuestas de la impresora fiscal HASAR. Consta de 4 caracteres ASCII que representan dígitos hexadecimales. El significado de cada bit de su representación binaria se describe en el apéndice 'Status de Impresora', incluido en el manual de comandos *publ….pdf*. Por ejemplo: 'C080'.
StatFis	Campo conteniendo el estado fiscal en que se encuentra la impresora fiscal HASAR. Este campo se halla presente en todas las respuestas de la impresora fiscal HASAR. Consta de 4 caracteres ASCII que representan dígitos hexadecimales. El significado de cada bit de su representación binaria se describe en el apéndice 'Status Fiscal', incluido en el manual de comandos *publ….pdf*. Por ejemplo: '0600'.
ID	Caracter de identificación del comando al cual corresponde la respuesta.
∟	Separador de campos de información: ASCII 28 (decimal).
Campo…	La cantidad de campos, formato y contenido se describen en el manual *publ….pdf*.
[…]	Indica que la presencia del campo depende del comando en cuestión.

La normativa fiscal vigente (en Argentina) obliga al software de gestión para puntos de venta a examinar cada respuesta de la impresora fiscal, a los efectos de determinar que ocurrió con el comando enviado, y proceder al tratamiento adecuado de errores, si éstos son reportados por la impresora fiscal HASAR. Además, las respuestas pueden contener información de interés para el software de gestión para puntos de venta con objeto de mantener la sincronización de datos entre el software y la impresora fiscal HASAR.

Función: UltimoStatus( )

int UltimoStatus (int Handler, short *FiscalStatus, short *PrinterStatus)

Argumentos:	**Handler**	Valor devuelto por la función *OpenComFiscal( )* –normalmente-, o por la función *ReOpenComFiscal( )* –excepcionalmente-.
	FiscalStatus	Valor numérico, resultado de la conversión del campo de estado fiscal contenido en el string de la última res-

		puesta recibida desde la impresora fiscal HASAR.
	PrinterStatus	Valor numérico, resultado de la conversión del campo de estado de impresora contenido en el string de la última respuesta recibida desde la impresora fiscal HASAR.
Retorna:	**= 0**	La función se ha ejecutado con éxito.
	< 0	Un número negativo indica que se ha producido un error. Más adelante se describen los valores posibles de error.

Esta función permite al software de gestión para puntos de venta ahorrarse la extracción y conversión a binario de los strings contenidos en los campos de estado incluidos en las respuestas de la impresora fiscal HASAR.

La normativa fiscal vigente (en Argentina) obliga al software de gestión para puntos de venta a examinar cada respuesta de la impresora fiscal, a los efectos de determinar que ocurrió con el comando enviado, y proceder al tratamiento adecuado de errores, si éstos son reportados por la impresora fiscal HASAR.

Para más información acerca del significado de cada bit de estado, consultar los apéndices 'Status de Impresora' y 'Status Fiscal', incluidos en el manual *publ….pdf* correspondiente al equipo en uso.

Un bit en 1 en el campo de estado deberá ser interpretado de acuerdo al contexto en el que se lo está analizando. Por ejemplo, si el status fiscal fuese '8620' (comando inválido) y forma parte de la respuesta al comando de cierre de un comprobante fiscal (comando *CloseFiscalReceipt*), es altamente probable que se trate de un error. Pero si se incluye en la respuesta al comando de cancelación general (*Cancel*), y este comando fue enviado en la previsión de que podría haber un comprobante abierto, en realidad debería interpretarse como 'Continuar'.

Supongamos que se ha invocado a la función *UltimoStatus( )* y se quiere analizar cada bit, por ejemplo, del estado fiscal. Una forma de hacerlo podría ser:

```
mask = 1                                                '// Bit cero, en 1

For ind = 1 To 16                                       '// Se recorren los bits del Status
     If ((FiscalStatus And mask) <> 0) Then             '// Está el bit en 1 ??
          '// Decidir si es un error o no, y como tratarlo
          '// Tener presente el estado del software
     End If

    mask = mask * 2     '// Se desplaza el bit en 1 de la máscara un lugar a la izquierda
Next
```

Función: CambiarVelocidad( )

int CambiarVelocidad(int PortNumber, long NewSpeed)

Argumentos:	**PortNumber**	Número entero que indica el puerto serie al que se le desea modificar la velocidad en baudios con la que trabaja. Este argumento puede tomar valores entre 1 y 30.
	NewSpeed	Velocidad en baudios, con la que se desea configurar el puerto serie indicado mediante el argumento **PortNumber**. Los valores posibles son: 1200, 2400, 4800, 9600, 19200, 38400 y 57600.
Retorna:	**= 0**	La función se ha ejecutado con éxito.
	< 0	Un número negativo indica que se ha producido un error. Más adelante se describen los valores posibles de error.

Esta función permite al software de gestión para puntos de venta modificar la velocidad de trabajo del puerto serie de la PC.

La función *CambiarVelocidad( )* no modifica la velocidad del puerto serie de la impresora fiscal HASAR. El uso de esta función solamente tiene sentido cuando se conectan impresoras fiscales HASAR que soportan distintas velocidades en baudios.

Sin embargo, si se desea modificar la velocidad del puerto serie de la impresora fiscal HASAR (suponiendo que estuviese permitido en el equipo en uso), se debe construir el string de comando *SetComSpeed* y hacerlo llegar a la impresora fiscal HASAR a través del función *MandaPaqueteFiscal( )* –para más información consultar el manual *publ….pdf*-.

Cabe aclarar que el cambio de velocidad afecta solamente a la transferencia electrónica de datos. No afecta la velocidad de impresión del equipo.

Por otro lado, recordar primero modificar la velocidad del puerto serie de la impresora fiscal HASAR, y luego la velocidad del puerto serie de la PC.

Función: SearchPrn( )

long SearchPrn(int Handler)

Argumentos:	**Handler**	Valor devuelto por la función *OpenComFiscal( )* –normalmente-, o por la función *ReOpenComFiscal( )* –excepcionalmente-.
Retorna:	**> 0**	La función se ha ejecutado con éxito. Velocidad en baudios a la que han podido sincronizarse ambos puertos serie (el de la

		PC y el de la impresora fiscal HASAR). Los valores posibles son: 1200, 2400, 4800, 9600, 19200, 38400 y 57600.
	< 0	Un número negativo indica que se ha producido un error. Más adelante se describen los valores posibles de error.

Esta función permite al software de gestión para puntos de venta determinar la velocidad a la que se encuentra configurada la impresora fiscal HASAR.

Función: ProtocolMode( )

void ProtocolMode(int Mode)

Argumentos:	**Mode**	Mediante este argumento se selecciona el modo de trabajo del protocolo de comunicaciones. Los valores posibles son: **0** No permite consulta de estado intermedio. **1** Permite la consulta de estado intermedio.
Retorna:	------	

Función: BusyWaitingMode( )

void BusyWaitingMode(int Mode)

Argumentos:	**Mode**	Mediante este argumento se selecciona el modo de control de retorno. Los valores posibles son: **0** La librería *winfis32.dll* cederá la CPU en los momentos de espera. En este modo hay que asegurar por medio del software de gestión para puntos de venta que lo usa que dos paquetes no se superpongan para un mismo handler. **1** La librería *winfis32.dll* no retornará hasta que el proceso se haya terminado. Este último modo es el default, y el más seguro, pero también es el más lento.
Retorna:	------	

Función: SetCommandRetries( )

DEPRECADA – A partir de la versión 04.00

int SetCommandRetries(int Retries)

Argumentos:	**Retries**	Configura el número de reintentos máximo de transmisiones de paquetes fiscales transmitidos por la función *MandaPaqueteFiscal( )* cuando la impresora fiscal HASAR no responde (time-out).
Retorna:	**> 0**	La función se ha ejecutado con éxito. Devuelve el número de reintentos previo a la reconfiguración.
	< 0	Un número negativo indica que se ha producido un error. Más adelante se describen los valores posibles de error.

Función: Abort( )

void Abort (int PortNumber)

Argumentos:	**PortNumber**	Interrumpe (aborta) el proceso en curso correspondiente al puerto especificado en este argumento.
Retorna:	------	

Se utiliza para abortar procesos que se encuentran en estado de espera hasta que se cumpla un cierto time-out.

Función: SetModoEpson( )

int SetModoEpson(void)

Esta función es utilizada para configurar el puerto para establecer una comunicación con el protocolo Epson, ya que por default este es configurado según el protocolo HASAR.

Argumentos:		No requiere.
Retorna:	**= 0**	La función se ha ejecutado con éxito.
	< 0	Un número negativo indica que se ha producido un error. Más adelante se describen los valores posibles de error.

Función: SetCmdRetries( )

int FAR PASCAL _export SetCmdReries(int retries)

Argumentos:	**retries**	Configura el número de reintentos máximo de transmisiones de paquetes fiscales transmitidos por la función *MandaPaqueteFiscal( )* cuando la impresora fiscal HASAR responde de manera inválida.
Retorna:	**= 0**	La función se ha ejecutado con éxito. Devuelve el número de reintentos previo a la reconfiguración.
	< 0	Un número negativo indica que se ha producido un error. Más adelante se describen los valores posibles de error.

La librería *winfis32.dll* permite una forma alternativa de establecer este valor, mediante la creación de una variable de entorno (de sistema) en Windows®:

Nombre: CMDRETRIES
Valor: Cantidad de reintentos. Por ejemplo, 3

Este es un recurso extremo que se puede utilizar cuando se está trabajando sobre redes donde puede haber una cierta lentitud en las comunicaciones.

Función: SetSndRetries( )

int FAR PASCAL _export SetSndReries(int retries)

Argumentos:	**retries**	Configura el número de reintentos máximo de transmisiones de paquetes fiscales transmitidos por la función *MandaPaqueteFiscal( )* cuando no se recibe el ACK, desde la impresora fiscal HASAR, al comando enviado.
Retorna:	**= 0**	La función se ha ejecutado con éxito. Devuelve el número de reintentos previo a la reconfiguración.
	< 0	Un número negativo indica que se ha producido un error. Más adelante se describen los valores posibles de error.

La librería *winfis32.dll* permite una forma alternativa de establecer este valor, mediante la creación de una variable de entorno (de sistema) en Windows®:

Nombre: SNDRETRIES
Valor: Cantidad de reintentos. Por ejemplo, 3

Este es un recurso extremo que se puede utilizar cuando se está trabajando sobre redes donde puede haber una cierta lentitud en las comunicaciones.

Función: SetRcvRetries( )

int FAR PASCAL _export SetRcvRetries(int retries)

Argumentos:	**retries**	Configura el número de reintentos máximo de transmisiones de paquetes fiscales transmitidos por la función *MandaPaqueteFiscal( )* cuando la impresora fiscal HASAR no responde (time-out).
Retorna:	**= 0**	La función se ha ejecutado con éxito. Devuelve el número de reintentos previo a la reconfiguración.
	< 0	Un número negativo indica que se ha producido un error. Más adelante se describen los valores posibles de error.

La librería *winfis32.dll* permite una forma alternativa de establecer este valor, mediante la creación de una variable de entorno (de sistema) en Windows®:

Nombre: RCVRETRIES
Valor: Cantidad de reintentos. Por ejemplo, 3

Este es un recurso extremo que se puede utilizar cuando se está trabajando sobre redes donde puede haber una cierta lentitud en las comunicaciones.

Función: OpenTpcFiscal( )

int FAR PASCAL _export OpenTcpFiscal(char *hostname, int socket, long mseg, int modo)

Argumentos:	**hostname**		A través de este parámetro se indica el nombre que identifica a una PC en la red –por ejemplo, “PCVentas1”-, o su dirección IP –por ejemplo, “192.0.5.135”-.
	socket		Número de socket mediante el cual se comerciarán strings de comandos y respuestas entre el software de gestión para puntos de venta y la PC **hostname**. Por ejemplo, 5500.
	mseg		Tiempo de espera en milisegundos para retornar informando error de time-out. Por ejemplo, 2000.
	modo	ANSI = 1	Este valor indica que el comercio de strings entre la librería *winfis32.dll* y el software de gestión para puntos de venta se realizará en base a caracteres ANSI. En este caso, la librería *winfis32.dll* convertirá los caracteres ANSI a sus equivalentes en ASCII antes de enviar los strings de comandos a la impresora fiscal HASAR. En

			sentido inverso, las respuestas en caracteres ASCII que entrega la impresora fiscal Hasar serán convertidos a sus equivalentes en ANSI antes de entregar el string al software de gestión para puntos de venta.
		ASCII = 0	Este valor indica que el comercio de strings entre la librería *winfis32.dll* y el software de gestión para puntos de venta se realizará en base a caracteres ANSI. En este caso la librería *winfis32.dll* no realiza conversión alguna de los strings de comandos, ni de los correspondientes a respuestas de la impresora fiscal HASAR.
Retorna:	**= 0**	La función se ha ejecutado con éxito.	
	< 0	Un número negativo indica que se ha producido un error. Más adelante se describen los valores posibles de error.	

Esta función permite al software de gestión para puntos de venta dialogar con la impresora fiscal en forma remota. Es decir, la impresora fiscal HASAR está conectada en otra PC en la cual debe estar corriendo el ejecutable *wspooler.exe* para "escuchar" la red, redireccionando al puerto serie los strings de comandos que llegan y devolviendo a la red los strings de respuesta que recibe desde la impresora fiscal HASAR.

Función: ObtenerNumeroDePaquetes( )

```
int FAR PASCAL _export  ObtenerNumeroDePaquetes(int handler, int *paqsend,
                                                int *paqrcv, int *idcmd)
```

Argumentos:	**handler**	Valor devuelto por la función *OpenComFiscal( )* –normalmente-, o por la función *ReOpenComFiscal( )* –excepcionalmente-.
	paqsend	En este argumento la función retorna el número de secuencia del último paquete fiscal, con un string de comando, que ha sido enviado a la impresora fiscal HASAR.
	paqrcv	En este argumento la función retorna el número de secuencia del último paquete fiscal, con un string de respuesta, que ha sido recibido desde la impresora fiscal HASAR.
	idcmd	En este argumento la función retorna el caracter identificador de comando al cual corresponde la respuesta contenida en el último paquete fiscal recibido desde la impresora fiscal HASAR.
Retorna:	**= 0**	La función se ha ejecutado con éxito.
	< 0	Un número negativo indica que se ha producido un error. Más adelante se describen los valores posibles de error.

Mediante esta función el software de gestión para puntos de venta puede controlar el sincronismo del protocolo fiscal verificando que el número de secuencia del paquete fiscal, correspondiente al string de comando enviado, se corresponda con el correspondiente al paquete fiscal que trae el string de respuesta, chequeando (además) que la respuesta recibida corresponda al comando enviado (mismo caracter de identificación).

Función: SetKeepAliveHandlerStdCall( )

No válida para programación en C, C++, y/o Visual C.
Para los lenguajes mencionados ver: SetKeepAliveHandler( )

void FAR PASCAL _export SetKeepAliveHandlerStdCall(PFVSTDCALL handler)

Argumentos:	**handler**	Valor devuelto por la función *OpenComFiscal( )* –normalmente-, o por la función *ReOpenComFiscal( )* –excepcionalmente-.
Retorna:	------	

En determinadas circunstancias la impresora fiscal HASAR puede enviar al software de gestión para puntos de venta, cada 400 milisegundos, caracteres ASCII especiales que pueden interpretarse como falta papel, o esperar (la respuesta al comando aún no está lista).

La impresora fiscal HASAR reporta la falta de papel (como una de sus formas) mediante el envío –cada 400 milisegundos- del caracter ASCII 'DC4' (18 decimal ; 12 hexa). Esto ocurrirá hasta que la situación se normalice.

Cuando el procesamiento de un comando enviado por el software de gestión para puntos de venta requiere de más de 400 milisegundos por parte de la impresora fiscal HASAR para entregar la respuesta, ésta recurre al envío –cada 400 milisegundos- del caracter ASCII 'DC2' (20 decimal ; 14 hexa). Esto debe ser interpretado por el software de gestión para puntos de venta como una espera para el envío del siguiente comando. Tener presente que no se puede enviar un nuevo comando hasta que se haya recibido y procesado la respuesta del anterior.

El siguiente es un ejemplo en Visual Basic de uso de esta función.

```
'// Rutina que atiende la llegada de caracteres 'DC2' o 'DC4'
'// -------------------------------------------------------------------

Public Sub ManejadorDeMantengaseVivo(ByVal Razon As Long, ByVal Handler As Long)

    MsgBox "Llegó un DC2 o un DC4...: " & Razon

End Sub

'// Cuando se carga el form se estable que rutina atenderá la llegada de caracteres 'DC2'
'// o 'DC4'
'// -------------------------------------------------------------------------------------------------
```

```
Public Sub PonerManejadorDeMantengaseVivo()

   SetKeepAliveHandlerStdCall AddressOf ManejadorDeMantengaseVivo

End Sub
```

Función: SetKeepAliveHandler( )

De uso exclusivo para programación en C, C++, y/o Visual C.
Para los lenguajes mencionados ver: SetKeepAliveHandlerStdCall( )

```
void FAR PASCAL _export SetKeepAliveHandlerStdCall(PFVSTDCALL handler)
```

Argumentos:	**handler**	Valor devuelto por la función *OpenComFiscal( )* –normalmente-, o por la función *ReOpenComFiscal( )* –excepcionalmente-.
Retorna:	------	

En determinadas circunstancias la impresora fiscal HASAR puede enviar al software de ges tión para puntos de venta, cada 400 milisegundos, caracteres ASCII especiales que pueden interpretarse como falta papel, o esperar (la respuesta al comando aún no está lista).

La impresora fiscal HASAR reporta la falta de papel (como una de sus formas) mediante el envío –cada 400 milisegundos- del caracter ASCII 'DC4' (18 decimal ; 12 hexa). Esto ocurrirá hasta que la situación se normalice.

Cuando el procesamiento de un comando enviado por el software de gestión para puntos de venta requiere de más de 400 milisegundos por parte de la impresora fiscal HASAR para entregar la respuesta, ésta recurre al envío –cada 400 milisegundos- del caracter ASCII 'DC2' (20 decimal ; 14 hexa). Esto debe ser interpretado por el software de gestión para puntos de venta como una espera para el envío del siguiente comando. Tener presente que no se puede enviar un nuevo comando hasta que se haya recibido y procesado la respuesta del anterior.

Errores devueltos por las funciones

Errores generales:

-1	Error general.
-2	Handler inválido.
-3	Intento de enviar un comando cuando se estaba procesando.
-4	Error de comunicaciones.
-5	Puerto ya abierto.
-6	No hay memoria.
-7	El puerto ya estaba abierto.
-8	La dirección del buffer de respuesta es inválida.
-9	El comando no finalizó, sino que volvió una respuesta tipo STAT_PRN.
-10	El proceso en curso fue abortado por el usuario.
-11	No hay más puertos disponibles.

Errores TCP/IP:

-12	Error estableciendo comunicación TCP/IP.
-13	No se encontró el host.
-14	Error de conexión con el host.

Errores internos:

-15	Se recibió NAK al comando enviado.

Detección de Problemas

La librería *winfis32.dll* existe un mecanismo que permite el rastreo problemas consistente en guardar en un archivo de texto todas las operaciones realizadas por las funciones de la librería, junto con los detalles de las comunicaciones. Esto permite seguir el protocolo "de cerca". Para disponer de esta opción se deben especificar un par de variables de entorno (de sistema) en Windows® antes de ejecutar funciones de la librería *winfis32.dll*. Estas variables de entorno (de sistema) son:

FILELOG	Especifica el archivo de texto donde se va a guardar la información. Si este archivo no se especifica, no se guardará la información. Por ejemplo, [path\]archivo.log El directorio debe existir en la PC, el archivo lo crea la librería *winfis-32.dll*.
DEBUGLEVEL	Especifica el nivel de detalle de la información. Es un número entero de 1 a 3, donde : 1 Sólo la información estrictamente necesaria. 2 Información complementaria a la anterior. 3 Toda la información. Si esta variable de entorno (de sistema) no se especifica, se asume el valor 3.

¿ Caracteres ANSI o ASCII ?

La impresora fiscal HASAR trabaja con el juego de caracteres ASCII según el cual a la letra "ñ" le corresponde el valor decimal164. Dentro de Windows®, el juego de caracteres es el ANSI, según el cual es el valor 241 el que equivale a la letra "ñ". Ambos juegos difieren a partir del caracter 7F hexa (127 decimal).

Para evitar divergencias, se provee una conversión interna que salva este problema. Si la apertura del puerto serie se realiza invocando a la función *OpenComFiscal( )* e indicando el valor '1' en el argumento 'Mode', se convierten todos los strings; tanto los que van a la impresora fiscal HASAR (comandos), como los que vuelven (respuestas a los comandos enviados). En el caso de strings de comandos la conversión es de ANSI a ASCII, y en el caso de los strings de respuestas la conversión es de ASCII a ANSI.

Si el argumento 'Mode' vale '0' cuando se invoca a la función OpenComFiscal( ) los strings no son convertidos, se trate de comandos o respuestas.

Modo de Operación

La forma de operación del software de gestión para puntos de venta debería seguir los siguientes criterios:

Inicialización del software.	· Invocar a la función *OpenComFiscal( )* para abrir el puerto serie. · Loop hasta que la función *InitFiscal( )* devuelva éxito.

Si se ha invocado a la función *BusyWaitingMode( )* indicando el valor '0' en el argumento 'Mode', el software de gestión para puntos de venta debe controlar posibilidad de interferencias.

Durante la emisión de los distintos comprobantes.	· Se envían comandos a la impresora fiscal HASAR mediante la función *MandaPaqueteFiscal( )* controlando los valores que devuelve (al igual que en el resto de las funciones). · Se recupera el estado de la impresora fiscal HASAR mediante la función *UltimoStatus( )*, realizando chequeo de errores. · Posible uso de información contenida en la respuesta mediante la función *UltimaRespuesta( )*.
Se cierra la aplicación.	Se cierra el puerto serie mediante la función *CloseComFiscal( )*.

Modo BusyWaitingMode(0)

La librería *winfis32.dll* manejada en este modo tiene la ventaja de poder detectar problemas de comunicación, o de falta de papel, desde la aplicación que la usa.
Si la impresora fiscal HASAR se queda sin papel al recibir un comando que causa impresión, no responderá hasta que se normalice la situación. En estos casos, la *función MandaPaqueteFiscal( )* no devuelve el control, y el software de gestión para punto de venta parece "colgado". Para salir de este estado, podría dispararse un timer que transcurridos 'n' segundos le avise que algo está ocurriendo.

Por ejemplo, en Visual Basic:

```
Sub Mandar(Comando As String)
Dim rc As Integer

   Screen.MousePointer = vbHourglass
   Timer.Enabled = True

   Do
      rc = MandaPaqueteFiscal(Handler, Comando)

      If ((rc < 0) And (rc <> ERR_ATOMIC)) Then          '// ERR_ATOMIC = -3
         MsgBox "Error " & rc & " enviando paquete"
         Screen.MousePointer = vbNormal
         Timer.Enabled = False
```

```
            Exit Sub
          End If

        DoEvents
    Loop Until rc = 0

    Timer.Enabled = False
    Screen.MousePointer = vbNormal
End Sub

Private Sub Timer_Timer()
    '// Mostrar un mensaje de advertencia...
    ShowWarning.Show
End Sub
```

El formulario "ShowWarning" es del tipo 'DialogModal' y contiene un mensaje del tipo "Controle la impresora fiscal…". Tiene un timer que al segundo y medio descarga el formulario para permitir la ejecución de las funciones de la liobrería *winfis32.dll* nuevamente. El timer del formulario principal se puede activar en un lapso de cinco segundos.

La librería *winfis32.dll* tiene que estar funcionando en modo *BusyWaitingMode(0)*, obviamente. En este modo, hay que controlar que no se superpongan comandos, motivo por el cual hay que proveer cierta capa de protección. Por ejemplo:

```
Sub Mandar (Comando as String)
Static Atomic as Boolean

    If Atomic Then
        MsgBox "No puede ejecutar más de un comando por vez"
        Exit Sub
    End If

    Atomic = True

    '// Proceso real de envío de comandos...
    ...

    Atomic = False
End Sub
```

Ejemplo de aplicación en Windows®

El siguiente es un ejemplo de aplicación en Visual Basic. La aplicación abre el COM1 como y cambia la fecha y hora del reloj de tiempo real de la impresora fiscal HASAR.

Los carateres Chr$(28) son los separadores de campo que se utilizan en la construcción de los strings de comandos.

```
Declare Function MandaPaqueteFiscal Lib "WinFis32" (ByVal Handler As Integer, _
                                                    ByVal Name As String) As Integer
Declare Function OpenComFiscal Lib "WinFis32" (ByVal Puerto As Integer, _
                                               ByVal Mode As Integer) As Integer
Declare Function InitFiscal Lib "WinFis32" (ByVal Handler As Integer) As Integer
Declare Function UltimoStatus Lib "WinFis32" (ByVal Handler as Integer, _
                  ByRef FiscalStatus as Integer, ByRef PrinterStatus as Integer) as Integer

Const MODE_ANSI = 1

Dim Handler As Integer

‘//-----------------------------------------------
Sub EnviarStringFiscal(Comando As String)
Dim PrinterStatus As Integer, FiscalStatus As Integer

   If (MandaPaqueteFiscal(Handler, Comando) < 0) Then
       Screen.MousePointer = 0
       MsgBox "Error mandando paquete"
   else
       If UltimoStatus (Handler, FiscalStatus, PrinterStatus) = 0 Then
          MsgBox "PrinterStatus: " & PrinterStatus & _
                                     "FiscalStatus: " & FiscalStatus, vbOkOnly, "Status"
       End If
    End If
End Sub

‘//----------------------------------
Sub OpenPort (Port As Integer)
Handler = OpenComFiscal(Port, MODE_ANSI)

  If Handler < 0 Then
      MsgBox "No pude abrir puerto" & Port
      End
  End If
End Sub

‘//------------------
Sub Form_Load ()

  OpenPort 1

  If InitFiscal(Handler) < 0 Then
      MsgBox "Falló InitFiscal()"
      End
  End If

  EnviarStringFiscal "X" & Chr$(28) & "971231" & Chr$(28) & "235900"
End Sub
```

Setiembre 01, 2010 – Rev. 001

CompañíaHasar

GrupoHasar

Marcos Sastre 2214 [B1618CSD] Ricardo Rojas | Tigre | Buenos Aires | Argentina
tel: [54.11] 4117.8900 | fax: [54.11] 4117.8998 | www.grupohasar.com