

Title :
Titre :

DRIVER-NT4-PCI 102

Edition : 1 (Document creation - *Création du document*)

Written by M. ROCHE on 22/12 Visa []

Revised by B. THOUËNON on 22/12 Visa []

Approved by D. PIMONT on 22/12 Visa []

Warning : Unless otherwise stated, this revision overwrites the previous one which must be destroyed, along with any copies given to your collaborators.

Avertissement : En l'absence d'indication contraire, cette nouvelle édition annule et remplace l'édition précédente qui doit être détruite, ainsi que les copies faites à vos collaborateurs.

Edition <i>Edition</i>	Nature of the modifications (key words) <i>Nature des évolutions (mots clés)</i>	Written <i>Rédigé</i>	Revised/Approved <i>Revu/Approuvé</i>
2		by <u> [] </u> on <u> [] </u> Visa	by <u> [] </u> on <u> [] </u> Visa
3		by <u> [] </u> on <u> [] </u> Visa	by <u> [] </u> on <u> [] </u> Visa
4		by <u> [] </u> on <u> [] </u> Visa	by <u> [] </u> on <u> [] </u> Visa
5		by <u> [] </u> on <u> [] </u> Visa	by <u> [] </u> on <u> [] </u> Visa
6		by <u> [] </u> on <u> [] </u> Visa	by <u> [] </u> on <u> [] </u> Visa

DOCUMENT ARCHIVED

DOCUMENT ARCHIVE

No Yes on []

Δ ed. .. [] = Document input/output (*Entrée/sortie modification de la documentation*)

ed. .. [] = Board new function input/output (*Entrée/sortie nouvelle fonctionnalité du produit*)



NOTES :

DRIVER – NT4 –PCI 102

SOMMAIRE

A. INSTALLATION DU MATERIEL.....	5
B. INSTALLATION DU LOGICIEL	7
D. FONCTIONS LOGICIELLES	9

A. INSTALLATION DU MATERIEL

- ⇒ L'ordinateur doit être hors tension.
- ⇒ Déchargez-vous de votre potentiel électrostatique éventuel en touchant une surface métallique.
- ⇒ Installez les cartes en prenant soin de les configurer suivant les spécifications techniques et vos propres besoins.

Rappelons que nos cartes PCI sont Plug en Play et vous dispensent de configurer l'interface avec l'ordinateur. (voir documentation technique de chaque carte et le chapitre annexe).

Pour les cartes ISA, il est vivement conseillé de connaître l'environnement PC, notamment pour la configuration des ressources des cartes (Espace IO, Espace Mémoire, I/Os).

Une mauvaise configuration entraîne un blocage du système.

- ⇒ Mettez l'ordinateur sous tension.

B. INSTALLATION DU LOGICIEL

⇒ Insérez la disquette ou le CDROM dans l'ordinateur

⇒ Lancez le programme nommé Install.exe

⇒ suivre la procédure éventuelle, affichée à l'écran

⇒ Redémarrez l'ordinateur

A partir de ce stade la machine est en adéquation avec la nouvelle carte installée. Le driver tourne au niveau Kernel.

Chaque ressource de la carte peut être sollicitée au niveau « user » en faisant appel aux différents IOControl que nous allons expliquer dans le chapitre suivant.

D. FONCTIONS LOGICIELLES

Rappels :

Windows NT exécute les applications dans le mode user.

Uniquement les drivers mode Kernel peuvent tourner en mode Privilégié.
C'est le mode qui permet d'accéder aux périphériques.

Dans ce contexte, pour gérer au mieux les périphériques, le programmeur utilise les IOCTLs pour communiquer directement avec le driver mode Kernel.

Directement dans une application ou par l'intermédiaire d'une DLL.

Nous vous conseillons de suivre l'exemple d'application fourni sur la disquette.
Le programme exécutable est nommé appli.exe et le fichier source appli.c

Ce programme reprend principalement les IOCTLs du driver.

IOCTLs

La fonction **DeviceloControl** envoie directement au driver un code de contrôle.
Nous l'utiliserons pour envoyer tous les codes de contrôle supportés par le driver de la carte.

BOOL DeviceloControl(

HANDLE *hDevice*, // handle to device of interest

DWORD *dwIoControlCode*, // control code of operation to perform

LPVOID *lpInBuffer*, // pointer to buffer to supply input data

DWORD *nInBufferSize*, // size of input buffer

LPVOID *lpOutBuffer*, // pointer to buffer to receive output data

DWORD *nOutBufferSize*, // size of output buffer

LPDWORD *lpBytesReturned*, // pointer to variable to receive output byte count

LPOVERLAPPED *lpOverlapped* // pointer to overlapped structure for asynchronous operation

);

IOCTL	Description	Elements envoyés	Elements Recus
IOCTL_PCI102_MAP_MEMORY_RAM_CFG_PCI	map la mémoire de configuration PCI	NULL	Pointeur sur la mémoire CFG PCI
IOCTL_PCI102_MAP_MEMORY_RAM_CFG_102	map la mémoire de configuration 102	NULL	Pointeur sur la mémoire CFG 102
IOCTL_PCI102_MAP_MEMORY_RAM_MESURE	map la mémoire mesure	NULL	Pointeur sur la mémoire mesure
IOCTL_PCI102_UNMAP_MEMORY	Unmap de la mémoire réservée	Pointeur sur la mémoire à libérer	NULL
IOCTL_PCI102_MAP_MEMORY_MASTER	Map la mémoire que le PCI Master peut utiliser	NULL	Pointeur sur la mémoire Master
IOCTL_PCI102_READ_MASTER_MEMORY	Lecture de x Dword de la mémoire Master	- les index par rapport à la base - x Dword	- les valeurs des index - le nombre d'index effectifs
IOCTL_PCI102_MASTER_NEW_BLOCK	Requête pour un nouveau bloque A définir A définir	- Master write transfer count	NULL
IOCTL_PCI102_START	Commande START de la carte	NULL	NULL
IOCTL_PCI102_STOP	Commande STOP de la carte	NULL	NULL
IOCTL_PCI102_TRIG	Commande TRIG de la carte	NULL	NULL
IOCTL_PCI102_MOTIF_NB_VOIES	Génère 0x8000 motifs pour x voies	x voies	NULL
IOCTL_PCI102_RESET_AMCC_MASTER_INIT	Reset de la carte et repositionnement du pointeur Master	NULL	NULL
IOCTL_PCI102_INTERRUPTION	Rapport des interruptions	NULL	1 valeurs 32 bits : - InterruptCount;
IOCTL_PCI102_NUM_VERSION	Retourne le numéro de version du driver	NULL	Pointeur sur la version

Δ# éd 2 []

IOCTL_PCI102_MAP_MEMORY_RAM_CFG_PCI

map la mémoire de configuration PCI, cette opération est utilisée à l'initialisation de l'application,

elle permet d'accéder à cette mémoire grâce au pointeur retourné.

```
dwIoControlCode = IOCTL_PCI102_MAP_MEMORY_RAM_CFG_PCI;
    // operation code
lpInBuffer = NULL; // address of input buffer; not used; must be NULL
nInBufferSize = 0; // size of input buffer; not used; must be zero
lpOutBuffer = &pMemCfgPCI; // address of output buffer;
nOutBufferSize = sizeof(PVOID); // size of output buffer;
lpBytesReturned =&cbReturned; // address of actual bytes of output
```

Parameters

lpInBuffer

Points to an input buffer. Not used with this operation. Set to NULL.

nInBufferSize

Specifies the size, in bytes, of the buffer pointed to by *lpInBuffer*. Not used with this operation. Set to zero.

lpOutBuffer

Points to the PCI Memory Config buffer .

nOutBufferSize

Specifies the size, in bytes, of the buffer pointed to by *lpOutBuffer*.

lpBytesReturned

Points to a **DWORD** that receives the actual size, in bytes, of the data stored into *lpOutBuffer*.

Return Values

If the operation succeeds and the device **PCI102** is accessible, **DeviceIoControl** returns TRUE.

If the operation fails, **DeviceIoControl** returns FALSE.

IOCTL_PCI102_MAP_MEMORY_RAM_CFG_102

map la mémoire de configuration PCI102, cette opération est utilisée à l'initialisation de l'application,

elle permet d'accéder à cette mémoire grâce au pointeur retourné.

```
dwIoControlCode = IOCTL_PCI102_MAP_MEMORY_RAM_CFG_102;
    // operation code
lpInBuffer = NULL; // address of input buffer; not used; must be NULL
nInBufferSize = 0; // size of input buffer; not used; must be zero
lpOutBuffer = &pMemCfg102; // address of output buffer;
nOutBufferSize = sizeof(PVOID); // size of output buffer;
lpBytesReturned =&cbReturned; // address of actual bytes of output
```

Parameters

lpInBuffer

Points to an input buffer. Not used with this operation. Set to NULL.

nInBufferSize

Specifies the size, in bytes, of the buffer pointed to by *lpInBuffer*. Not used with this operation. Set to zero.

lpOutBuffer

Points to the PCI102 Memory Config buffer .

nOutBufferSize

Specifies the size, in bytes, of the buffer pointed to by *lpOutBuffer*.

lpBytesReturned

Points to a **DWORD** that receives the actual size, in bytes, of the data stored into *lpOutBuffer*.

Return Values

If the operation succeeds and the device PCI102 is accessible, **DeviceIoControl** returns TRUE.

If the operation fails, **DeviceIoControl** returns FALSE.

IOCTL_PCI102_MAP_MEMORY_RAM_MESURE

map la mémoire mesure de la carte PCI102, cette opération est utilisée à l'initialisation de l'application,

elle permet d'accéder à cette mémoire grâce au pointeur retourné.

```
dwIoControlCode = IOCTL_PCI102_MAP_MEMORY_RAM_MESURE;  
// operation code  
lpInBuffer = NULL; // address of input buffer; not used; must be NULL  
nInBufferSize = 0; // size of input buffer; not used; must be zero  
lpOutBuffer = &pMemRamMesure; // address of output buffer;  
nOutBufferSize = sizeof(PVOID); // size of output buffer;  
lpBytesReturned =&cbReturned; // address of actual bytes of output
```

Parameters

lpInBuffer

Points to an input buffer. Not used with this operation. Set to NULL.

nInBufferSize

Specifies the size, in bytes, of the buffer pointed to by *lpInBuffer*. Not used with this operation. Set to zero.

lpOutBuffer

Points to the PCI102 Memory of Measure buffer .

nOutBufferSize

Specifies the size, in bytes, of the buffer pointed to by *lpOutBuffer*.

lpBytesReturned

Points to a **DWORD** that receives the actual size, in bytes, of the data stored into *lpOutBuffer*.

Return Values

If the operation succeeds and the device PCI102 is accessible, **DeviceIoControl** returns TRUE.

If the operation fails, **DeviceIoControl** returns FALSE.

IOCTL_PCI102_UNMAP_MEMORY

Libère la mémoire mappée, cette opération est utilisée généralement à la sortie de l'application.

Cette fonction est la fonction inverse des fonctions suivantes :

IOCTL_PCI102_MAP_MEMORY_RAM_CFG_PCI
IOCTL_PCI102_MAP_MEMORY_RAM_CFG_102
IOCTL_PCI102_MAP_MEMORY_RAM_MESURE

```
dwIoControlCode = IOCTL_PCI102_UNMAP_MEMORY;  
// operation code  
lpInBuffer = &pMemRam; // address of input buffer;  
nInBufferSize = sizeof(PVOID); // size of input buffer;  
lpOutBuffer = NULL; // address of output buffer; not used; must be NULL  
nOutBufferSize = 0; // size of output buffer; not used; must be zero  
lpBytesReturned = &cbReturned; // address of actual bytes of output
```

Parameters

lpInBuffer

Points to an input buffer. (&pMemCfgPCI, &pMemCfg102, &pMemRamMeasure)

nInBufferSize

Specifies the size, in bytes, of the buffer pointed to by *lpInBuffer*. Not used with this operation. Set to zero.

lpOutBuffer

Points to the PCI102 Memory of Measure buffer .

nOutBufferSize

Specifies the size, in bytes, of the buffer pointed to by *lpOutBuffer*.

lpBytesReturned

Points to a **DWORD** that receives the actual size, in bytes, of the data stored into *lpOutBuffer*.

Return Values

If the operation succeeds and the device PCI102 is accessible, **DeviceIoControl** returns TRUE.

If the operation fails, **DeviceIoControl** returns FALSE.

IOCTL_PCI102_MAP_MEMORY_MASTER

Map la mémoire pointée par les accès Bus Master de la carte PCI102, cette opération est utilisée à l'initialisation de l'application,

elle permet d'accéder à cette mémoire grâce au pointeur retourné.

```
dwIoControlCode = IOCTL_PCI102_MAP_MEMORY_MASTER;  
// operation code  
lpInBuffer = NULL; // address of input buffer; not used; must be NULL  
nInBufferSize = 0; // size of input buffer; not used; must be zero  
lpOutBuffer = &pMaster; // address of output buffer;  
nOutBufferSize = sizeof(PVOID); // size of output buffer;  
lpBytesReturned =&cbReturned; // address of actual bytes of output
```

Parameters

lpInBuffer

Points to an input buffer. Not used with this operation. Set to NULL.

nInBufferSize

Specifies the size, in bytes, of the buffer pointed to by *lpInBuffer*. Not used with this operation. Set to zero.

lpOutBuffer

Points to the PCI102 MemoryBus Master buffer .

nOutBufferSize

Specifies the size, in bytes, of the buffer pointed to by *lpOutBuffer*.

lpBytesReturned

Points to a **DWORD** that receives the actual size, in bytes, of the data stored into *lpOutBuffer*.

Return Values

If the operation succeeds and the device PCI102 is accessible, **DeviceIoControl** returns TRUE.

If the operation fails, **DeviceIoControl** returns FALSE.

IOCTL_PCI102_READ_MASTER_MEMORY

Permet de lire le contenu de la mémoire pointée par les accès Bus Master de la carte PCI102.

```
dwIoControlCode = IOCTL_PCI102_READ_MASTER_MEMORY;
// operation code
lpInBuffer = BufferR; // address of input buffer;
nInBufferSize = 4 * nbByteTransfer; // size of input buffer; 512 Koctets max
lpOutBuffer = BufferW; // address of output buffer;
nOutBufferSize = 4 * 131072; // size of output buffer; 512 Koctets max
lpBytesReturned = &NumberOfBytesRead; // address of number of bytes read
```

Parameters

lpInBuffer

Points to an input buffer. You send an index for each value return.

nInBufferSize

Specifies the size, in bytes, of the buffer pointed to by *lpInBuffer*.

lpOutBuffer

Points to an output buffer.

nOutBufferSize

Specifies the size, in bytes, of the buffer pointed to by *lpOutBuffer*.

lpBytesReturned

Points to a **DWORD** that receives the actual size, in bytes, of the data stored into *lpOutBuffer*.

Return Values

If the operation succeeds and the device PCI102 is accessible, **DeviceloControl** returns TRUE.

If the operation fails, **DeviceloControl** returns FALSE.

Examples

You want to read adresse 0 2 4 7 of the RAM Master you can do :

...

```
BufferRW[0] = 0x0 ;
BufferRW[1] = 0x2 ;
BufferRW[2] = 0x4 ;
BufferRW[3] = 0x7 ;
```

```
DeviceloControl (
    hDevice,
    IOCTL_PCI102_READ_MASTER_MEMORY,
    BufferRW,
    4*4, // number of byte transfer
    BufferRW,
    4*131072, // =512 Koctets max
    &NumberOfBytesRead, // address of number of bytes read
    NULL // address of structure for data
);
```

....

IOCTL_PCI102_MASTER_NEW_BLOCK

Permet de relancer un nouveau transfert de bloque, de la carte PCI102 vers la mémoire UC.

On paramètre le nombre d'octets à transférer.

Cette opération s'effectue uniquement en mode Master.

```
dwIoControlCode = IOCTL_PCI102_MASTER_NEW_BLOCK;
Δ#éd 2 [ // operation code
lpInBuffer = BufferR; // address of input buffer;
// Represent the Master Write Transfer Count.

nInBufferSize = 4 * 1; // Size of the Transfer Count.
Δ#éd 2 ]
lpOutBuffer = NULL; // address of output buffer; Not used with this operation. Set to NULL.
nOutBufferSize = 0 ; // size of output buffer; Not used with this operation. Set to zero.
lpBytesReturned = &cbReturned; // address of number of bytes read
```

Parameters

```
Δ#éd 2 [
lpInBuffer
Points to an input buffer. Represent the Master write Transfer Count.
Δ#éd 2 ]
nInBufferSize
Specifies the size, in bytes, of the buffer pointed to by lpInBuffer. Not used with this
operation. Set to zero.
lpOutBuffer
Points to an output buffer. Not used with this operation. Set to NULL.
nOutBufferSize
Specifies the size, in bytes, of the buffer pointed to by lpOutBuffer. Not used with this
operation. Set to zero.
lpBytesReturned
Points to a DWORD that receives the actual size, in bytes, of the data stored into
lpOutBuffer.
```

Return Values

If the operation succeeds and the device PCI102 is accessible, **DeviceloControl** returns TRUE.

If the operation fails, **DeviceloControl** returns FALSE.

IOCTL_PCI102_START

Lance la commande START de la carte.

```
dwIoControlCode = IOCTL_PCI102_START;  
                // operation code  
lpInBuffer = NULL; // Not used with this operation. Set to NULL.  
nInBufferSize = 0; // Not used with this operation. Set to zero.  
lpOutBuffer = NULL; // Not used with this operation. Set to NULL.  
nOutBufferSize = 0; // Not used with this operation. Set to zero.  
lpBytesReturned = &cbReturned; // address of actual bytes of output
```

Parameters

lpInBuffer

Points to an input buffer. Not used with this operation. Set to NULL.

nInBufferSize

Specifies the size, in bytes, of the buffer pointed to by *lpInBuffer*. Not used with this operation. Set to zero.

lpOutBuffer

Points to an output buffer. Not used with this operation. Set to NULL.

nOutBufferSize

Specifies the size, in bytes, of the buffer pointed to by *lpOutBuffer*. Not used with this operation. Set to zero.

lpBytesReturned

Points to a **DWORD** that receives the actual size, in bytes, of the data stored into *lpOutBuffer*.

Return Values

If the operation succeeds and the device PCI102 is accessible, **DeviceloControl** returns TRUE.

If the operation fails, **DeviceloControl** returns FALSE.

IOCTL_PCI102_STOP

Lance la commande STOP de la carte.

```
dwIoControlCode = IOCTL_PCI102_STOP;  
                // operation code  
lpInBuffer = NULL; // Not used with this operation. Set to NULL.  
nInBufferSize = 0; // Not used with this operation. Set to zero.  
lpOutBuffer = NULL; // Not used with this operation. Set to NULL.  
nOutBufferSize = 0; // Not used with this operation. Set to zero.  
lpBytesReturned =&cbReturned; // address of actual bytes of output
```

Parameters

lpInBuffer

Points to an input buffer. Not used with this operation. Set to NULL.

nInBufferSize

Specifies the size, in bytes, of the buffer pointed to by *lpInBuffer*. Not used with this operation. Set to zero.

lpOutBuffer

Points to an output buffer. Not used with this operation. Set to NULL.

nOutBufferSize

Specifies the size, in bytes, of the buffer pointed to by *lpOutBuffer*. Not used with this operation. Set to zero.

lpBytesReturned

Points to a **DWORD** that receives the actual size, in bytes, of the data stored into *lpOutBuffer*.

Return Values

If the operation succeeds and the device PCI102 is accessible, **DeviceIoControl** returns TRUE.

If the operation fails, **DeviceIoControl** returns FALSE.

IOCTL_PCI102_TRIG

Lance la commande TRIG de la carte.

```
dwIoControlCode = IOCTL_PCI102_TRIG;  
                // operation code  
lpInBuffer = NULL; // Not used with this operation. Set to NULL.  
nInBufferSize = 0; // Not used with this operation. Set to zero.  
lpOutBuffer = NULL; // Not used with this operation. Set to NULL.  
nOutBufferSize = 0; // Not used with this operation. Set to zero.  
lpBytesReturned =&cbReturned; // address of actual bytes of output
```

Parameters

lpInBuffer

Points to an input buffer. Not used with this operation. Set to NULL.

nInBufferSize

Specifies the size, in bytes, of the buffer pointed to by *lpInBuffer*. Not used with this operation. Set to zero.

lpOutBuffer

Points to an output buffer. Not used with this operation. Set to NULL.

nOutBufferSize

Specifies the size, in bytes, of the buffer pointed to by *lpOutBuffer*. Not used with this operation. Set to zero.

lpBytesReturned

Points to a **DWORD** that receives the actual size, in bytes, of the data stored into *lpOutBuffer*.

Return Values

If the operation succeeds and the device PCI102 is accessible, **DeviceIoControl** returns TRUE.

If the operation fails, **DeviceIoControl** returns FALSE.

IOCTL_PCI102_MOTIF_NB_VOIES

Configure la carte pour le nombre de voies voulue et lance le bloque transfert.
Avec un reset carte et positionnement en mode Master
pour n voies n avec $n \in [3, \dots, 256]$ suivant le nombre de voies utilisable sur l'PCI102.
pour 0x8000 motifs
pour une fréquence d'acquisition à 40 KHz
avec dans chaque motif un mapping des voies du type
[V254, V255, V2, V3, ..., Vn] Attention la numérotation des voies commence à 0.

```
dwIoControlCode = IOCTL_PCI102_MOTIF_NB_VOIES;  
                // operation code  
lpInBuffer = BufferR; // // address of input buffer; Represent the number of way  
nInBufferSize = 4 * 1; // Size of the number.  
lpOutBuffer = NULL; // Not used with this operation. Set to NULL.  
nOutBufferSize = 0; // Not used with this operation. Set to zero.  
lpBytesReturned =&cbReturned; // address of actual bytes of output
```

Parameters

lpInBuffer

Points to an input buffer. Represent the number of way

nInBufferSize

Specifies the size, in bytes, of the buffer pointed to by *lpInBuffer*. Not used with this operation. Set to zero.

lpOutBuffer

Points to an output buffer. Not used with this operation. Set to NULL.

nOutBufferSize

Specifies the size, in bytes, of the buffer pointed to by *lpOutBuffer*. Not used with this operation. Set to zero.

lpBytesReturned

Points to a **DWORD** that receives the actual size, in bytes, of the data stored into *lpOutBuffer*.

Return Values

If the operation succeeds and the device PCI102 is accessible, **DeviceloControl** returns TRUE.

If the operation fails, **DeviceloControl** returns FALSE.

IOCTL_PCI102_RESET_AMCC_MASTER_INIT

Cette commande réinitialise la carte PCI102 et positionne l'AMCC dans le mode Master.

```
dwIoControlCode = IOCTL_PCI102_RESET_AMCC_MASTER_INIT;  
                // operation code  
lpInBuffer = NULL; // Not used with this operation. Set to NULL.  
nInBufferSize = 0; // Not used with this operation. Set to zero.  
lpOutBuffer = NULL; // Not used with this operation. Set to NULL.  
nOutBufferSize = 0; // Not used with this operation. Set to zero.  
lpBytesReturned =&cbReturned; // address of actual bytes of output
```

Parameters

lpInBuffer

Points to an input buffer. Not used with this operation. Set to NULL.

nInBufferSize

Specifies the size, in bytes, of the buffer pointed to by *lpInBuffer*. Not used with this operation. Set to zero.

lpOutBuffer

Points to an output buffer. Not used with this operation. Set to NULL.

nOutBufferSize

Specifies the size, in bytes, of the buffer pointed to by *lpOutBuffer*. Not used with this operation. Set to zero.

lpBytesReturned

Points to a **DWORD** that receives the actual size, in bytes, of the data stored into *lpOutBuffer*.

Return Values

If the operation succeeds and the device PCI102 is accessible, **DeviceIoControl** returns TRUE.

If the operation fails, **DeviceIoControl** returns FALSE.

IOCTL_PCI102_INTERRUPTION

Permet d'obtenir un rapport sur les interruptions .

Ce rapport est remis a zéro une fois l'opération terminée.

Les valeurs de retour sont :

BufferRW [0]=InterruptCount; nombre total d'interruptions reçues depuis le dernier rapport

```
dwIoControlCode = IOCTL_PCI102_INTERRUPTION;
// operation code
lpInBuffer = NULL; // address of input buffer; Not used with this operation. Set to
NULL.
nInBufferSize = 0; // size of input buffer; Not used with this operation. Set to zero.
lpOutBuffer = &BufferRW; // address of output buffer
nOutBufferSize = 4*1 ; // size of output buffer; 1 DWORDs return
lpBytesReturned = &cbReturned; // address of number of bytes read
```

Parameters

lpInBuffer

Points to an input buffer. Not used with this operation. Set to NULL.

nInBufferSize

Specifies the size, in bytes, of the buffer pointed to by *lpInBuffer*. Not used with this operation. Set to zero.

lpOutBuffer

Points to an output buffer.

nOutBufferSize

Specifies the size, in bytes, of the buffer pointed to by *lpOutBuffer*.

The driver return 1 DWORDs

lpBytesReturned

Points to a **DWORD** that receives the actual size, in bytes, of the data stored into *lpOutBuffer*.

Return Values

If the operation succeeds and the device **PCI 102** is accessible, **DeviceloControl** returns TRUE.

The **DeviceloControl** couldn't return FALSE.

IOCTL_PCI102_NUM_VERSION

Cette commande retourne le numéro de version du driver de la carte.

Ce numéro comporte 4 octets.

Il est codé en ASCII suivant l'exemple :

BufferRW= 0x56313130; soit V 1 1 0

```
dwIoControlCode = IOCTL_PCI102_NUM_VERSION;
                // operation code
lpInBuffer = NULL; // Not used with this operation. Set to NULL.
nInBufferSize = 0; // Not used with this operation. Set to zero.
lpOutBuffer = &BufferRW; // address of output buffer
nOutBufferSize = 4*1 ; // size of output buffer; 1 DWORDs return
lpBytesReturned =&cbReturned; // address of actual bytes of output
```

Parameters

lpInBuffer

Points to an input buffer. Not used with this operation. Set to NULL.

nInBufferSize

Specifies the size, in bytes, of the buffer pointed to by *lpInBuffer*. Not used with this operation. Set to zero.

lpOutBuffer

Points to an output buffer.

nOutBufferSize

Specifies the size, in bytes, of the buffer pointed to by *lpOutBuffer*.

lpBytesReturned

Points to a **DWORD** that receives the actual size, in bytes, of the data stored into *lpOutBuffer*.

Return Values

The operation always succeeds and the device **PCI 102** is accessible, **DeviceIoControl** returns TRUE.

APPLI.EXE :

Ce programme est fourni sur la disquette.

Il reprend les codes IOCTL dans des exemples simples, et vous permettra avec son code source de démarrer votre projet rapidement.

Le programme source peut être directement compilé avec la chaîne Microsoft Developer Studio.

ANNEXE :

Memory Address Map

0-9ffffh	Base Memory	640 KB
A0000h-Bffffh	Video RAM	128 KB
C0000h-C7ffffh	Video BIOS	32 KB
C8000h-Dffffh	BIOS Extension ROM	96 KB
E0000h-Effffh	BIOS Extension ROM	64 KB
F0000h-Fffffh	System BIOS ROMs In Real mode	64 KB
100000h-FFF0000h	System RAM	~ 4 GB
FFFF0000h-FFFFFFFFh	System BIOS ROM After hard RESET	64 KB



Ressource éventuellement exploitable par une carte ISA

I/O Address Map

000-01F	DMA controller 1
020-03f	Interrupt controller (Master)
040-05F	Interval Timer
060-06F	Keyboard controller
070-07F	Real Time Clock RTC
080-09F	DMA page register
0A0-0BF	Interrupt controller (Slave)
0C0-0DF	DMA controller 2
0F0	Clear Math co-processor busy
0F1	Reset Math co-processor
0F2-0FF	Math co-processor
170-177	Secondary IDE controller
1F0-1F7	Primary IDE controller
220-22F	Audio
278-27E	LPT2
2E8-2EF	COM4
2F8-2FF	COM2
300-31F	Prototype Card
360-36F	Network
378-37E	LPT1
388-38B	Audio synthesizer
3BC-3BE	LPT3
3E8-3EF	COM3
3F0-3F7	Diskette controller
3F8-3FF	COM1
4D0	Edge/level control register INTCNTRL1
4D1	Edge/level control register INTCNTRL2
0534 – 0537	Windows Sound System- compatible
CF8-CFF	PCI configure space control register

Computer Interrupt Levels

IRQ0	Timer Tick
IRQ1	Keyboard Controller
IRQ2	Cascade interrupt
IRQ3	COM2, COM4
IRQ4	COM1, COM3
IRQ5	Audio
IRQ6	Diskette
IRQ7	LPT1, LPT3
IRQ8	Real time clock
IRQ9	(PCI device) default video
IRQ10	(PCI device) default SCSI
IRQ11	(PCI device) default audio
IRQ12	Mouse interrupt
IRQ13	Math co-processor
IRQ14	IDE primary
IRQ15	IDE secondary

