



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.

NOTES :

DRIVER-NT4-PCI LMC

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 and 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
);
```

IOCTL	Description	Éléments envoyés	Éléments reçus
IOCTL_PCILMC_MAP_MEMORY_RAM_CFG_PCI	map l'espace des registres opérationnel PCI dans l'espace user	NULL	Pointeur sur l'espace désiré
IOCTL_PCILMC_MAP_MEMORY_RAM_CFG_LMC	map l'espace I/O de la carte dans l'espace user	NULL	Pointeur sur l'espace désiré
IOCTL_PCILMC_MAP_MEMORY_RAM_LMC	map l'espace mémoire de la carte dans l'espace user	NULL	Pointeur sur l'espace désiré
IOCTL_PCILMC_UNMAP_MEMORY	Unmap la mémoire réservée	Pointeur sur la mémoire à libérer	NULL
IOCTL_PCILMC_RESET_AMCC_INIT	Reset de la carte	NULL	NULL
IOCTL_PCILMC_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_PCILMC_NB_DEVICE	Retourne le nombre de PCI LMC détectées	NULL	Pointeur sur le nombre de carte
IOCTL_PCILMC_NUM_VERSION	Retourne le numéro de version du driver	NULL	Pointeur sur la version

IOCTL_PCILMC_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_PCILMC_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 **PCILMC** 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 LMC** is accessible, **DeviceIoControl** returns TRUE.

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

IOCTL_PCILMC_MAP_MEMORY_RAM_CFG_LMC

map l'espace I/O (short I/O) de la carte dans l'espace user (fenêtre de 1 KO)

cette opération est utilisée à l'initialisation de l'application,

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

```
dwIoControlCode = IOCTL_PCILMC_MAP_MEMORY_RAM_CFG_LMC;  
                // 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 = & pMemIOLMC; // 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 **PCILMC** 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 LMC** is accessible, **DeviceIoControl** returns TRUE.

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

IOCTL_PCILMC_MAP_MEMORY_RAM_LMC

map l'espace mémoire de la carte dans l'espace user (fenêtre de 4 MO)

cette opération est utilisée à l'initialisation de l'application,

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

```
dwIoControlCode = IOCTL_PCILMC_MAP_MEMORY_RAM_LMC;  
                // 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 = & pMemRAM_LMC; // 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 PCILMC User Memory.

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 LMC** is accessible, **DeviceIoControl** returns TRUE.

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

IOCTL_PCILMC_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_PCILMC_MAP_MEMORY_RAM_CFG_PCI  
IOCTL_PCILMC_MAP_MEMORY_RAM_CFG_LMC  
IOCTL_PCILMC_MAP_MEMORY_RAM_LMC
```

```
dwIoControlCode = IOCTL_PCILMC_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 LMC** is accessible, **DeviceIoControl** returns TRUE.

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

IOCTL_PCILMC_RESET_AMCC_INIT

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

```
dwIoControlCode = IOCTL_PCILMC_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 LMC** is accessible, **DeviceIoControl** returns TRUE.

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

IOCTL_PCILMC_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

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_PCILMC_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 LMC** is accessible, **DeviceloControl** returns TRUE.

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

IOCTL_ PCILMC_NB_DEVICE

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

```
dwIoControlCode = IOCTL_ PCILMC_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 LMC** is accessible, **DeviceIoControl** returns TRUE.

IOCTL_PCILMC_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_PCILMC_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 LMC** 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.

Ce programme est spécifiquement un exemple d'utilisation des modules **LMC ESCC**, et des modules **LMC ADC**.

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

ANNEXE :

PC 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	64 KB
	In Real mode	
100000h-FFF0000h	System RAM	~ 4 GB
FFFF0000h-FFFFFFFFh	System BIOS ROM After hard RESET	64 KB



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
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



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