



### *Licence pour le logiciel ADAS Electronique :*

- Ce logiciel est employé pour configurer et utiliser les ressources de la carte conformément à la documentation technique du produit.  
Il est conforme aux exigences et aux possibilités exceptionnelles du produit HARDWARE.
- Grâce à ce produit, nous gérons tous les particularismes inhérents aux systèmes électroniques et les pièges à éviter, ainsi vous pourrez vous consacrer directement à votre projet (application).
- La copie de ce logiciel est interdite.
- Désassembler, décompiler ou décoder ce logiciel est interdit.
- L'utilisateur doit acquérir une licence logicielle pour l'utilisation du driver de la carte.
- Ce logiciel fonctionne sous Windows NT4 et peut gérer jusqu'à 10 cartes.

**Title:**  
Titre :

# DRIVER-NT4-PCI 307

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

Written by  M. ROCHE  on  22 / 06  Visa    
 Revised by  B. THOUËNON  on  22 / 06  Visa    
 Approved by  D. PIMONT  on  22 / 06  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.

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

## S O M M A I R E

A.	INSTALLATION DU MATERIEL.....	4
B.	INSTALLATION DU LOGICIEL .....	5
C.	GUIDE DE SURVIE .....	6
D.	FONCTIONS LOGICIELLES.....	7

## 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

⇒ Cliquez sur le bouton Add

⇒ 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.

## C. GUIDE DE SURVIE

Avec une carte **PCI**, vous ne devez pas rencontrer de problèmes.

Avec une carte **ISA**, si vous rencontrez un **écran bleu** (BSOD) après l'installation du driver, il faut orienter la recherche du problème vers :

- un conflit dans l'adressage de la carte IO et/ou mémoire
- un conflit dans les numéros d'interruption. Sur un Bus ISA, les interruptions ne sont pas partageables.

***Pour désactiver un driver qui se charge automatiquement à la mise sous-tension, il faut :***

⇒ Bootez sur une disquette DOS.

⇒ Ensuite allez dans le répertoire <winnt>/system32/drivers et détruisez le driver malheureux.

⇒ Rebootez le système normalement sur le disque. Le message d'erreur au moment du logon indique que le driver n'est plus là.

⇒ Relancez la procédure d'installation avec une nouvelle configuration adaptée aux ressources.

Pour choisir une ressource, on peut utiliser le programme qui se trouve dans Démarrer>

Programmes>

Outils d'administration(Commun)>

Diagnostics Windows NT

Une autre façon de le lancer est d'écrire dans la fenêtre Démarrer>

Exécuter winmsd

Dans l'onglet Ressources, on trouvera une liste de ressources utilisées. Attention, elle n'est pas forcément exhaustive.

## 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
);
```

<b>IOCTL</b>	<b>Description</b>	<b>Éléments envoyés</b>	<b>Éléments reçus</b>
<b>IOCTL_PCI307_MAP_MEMORY_RAM_CFG_PCI</b>	map l'espace des registres opérationnel PCI dans l'espace user	NULL	Pointeur sur l'espace désiré
<b>IOCTL_PCI307_MAP_MEMORY_RAM</b>	map l'espace utilisateur de la carte dans l'espace user	NULL	Pointeur sur l'espace désiré
<b>IOCTL_PCI307_UNMAP_MEMORY</b>	Unmap la mémoire réservée	Pointeur sur la mémoire à libérer	NULL
<b>IOCTL_PCI307_RESET_AMCC_INIT</b>	Reset de la carte	NULL	NULL
<b>IOCTL_PCI307_INTERRUPT</b>	Rapport des interruptions	NULL	7 valeurs 32 bits dans l'ordre : - InterruptCount; - Mailbox_1; - Mailbox_2; - Mailbox_3; - Mailbox_4; - Mailbox_1ou2ou3ou4; - Mbef;
<b>IOCTL_PCI307_NB_DEVICE</b>	Retourne le nombre de PCI 307 détectées	NULL	Pointeur sur le nombre de carte
<b>IOCTL_PCI307_NUM_VERSION</b>	Retourne le numéro de version du driver	NULL	Pointeur sur la version

## IOCTL\_PCI307\_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_PCI307_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 PCI307 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 307** is accessible, **DeviceIoControl** returns TRUE.

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

## IOCTL\_PCI307\_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_PCI307_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 = & pMemIO307; // 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 PCI307 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 307** is accessible, **DeviceIoControl** returns TRUE.

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

## IOCTL\_PCI307\_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\_PCI307\_MAP\_MEMORY\_RAM\_CFG\_PCI**  
**IOCTL\_PCI307\_MAP\_MEMORY\_RAM**

```
dwIoControlCode = IOCTL_PCI307_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 307** is accessible, **DeviceIoControl** returns TRUE.

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

## IOCTL\_PCI307\_RESET\_AMCC\_INIT

Cette commande réinitialise la carte **PCI 307** par un reset de l'AMCC.

```
dwIoControlCode = IOCTL_PCI307_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 307** is accessible, **DeviceIoControl** returns TRUE.

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

## IOCTL\_PCI307\_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_PCI307_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 307** is accessible, **DeviceloControl** returns TRUE.

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

## IOCTL\_ PCI307\_NB\_DEVICE

Cette commande retourne le nombre de **PCI 307** qui sont détectées par le Driver.  
Ce numéro comporte 4 octets.

```
dwIoControlCode = IOCTL_ PCI307_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 307** is accessible, **DeviceIoControl** returns TRUE.

## IOCTL\_PCI307\_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= 0x56313230; soit V 1 2 0

```
dwIoControlCode = IOCTL_PCI307_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 307** 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 la chaîne Microsoft Developer Studio.

Le driver peut gérer 10 cartes numérotées de 0 à 9.

Pour ce faire, on accède à la carte avec le handle (hDriver) retourné par la fonction CreateFile.

Syntaxe :

```
HDriver = CreateFile (“\\.\IntPci307_0”, GENERIC_READ | GENERIC_WRITE,  
0, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL) ;
```

IntPci307\_0 correspond à la carte 0

IntPci307\_1 correspond à la carte 1

...

IntPci307\_9 correspond à la carte 9

Ainsi, le handle retourné peut gérer toutes les ressources de la carte correspondante.

Les cartes sont numérotées de 0 à 9 suivant leur emplacement croissant (numéro de Bus puis numéro de slot).

**ANNEXE :**

**PC Memory Address Map**

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



**Ressource éventuellement exploitable par une carte ISA**

## PC 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
<b>300-31F</b>	<b>Prototype Card</b>
<b>360-36F</b>	<b>Network</b>
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



**Ressource éventuellement exploitable par une carte ISA**

## 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 ISA**