

NOTES:

DRIVER-2K-XP-PCI 2X8

SOMMAIRE

| | | |
|-------------------|---|-----------|
| Chapitre A | Installation du matériel | 4 |
| Chapitre B | Installation du logiciel..... | 5 |
| Chapitre C | Guide de survie..... | 11 |
| Chapitre D | Fonctions logicielles | 12 |
| | IOCTL_PCI2X8_MAP_MEMORY_RAM_CFG_PCI | 14 |
| | IOCTL_PCI2X8_MAP_MEMORY_RAM..... | 15 |
| | IOCTL_PCI2X8_UNMAP_MEMORY | 16 |
| | IOCTL_PCI2X8_RESET_AMCC_INIT | 17 |
| | IOCTL_PCI2X8_INTERRUPTIO..... | 18 |
| | IOCTL_PCI2X8_WAIT_MICROSECONDES | 20 |
| | IOCTL_PCI2X8_NB_DEVICE | 21 |
| | IOCTL_PCI2X8_NUM_VERSION | 22 |
| APPLI.EXE | | 23 |
| Annexe | | 24 |

Chapitre A Installation du matériel



- ✚ 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 and Play » et vous dispensent de configurer l'interface avec l'ordinateur. (Voir documentation technique de chaque carte et le chapitre annexe).

- ✚ Mettez l'ordinateur sous tension.

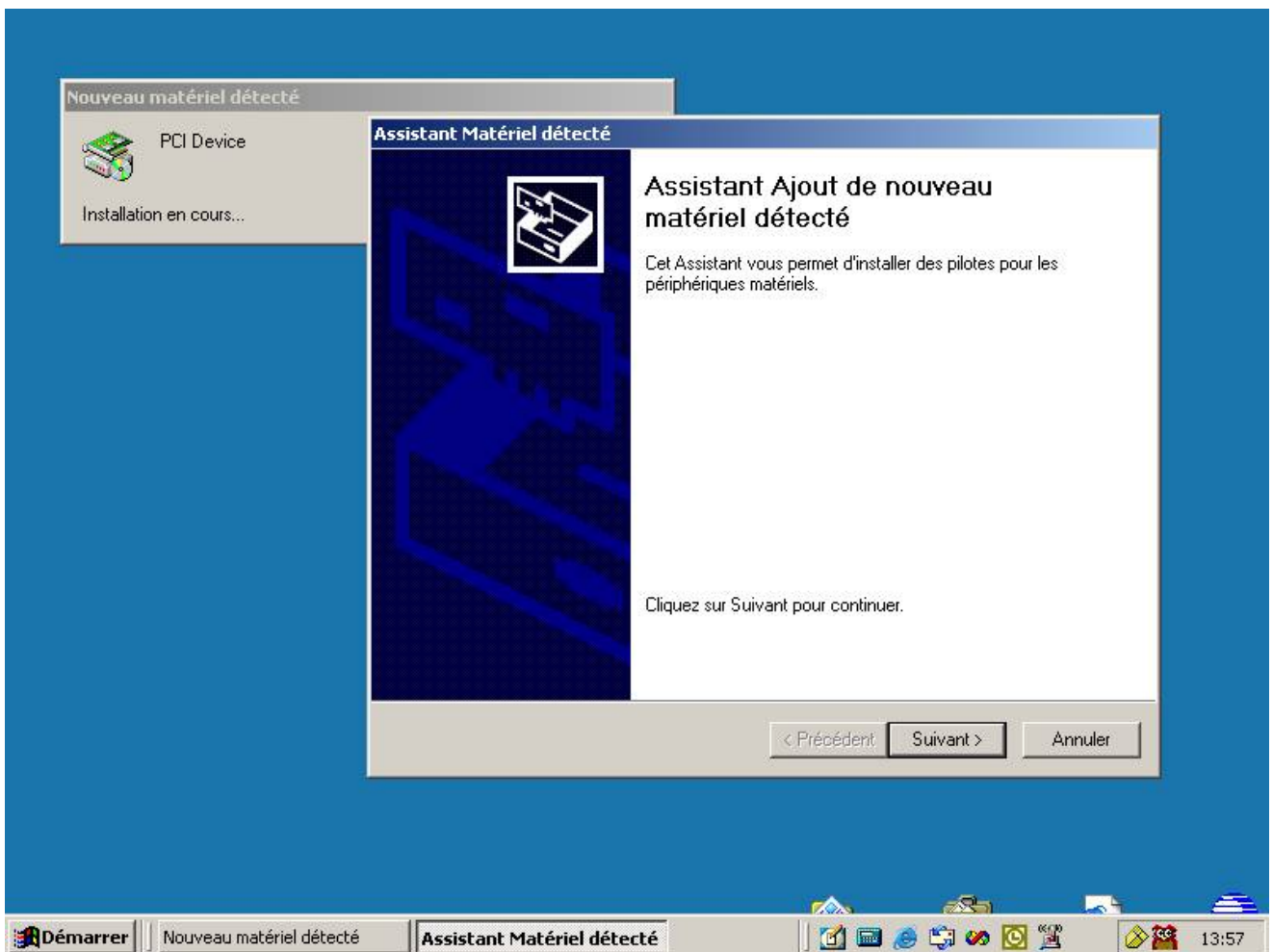


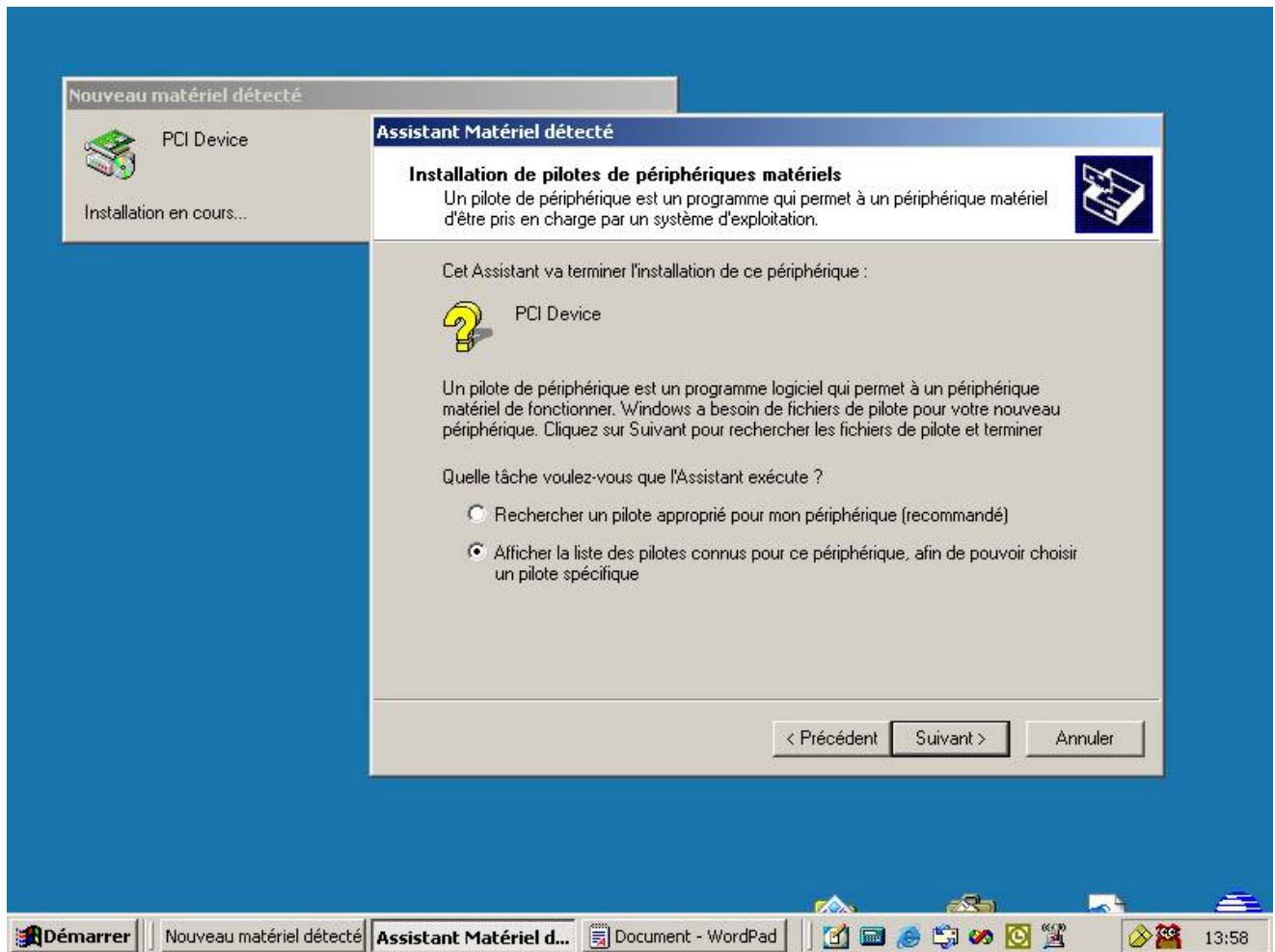
>>>Nota :

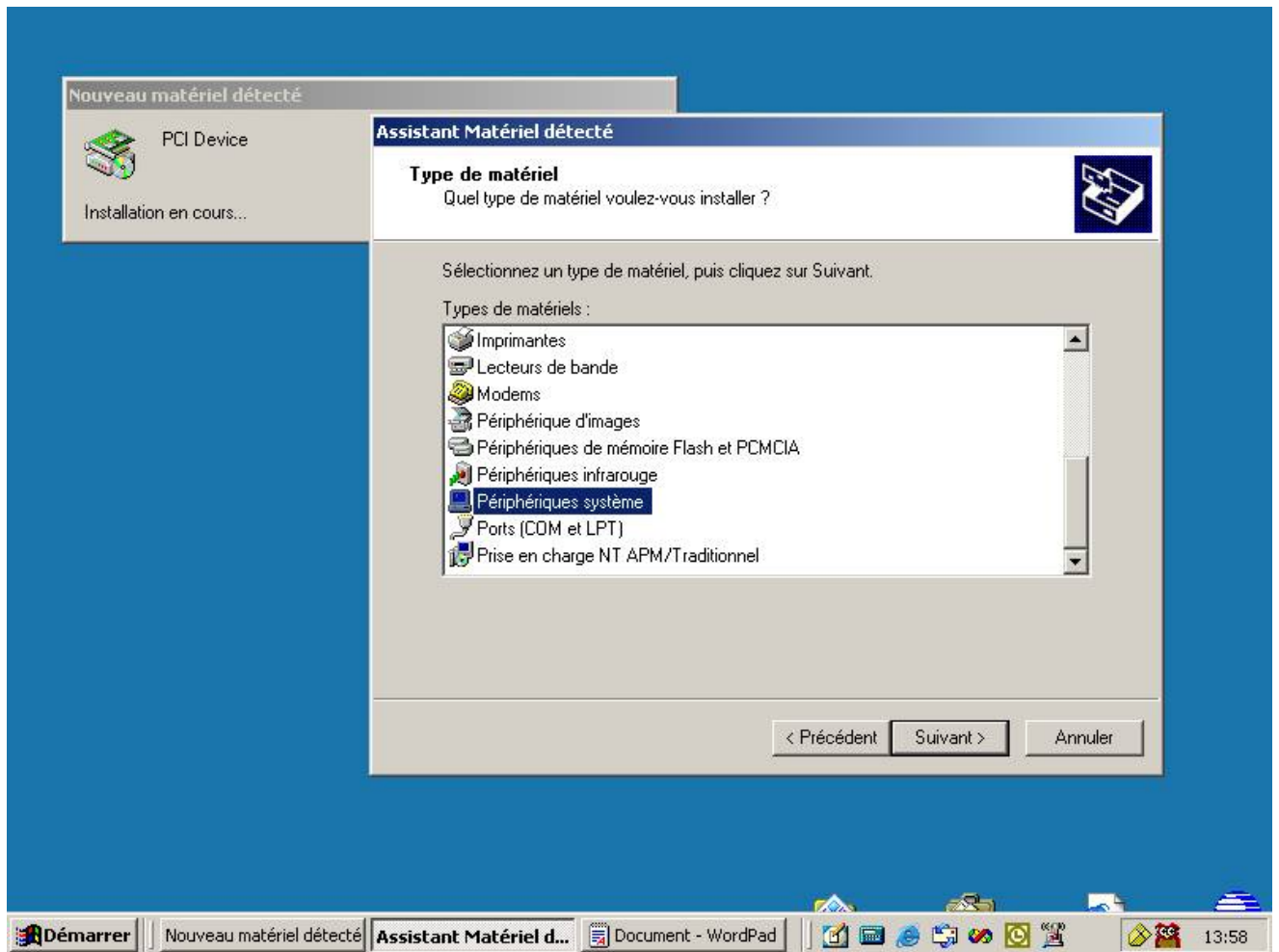
Par souci de simplification, et de bon déroulement de la procédure d'installation, nous vous conseillons de ne pas changer les autres cartes d'emplacement lors de l'ajout de la carte **PCI 2X8**

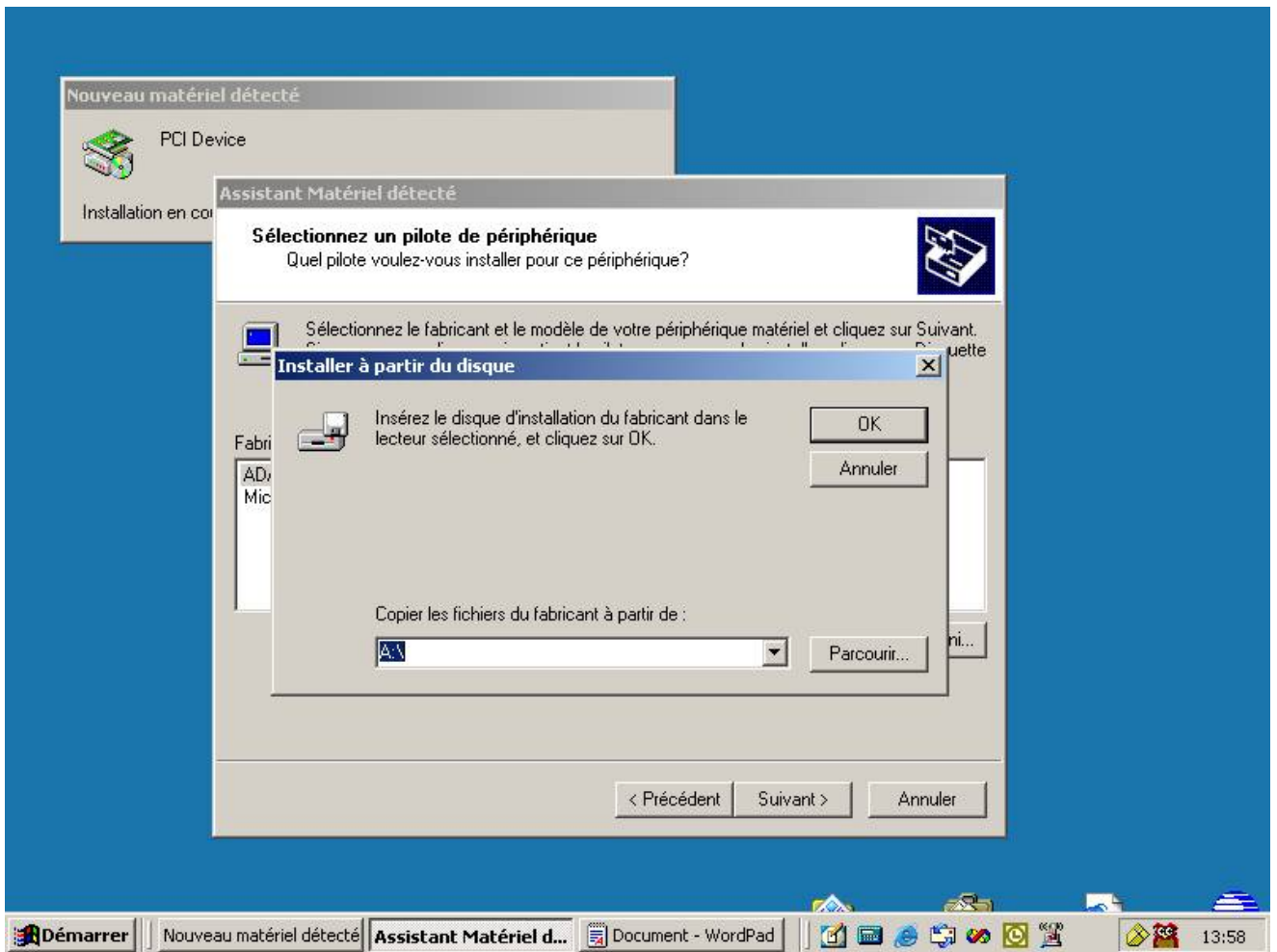
Chapitre B Installation du logiciel

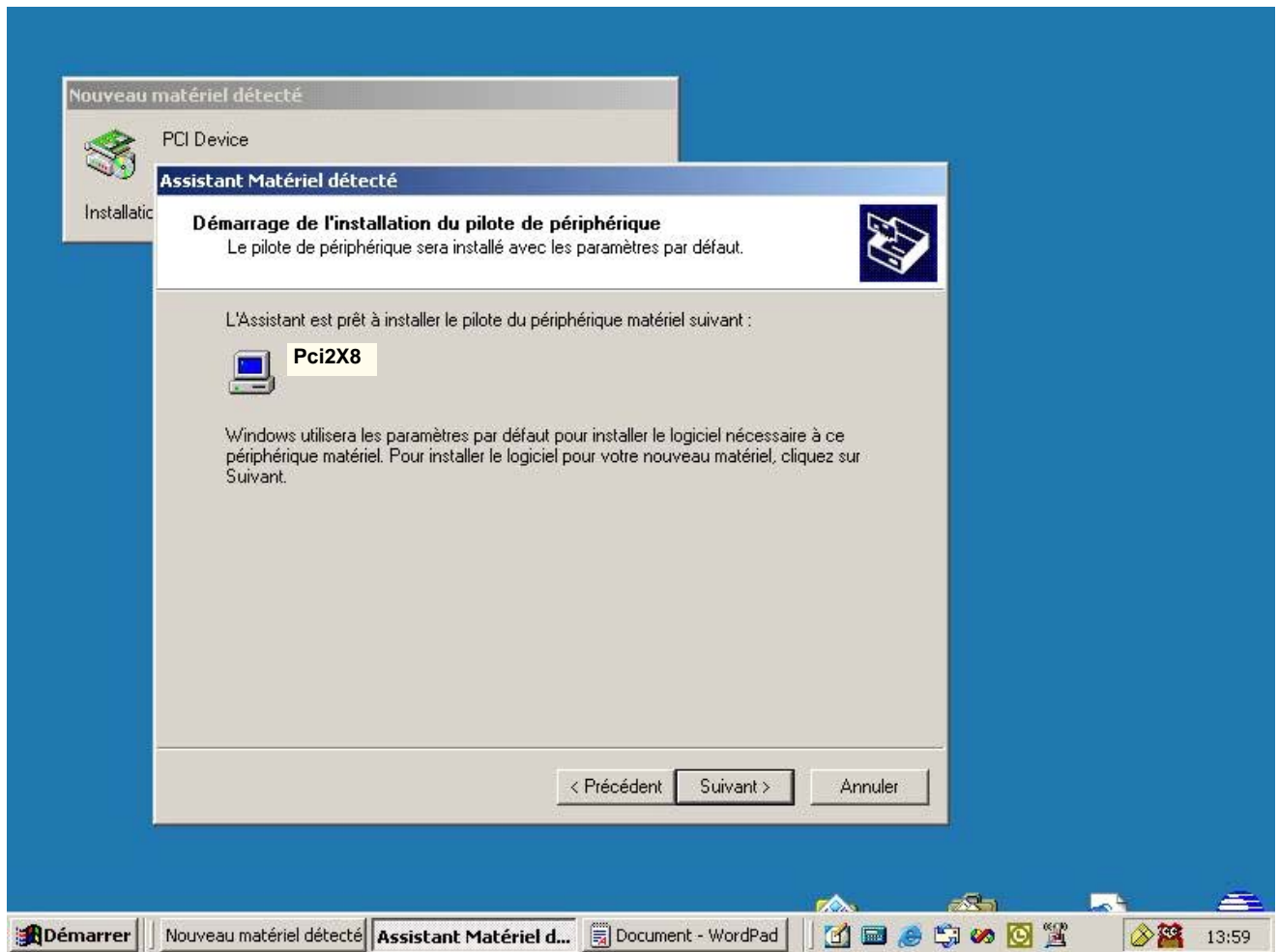
- ✚ Une fois que l'OS est booté son utilitaire va détecter le nouveau Device.
- ✚ A ce moment-là, suivre la procédure indiquée à l'écran pour ajouter le driver à partir de la disquette, c'est à dire :

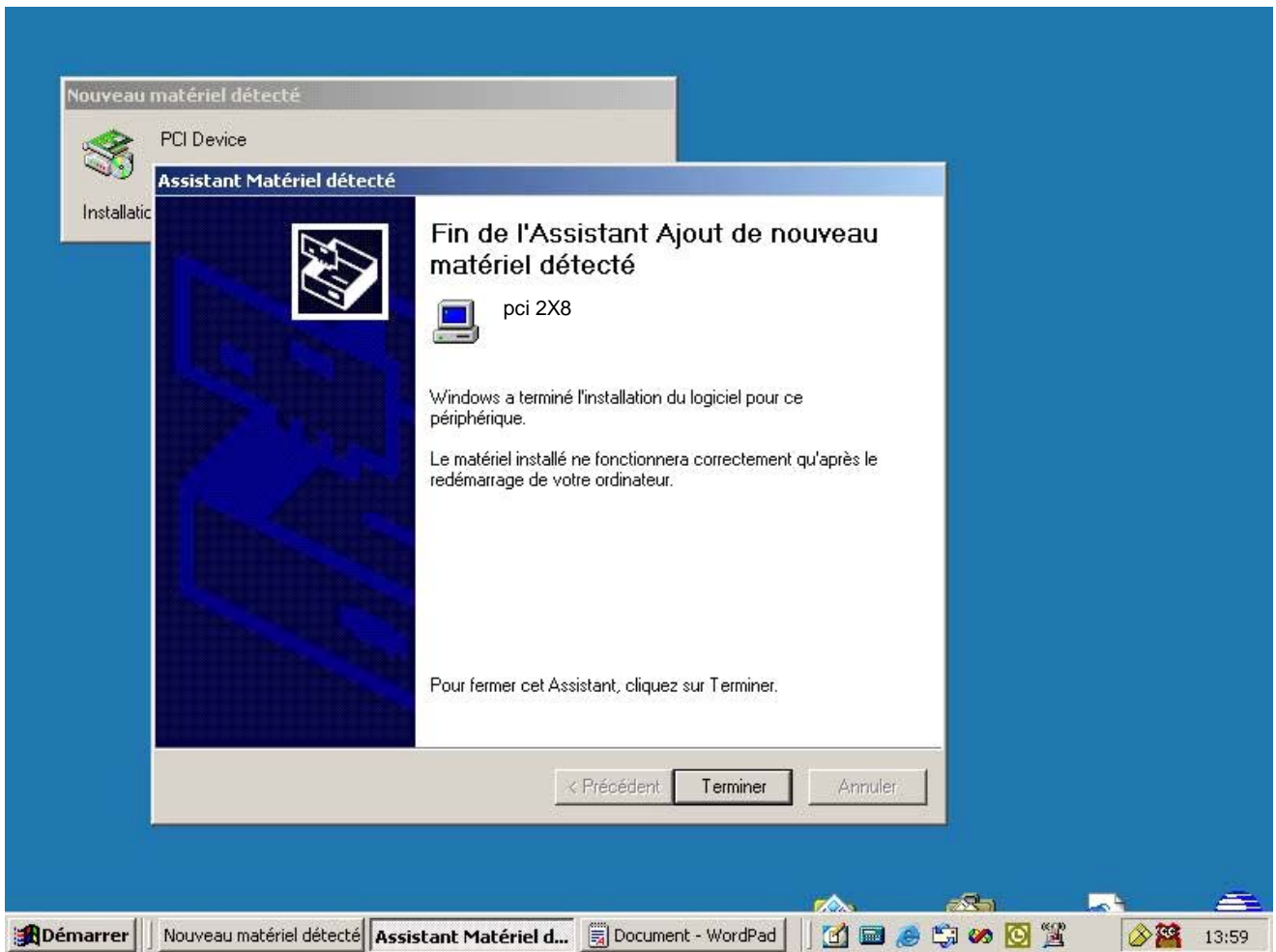












⇒ Redémarrez l'ordinateur :

Maintenant l'ordinateur a identifié le nouveau matériel et il est prêt à l'emploi.

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.

Nous vous indiquons que les anciens logiciels applicatifs (mode user) sont compatibles avec ce nouveau driver.

Une simple re-compilation est nécessaire.

Avec une carte **PCI**, et les OS Windows 2000 ou Windows XP vous ne devez pas rencontrer de problèmes.

Le « Plug and Play » est total aussi bien Hardware que Software.

>>>Rappels :

Windows 2000 ou Windows XP 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 **DeviceIoControl** 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 DeviceIoControl(

```
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
```

);



Si vous possédez des anciens IOCTL (sous Windows NT4) conservez-les. Les mêmes ont été reportés avec cette nouvelle version.

| IOCTL | Description | Éléments envoyés | Éléments reçus |
|--|--|-----------------------------------|---|
| IOCTL_PCI2X8_MAP_MEMORY_RAM_CFG_PCI | map l'espace des registres opérationnel PCI dans l'espace user | NULL | Pointeur sur l'espace désiré |
| IOCTL_PCI2X8_MAP_MEMORY_RAM | map l'espace memoire de la carte dans l'espace user | NULL | Pointeur sur l'espace désiré |
| IOCTL_PCI2X8_UNMAP_MEMORY | Unmap la mémoire réservée | Pointeur sur la mémoire à libérer | NULL |
| IOCTL_PCI2X8_RESET_AMCC_INIT | Reset de la carte | NULL | NULL |
| IOCTL_PCI2X8_INTERRUPTION | Rapport des interruptions | NULL | 7 valeurs 32 bits dans l'ordre : - InterruptCount; - Mailbox_1; - Mailbox_2; - Mailbox_3; - Mailbox_4; - Mailbox_1ou2ou3ou4; - Mbef; |
| IOCTL_PCI2X8_WAIT_MICROSECONDES | Génère une tempo de 1 Microsecondes minimum | NULL | NULL |
| IOCTL_PCI2X8_NB_DEVICE | Retourne le nombre de PCI 2X8 détectées | NULL | Pointeur sur le nombre de carte |
| IOCTL_PCI2X8_NUM_VERSION | Retourne le numéro de version du driver | NULL | Pointeur sur la version |

IOCTL_PCI2X8_MAP_MEMORY_RAM_CFG_PCI

map l'espace des registres opérationnel PCI dans l'espace user.

Cette opération est utilisée à l'initialisation de l'application.

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

```
dwIoControlCode = IOCTL_PCI2X8_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 .n 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 PCI2X8 Operation Register.

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 PCI 2X8 is accessible, `DeviceIoControl` returns `TRUE`.

If the operation fails, `DeviceIoControl` returns `FALSE`. (may be insufficient resource)

IOCTL_PCI2X8_MAP_MEMORY_RAM

map l'espace utilisateur de la carte dans l'espace user
cette opération est utilisée à l'initialisation de l'application,
Ce code permet d'accéder à cette mémoire grâce au pointeur retourné.

```
dwIoControlCode = IOCTL_PCI2X8_MAP_MEMORY_RAM;  
// 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 = & pMemIO2X8; // 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 PCI2X8 User Register.

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 PCI 2X8 is accessible, `DeviceIoControl` returns TRUE.

If the operation fails, `DeviceIoControl` returns FALSE. (may be insufficient resource)

IOCTL_PCI2X8_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_PCI2X8_MAP_MEMORY_RAM_CFG_PCI

IOCTL_PCI2X8_MAP_MEMORY_RAM

```
dwIoControlCode = IOCTL_PCI2X8_UNMAP_MEMORY;  
// operation code  
lpInBuffer = &pMem; // 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 the input buffer.

nInBufferSize

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

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 **PCI 2X8** is accessible, **DeviceloControl** returns TRUE.

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

IOCTL_PCI2X8_RESET_AMCC_INIT

Cette commande réinitialise la carte **PCI 2X8** par un reset de l'AMCC.

```
dwIoControlCode = IOCTL_PCI2X8_RESET_AMCC_INIT;  
// 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 = 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. 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 **PCI 2X8** is accessible, **DeviceloControl** returns TRUE.

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

IOCTL_PCI2X8_INTERRUPTION

Permet d'obtenir un rapport sur les interruptions.

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

Les valeurs de retour sont :

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

BufferRW [1] = Mailbox_1; nombre d'interruptions reçues dans la Mailbox1

BufferRW [2] = Mailbox_2; non utilisé par défaut

BufferRW [3] = Mailbox_3; non utilisé par défaut

BufferRW [4] = Mailbox_4; non utilisé par défaut

BufferRW [5] = Mailbox_1ou2ou3ou4; nombre d'interruptions reçues sur les Mailbox

BufferRW [6] = Mbef; Registre Mbef au moment de la dernière interruption prise en compte.

```
dwIoControlCode = IOCTL_PCI2X8_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*7 ; // size of output buffer; 7 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 7 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 2X8** is accessible, **DeviceloControl** returns TRUE.

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

IOCTL_PCI2X8_WAIT_MICROSECONDES

Cette commande permet d'utiliser une temporisation.

Elle est notamment utilisée pendant les opérations de changement des états des voies en sortie.

Nous rappelons qu'il y a un temps de commutations de quelques microsecondes sur les voies en sortie.

Cette commande ne donne pas un temps référentiel unique pour toutes les plates-formes.

Elle assure cependant un temps d'attente minimum de 1 microseconde.

```
dwIoControlCode = IOCTL_PCI2X8_WAIT_MICROSECONDES;  
// 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 **PCI 2X8** is accessible, **DeviceloControl** returns TRUE.

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

IOCTL_PCI2X8_NB_DEVICE

Cette commande retourne le nombre de **PCI 2X8** qui sont détectées par le Driver.

Ce numéro comporte 4 octets.

Le driver supporte au maximum 10 cartes numérotées de 0 à 9

```
dwIoControlCode = IOCTL_PCI2X8_NB_DEVICE ;  
// 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 2X8** is accessible, **DeviceloControl** returns TRUE.

IOCTL_PCI2X8_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_PCI2X8_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 2X8** 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 et modifié avec Microsoft

Visual C++ 6.0.

Annexe

PC 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 |



Ressource éventuellement exploitable par une carte PCI