Woodhead
Woodhead Industries, Inc.

applicomIO® O.S. Drivers Solution

*For **ETS, Linux, MS-DOS, QNX, VxWorks, WinCE .net***

-= Documentation =-

June 2003

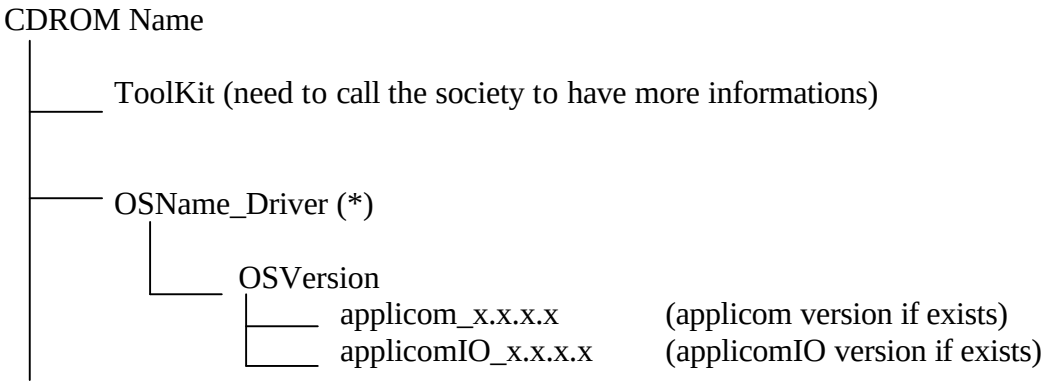
SUMMARY

CD-ROM CONTAINS

This CD-ROM contains applicom solution for:

O.S. Supported	O.S. Version
ETS	10.1
Linux	2.4.21
MS-DOS	6.20
QNX	6.1 / 6.2
VxWorks	5.4.2
WinCE .net	4.1

The Folder architecture is define as this:



This Documentation is avalaible for all O.S. described above, if your O.S. doesn't appear in this list, contact us.

In each applicomIO solution version folders, you will find a specific documentation related to the O.S. selected. (if you need to ***download acrobat reader,*** got **here**)

(*): for Linux O.S., the folder's name is "Linux_ToolKit", due to the include of driver source code to be able you to build to build the applicom driver for your Linux Kernel.

1. FEATURES & ARCHITECTURE

1.1 Features

The ***O.S. solution*** allows you to develop and to run your own applications using the applicomIO® API. The ***O.S. solution*** contains the different layers of the product (kernel drivers, API and utilities softwares). All the supplied layers are already built and can be used without recompilation. This documentation doesn't include the description of the applicomIO API. For this, refer to the product documentation under Windows (Reference Manual).

1.2 Architecture chosen

Two different kinds of mechanism are used to communicate with the applicomIO boards.

- The first mechanism is named **"mail access"**. It allows to send some messages to the board and to wait for answers. This mechanism is used by the following features:

    - The applicom® API
    - The initialisation process (*with the player tool)*
    - Some diagnostic tools *(with pcmon, etc)*
    - Few functions of the applicomIO API (free messaging mode functions only)

    The "mail access" mechanism is implemented inside the **"User library"** named **applicomio** (applicomIO product)

- The second is named **"I/O access"**. It allows getting directly the I/O data in the dual port memory of the board. This mechanism is used by the following features:

    - The applicomIO® API

The **"I/O access"** mechanism is implemented inside the **"I/O library"** named **appio**

For applicomIO product, according to your needs, two **"mail access libraries"** are available on this CD-ROM. So, you can use:

- A mono-process solution (in the sub-directory **monoprocess**)
  - This solution allows only one application to use the **"mail access"**. For this solution, you don't need to start and use the two serialization daemons. This solution may be enough for example if you want only to develop an application using the appio library. In this case, the **"mail access"** will be however available for configuration or particular diagnostic but only by one application.

- A multi-process solution (in the sub-directory **multiprocess**)
  - This solution allows several applications to use the **"mail access"** simultaneously. For this, you need to start and use the two serialization daemons.

2. BOARD CONFIGURATION & INITIALISATION

2.1 Initialisation Solutions

No tool is available today for the board configuration under Linux. So, the configuration step is done under windows, from a computer containing a version:

- **applicomIO 2.1** or upper for the applicomIO product.

Several ways to configure the boards are possible regarding on your needs and on your material.

2.2 Initialisation through serial connection ***(except PCI-IBSIO & PCI-DPIO boards)***

- Step 1 (on your O.S.):
  - Insert the board on your machine.
  - Connect the serial channel with the COM port of your NT computer.

- Step 2 (on WinNT):
  - Start the applicom configuration **console**.
  - Choose the remote (serial link) mode in the configuration manager.
  - Configure the board and the network.
  - Download the configuration in the flash.

The Initialisation will be automatically executed from the flash each time the power is switched on.

Note: With this method, you can also use all the configuration and diagnostic tools remotely (network configuration detection, network diagnostic...) exactly as if you are under Windows.

2.3 Initialisation with the recorder/player feature

- Step 1 (on WinNT):
  - Start the applicom configuration console.
  - Choose the simulation mode in the configuration manager and select the initialization recording feature
  - Configure manualy the board and the network
  - Start the Download in flash or **PCINITIO.** command. A file named a**pplicomioflash.ply (applicomio.ply)** will be generated

- Step 2 (on your O.S.):
  - Transfer the generated player file (TCP/IP, disk ...).
  - Play this file in the applicomIO board with the **player** utility software.

The Initialisation will be automatically executed from the flash each time the power is switched on.

Note: With this method, you must use the configuration tool in simulation mode. So, you cannot use the network detection feature during the configuration step.

2.4 Initialisation with dual boot system

- Step 1 (PC started with WinNT):
  - Install the board on the computer.
  - Configure the board and the network.
  - Initialise the applicom or applicomIO board with **PCINIT**.

- Step 2 (PC started with your O.S.):
  - Reset the computer without cutting the power.

The configuration must be played in the board each time the power is switched on.

2.5 Initialisation by downloading in flash under Windows NT

- Step 1 (on WinNT):
  - Install the board.
  - Configure the board and the network.
  - Download the configuration in the flash.

- Step 2 (on your O.S.):
  - Insert and use the flashed board.

The Initialisation will be automatically executed from the flash each time the power is switched on.

2.6 Initialisation through serial connection (except PCI-IBSIO and PCI-DPIO boards)

- Step 1 (on your O.S.):

  Insert the board on your machine. Connect the serial channel with the COM port of your Windows computer.

- Step 2 (on Windows):

  Start the applicomIO® configuration console. Choose the remote (serial link) mode in the configuration manager. Configure the board and the network. Download the configuration in the flash. The initialisation will be automatically executed from the flash each time the power is switched on.

With this method, you can also use all the configuration and diagnostic tools remotely (network configuration detection, network diagnostic...) exactly as if you are under Windows.

.

3. TAG BROWSING LIBRARY

3.1 Development principle

The ***Tag Browsing Library*** offers a set of functions which can be split into five groups:

- The initialisation and termination functions: These functions are used to initialise and stop IO mode. They must be called for each board configured:
  - before calling any other library function, as regards initialisation.
  - at the end of the program, as regards the termination function.

- The browse functions: These functions are used to browse the configuration of card in the computer. This functions can be used to browse:
  - Topics,
  - Modules inside topics,
  - Items inside modules.

- The position functions: These functions are used to know the position of an object when we know is name:
  - Topics
  - Modules,
  - Items.

- ·The configuration functions: these functions are used to find a structure which describe the object:
  - Topics config,
  - Modules config,
  - Items config.

- The path functions: these functions are used to set the path and name of XML file used by the library

3.2 Initialising the tag library => *InitLib*

Prototype in C:
unsigned char InitLib (void);

Syntaxe:
C: InitLib( )

Description:
This function is used to initialise the Tag Library on an ***applicomIO***® interface. It is essential to call this function before any other function in the library (expect app_setConfigPath and app_setConfigFileName must call before). It performs all operations required to run applicomIO® (browsing, etc.).

Value returned:
None

example in language C

```
#include "dvwaibrw.h"
Int main( ){
        InitLib ( );

        /* your application */

        ExitLib ( );
        return 0 ;
}
```

3.3 End of Program => ExitLib

Prototype in C:
void ExitLib (void);

Syntax:
C: ExitLib( )

Description:
This function must be called at the end of the program using ***applicomIO***®.

Value returned:
None

example in language C

```
#include "dvwaibrw.h"
Int main( ){
        InitLib ( );

        /* your application */

        ExitLib ( );
        return 0 ;
}
```

3.4 Browsing of Topics => BrowseTopics

Prototype in C:
unsigned char BrowseTopics (char* pszTopicName, PTOPIC_POS ptopicPos);

Syntax:
C: BrowseTopics (szTopicName, &topicPos);

Parameter	**Type**
szTopicName	String, Name of the current topic
ptopicPos	32 bit integer, Position of the current topic

Value returned:
Different of 0 if it's OK. If this function returns **0**, there is no other topic to browse.

Description:
This function is used to browse topics of a board.

example in language C

```
#include ”dvwaibrw.h”
Int main( ){

        TOPIC_POS topicPos = FIRST_TOPIC;          // To point on the first Topic
        char szTopicName[30];
        InitLib ( );

        while ( BrowseTopics ( szTopicName, &topicPos ) != 0 ){

                // Add code to handle topic

        }

          ExitLib ( );
          return 0 ;
}
```

3.5 Browsing of Modules => BrowseModules

Prototype in C:
unsigned char BrowseModules (TOPIC_POS topicPos, char* pszModuleName, PMODULE_POS pmodulePos);

Syntax:
C: BrowseModules (topicPos, szModuleName, &modulePos);

Parameter	**Type**
topicPos	32 bit integer, Position of the current topic return by BrowsTopics
szModuleName	String, Name of the current module
pmodulePos	32 bit integer, Position of the current module

Value returned:
Different of 0 if it's OK. If this function returns **0**, there is no other module to browse.

Description:
This function is used to browse modules in a topic.

example in language C

```
#include "dvwaibrw.h"
Int main( ){

        TOPIC_POS topicPos = FIRST_TOPIC;     // To point on the first Topic
        MODULE_POS modulePos ;
        char szTopicName[30], szModuleName [30];
        InitLib ( );

        while ( BrowseTopics ( szTopicName, &topicPos ) != 0 ){

                // Add code to handle topic

                MODULE_POS moduePos = FIRST_MODULE;
                while ( BrowseModules (topicPos, szModuleName, &modulePos ) != 0 ){

                        // Add code to handle module

                }

        }

          ExitLib ( );
          return 0 ;
}
```

3.6 Browsing of Items => BrowseItems

Prototype in C:

unsigned char BrowseItems (MODULE_POS pmodulePos, char* pszItemName, PITEM_POS pitemPos);

Syntax:
C: BrowseModules (modulePos, szItemName, &itemPos);

Parameter	**Type**
modulePos	32 bit integer, Position of the current module return by BrowsModules
szItemName	String, Name of the current item
pitemPos	32 bit integer, Position of the current item

Value returned:
Different of 0 if it's OK. If this function returns **0**, there is no other topics to browse.

Description:
This function is used to browse items in a module

example in language C

```
#include "dvwaibrw.h"
Int main( ){
TOPIC_POS topicPos = FIRST_TOPIC;                    // To point on the first Topic
MODULE_POS modulePos ;
char szTopicName[30], szModuleName [30], szItemName [30];;
InitLib ( );

        while ( BrowseTopics ( szTopicName, &topicPos ) != 0 ){
                // Add code to handle topic

MODULE_POS moduePos = FIRST_MODULE;
                while ( BrowseModules (topicPos, szModuleName, &modulePos ) != 0 ){
                        // Add code to handle module

ITEM_POS moduePos = FIRST_ITEM;
                        while ( BrowseModules (topicPos, szModuleName, &modulePos ) != 0 ){
                                // Add code to handle item
                                }
                        }
        }
        ExitLib ( );
        return 0 ;
}
```

3.7 Position of Topics => GetTopicPosition

Prototype in C:
TOPIC_POS GetTopicPosisition (const char* pzTopicName);

Syntax:
C: GetTopicPosisition (szTopicName);

Parameter	Type
szTopicName	String, Name of the search topic

Value returned:
this function returns the position of the named topic.

Description:
This function is used to get the position of a topic. This position can be used in all the function which accept position in parameters

example in language C

```
#include ”dvwaibrw.h”
Int main( ){
        TOPIC_POS topicPos = FIRST_TOPIC;      // To point on the first Topic
        char szTopicName[30];
        unsigned short posT ;

        InitLib ( );

        posT = GetTopicPosisition (szTopicName);
        printf(“Position of Topic = %d\n”,posT);

        ExitLib ( );
        return 0 ;

    }
```

3.8 Position of Modules => GetModulePosition

Prototype in C:
MODULE_POS GetModulePosisition (const char* pzMModuleName);

Syntax:
C: GetModulePosisition (szModuleName);

Parameter	**Type**
szModuleName	String, Name of the search module

Value returned:
this function returns the position of the named topic.

Description:
This function is used to get the position of a module. This position can be used in all the function which accept position in parameters

example in language C

```
#include ”dvwaibrw.h”
Int main( ){
        MODULE_POS modulePos ;
        char szModuleName[30];
        unsigned short posM ;

        InitLib ( );

        MODULE_POS modulePos = FIRST_MODULE;

        posM = GetModulePosisition (szModuleName);
        printf(“Position of Module = %d\n”,posM);

        ExitLib ( );
        return 0 ;

    }
```

3.9 Position of Items => GetItemPosition

Prototype in C:
ITEM_POS GetItemPosisition (const char* pzMItemName);

Syntax:
C: GetItemPosisition (szItemName);

Parameter	Type
szItemName	String, Name of the search item

Value returned:
this function returns the position of the named item.

Description:
This function is used to get the position of an Item. This position can be used in all the function which accept position in parameters

example in language C

```
#include ”dvwaibrw.h”
Int main( ){
        MODULE_POS modulePos ;
        char szModuleName[30];
        unsigned short posI ;

        InitLib ( );

        MODULE_POS itemPos = FIRST_ITEM;

        posI = GetItemPosisition (szItemName);
        printf(“Position of Item = %d\n”,posI);

        ExitLib ( );
        return 0 ;
}
```

3.10 Description of Topics => GetTopicCfg

Prototype in C:
void GetTopicCfg (TOPIC_POS topicPos, PST_TOPIC_CFG pstTopicCfg, unsigned short wSizeOfTopic);

Syntax:
C: GetTopicCfg (topicPos, &stTopicCfg, wSizeOfTopic);

Parameter	Type
topicPos	32 bit integer, Position of the current topic
stTopicCfg	32 bit integer, Pointer on the stTopicCfg structure
wSizeOfTopic	32 bit integer, Size of stTopicCfg buffer

Value returned:
None

Description:
This function return structure which describe the topic. See file dvwaibrw.h for more information about the structures.

example in language C

```
#include dvwaibrw.h
Int main( ){
        TOPIC_POS topicPos = FIRST_TOPIC;                    // To point on the first Topic
        char szTopicName[30];
        unsigned short wSizeOfTopic, posT ;

        InitLib ( );

        while ( BrowseTopics ( szTopicName, &topicPos ) != 0 ){
                posT = GetTopicPosisition (szTopicName);
                printf("Position of Topic = %d\n",posT);

                wSizeOfTopic = sizeof (ST_TOPIC_CFG);
                GetTopicCfg (topicPos, &stTopicCfg, wSizeOfTopic);

                Printf("      Card = %d\n",stTopicCfg.wCard);        //give the nb card
                Printf("Equipement = %d\n",stTopicCfg.wEquip);       //give the nb equipement

                // Add code to handle topic
        }
          ExitLib ( );
          return 0 ;
}
```

3.11 Description of Modules => GetModuleCfg

Prototype in C:
void GetModuleCfg (MODULE_POS modulePos, PST_MODULE_CFG pstModuleCfg, unsigned short wSizeOfModule);

Syntax:
C: GetModuleCfg (modulePos, &stModuleCfg, wSizeOfModule);

Parameter	Type
modulePos	32 bit integer, Position of the current module
stModuleCfg	32 bit integer, Pointer on the stModuleCfg structure
wSizeOfModule	32 bit integer, Size of stModuleCfg buffer

Value returned:
None

Description:
This function return structure which describe the module. See file dvwaibrw.h for more information about the structures.

example in language C

```
 #include dvwaibrw.h
Int main( ){
        MODULE_POS modulePos ;
        char szModuleName[30];
        unsigned wSizeOfModule, short posM ;

        InitLib ( );

        MODULE_POS modulePos = FIRST_MODULE;
        while ( BrowseModules (topicPos, szModuleName, &modulePos ) != 0 ){
        posM = GetModulePosisition (szModuleName);
                printf("Position of Module = %d\n",posM);

                wSizeOfModule = sizeof (ST_MODULE_CFG);
                GetModuleCfg (modulePos, &stModuleCfg, wSizeOfModule);

                Printf("  Input Size = %d\n",stModuleCfg.wInputSize);     //InputSize of Module
                Printf("OutputSize = %d\n",stModuleCfg.wOutputSize);  //OutputSize of Module

                // Add code to handle module
        }
           ExitLib ( );
           return 0 ;
}
```

3.12 Description of Items => GetItemCfg

Prototype in C:
void GetItemCfg (ITEM_POS itemPos, PST_ITEM_CFG pstItemCfg, unsigned short wSizeOfItem);

Syntax:
C: GetItemCfg (itemPos, &stItemCfg, wSizeOfItem);

Parameter	Type
itemPos	32 bit integer, Position of the current item
stItemCfg	32 bit integer, Pointer on the stItemCfg structure
wSizeOfItem	32 bit integer, Size of stItemCfg buffer

Value returned:
None

Description:
This function returns structure which describe the item. See file dvwaibrw.h for more information about the structures.

example in language C

```
#include dvwaibrw.h
Int main( ){
        ITEM_POS itemPos ;
        char szItemName[30];
        unsigned short wSizeOfItem, posI ;

        InitLib ( );

        ITEM_POS itemPos = FIRST_ITEM
        while ( BrowseItems (itemPos, szItemName, &itemPos ) != 0 ){
                posI = GetItemPosisition (szItemName);
                printf("Position of Item = %d\n",posI);

                wSizeOfItem = sizeof (ST_ITEM_CFG);
                GetItemCfg (itemPos, &stItemCfg, wSizeOfItem);

                Printf("Offset in Equipement = %d\n",stItemCfg.wOffsetInEquip);
                Printf("Offset in module = %d\n",stItemCfg.wOffsetInModule;

                // Add code to handle module
        }
          ExitLib ( );
          return 0 ;
}
```

3.13 Set XML file Path => app_setConfigPath

Prototype in C:
int app_setConfigPath (char* newFilePath);

Syntax:
C: app_setConfigPath (newFilePath);

Parameter	Type
newFilePath	String, representing the path of the XML file

Value returned:
This function return 0 if some error occurs.

Description:
This function allows to set the path of XML file used by the library. It takes as parameter a NULL terminated string representing the path of the XML file. If the user doesn't specified any path the library looks for the XML file in the current folder.

The function must be called before calling the InitLib procedure; if it is called newly before calling the ExitLib procedure these functions have not effect.

Do not add a "/" at the end of the string because it will be automatically added.

example in language C

```
#include dvwaibrw.h
Int main( ){

        app_setConfigPath ("/home/root/applicom/XML");

        app_setConfigFileName("MyconfigTag.xml");


        InitLib ( );

                // Add your code
        }

          ExitLib ( );
          return 0 ;
}
```

3.14 Set XML file name => app_setConfigFileName

Prototype in C:
int app_setConfigFileName (char* newFileName);

Syntax:
C: app_setConfigFileName (newFileName);

Parameter	Type
newFileName	String, representing the name of the XML file

Value returned:
This function return 0 if some error occurs.

Description:
This function allows to set the name of XML file used by the library. It takes as parameter a NULL terminated string representing the name of the XML file. If the user doesn't specified any name the library looks for the "ConfigTag.xml" file.
The function must be called before calling the InitLib procedure; if it is called newly before calling the ExitLib procedure these functions have not effect.

example in language C

```
#include dvwaibrw.h
Int main( ){

        app_setConfigPath (“/home/root/applicom/XML”);

        app_setConfigFileName(“MyconfigTag.xml”);


        InitLib ( );

                // Add your code
        }

          ExitLib ( );
          return 0 ;
}
```

4. Versioning function

The developer can use these function call to retrieve version information about library, card bios, date and time of card configuration and so on. The function has the following signature:

```
int AuGetNumInformation (unsigned short wCodeInfo,
void *pvGlobalInfo,
unsigned long *pdwSize,
unsigned long *pdwStatus);
```

Parameters:

Parameter	Type
wCodeInfo	**Used to specify the requested information**
pvGlobalInfo	**Used to indicate parameters according to the wCodeInfo requested.**
pdwSize	**Used to specify in input the size of the pvGlobalInfo buffer and in return to find the size of the returned information.**
pdwStatus	**function return status**

Value returned :

TRUE if OK. If this function returns **FALSE**, the status variable contains one of the following error values.

Status = 0 Indicates that the function was executed correctly
Status = 32 Indicates a bad parameter during the function call

Possible values for wCodeInfo

4.1 AU_SOFT_VERSION (0x01)

Returns the applicom library version.
pvGlobalInfo format: Unsigned 32-bit integer

32	28	24	20	16	12	8	4	0
	Reserved	Reserved	Reserved	Reserved	Major Version	Minor Version	Reserved	Service pack

Notes:
If the value of ***pvGlobalInfo** is 0, this value cannot be used.

Example of use:

```
unsigned long LibraryVersion;
unsigned long gdwStatus;
unsigned long gdwSize;
short MajorVersion,MinorVersion,spVersion;
gdwSize = sizeof(unsigned long);

if(AuGetNumInformation(0x01,&LibraryVersion,&gdwSize,&gdwStatus)){

if (0 != LibraryVersion ){
  MajorVersion = (short) LibraryVersion >>12 & 0xF;
  MinorVersion = (short) LibraryVersion >>8 & 0xF;

  spVersion = (short) LibraryVersion & 0xF;

  printf("Version : %d.%d sp %d", MajorVersion, MinorVersion, spVersion);
}
else{
  printf("  Status= %d ", gdwStatus);
}
```

4.2 AU_EMBEDDED_SOFT_VERSION (0x02)

Returns the version of the applicom library which was used to initialize the applicom interfaces for each interface present.
pvGlobalInfo format: Table of 8 unsigned 32-bit integers

index	32	28	24	20	16	12	8	4	0
0	Board 1	Reserved	Reserved	Reserved	Reserved	Major Version	Minor Version	Reserved	Service pack
N	Board n					...	...	...	...
7	Board 8	Reserved	Reserved	Reserved	Reserved	Major Version	Minor Version	Reserved	Service pack

Notes:
If the value of **pvGlobalInfo[n]** is 0 or the applicom interface "**n**" has not been initialized or it is not present.

Example of use:

```
unsigned long EmbeddedVersion[8];
unsigned long gdwStatus;
unsigned long gdwSize;
short MajorVersion,MinorVersion,spVersion;

gdwSize = sizeof(EmbeddedVersion);
if(AuGetNumInformation(0x02,EmbeddedVersion,&gdwSize,&gdwStatus)){
  for(int i=0;i<8;i++){
    if(0 != EmbeddedVersion[i] ){
      MajorVersion = (short) (EmbeddedVersion[i]>>12 & 0xF);
      MinorVersion = (short) (EmbeddedVersion[i]>>8 & 0xF);
      spVersion = (short) (EmbeddedVersion[i] & 0xF);
      printf("Board %d : Version %d.%d sp %d", i+1 ,MajorVersion, MinorVersion, spVersion);
      }
    }
  }
}
else{
  printf(" Status= %d ", gdwStatus);
}
```

4.3 AU_BIOS_VERSION (0x03)

Returns the bios version of each applicom interface present in the computer.
pvGlobalInfo format: Table of 8 unsigned 32-bit integers

index	32	28	24	20	16	12	8	4	0
0	Board 1	Reserved	Reserved	Major Version		Minor Versio		Revision	
N	Board n			...		...		...	
7	Board 8	Reserved	Reserved	Major Version		Minor Version		Revision	

Notes:
If the value of **pvGlobalInfo[n]** is 0 the applicom interface "**n**" is not present.

Example of use:

```
unsigned long BiosVersion[8];
unsigned long gdwStatus;
unsigned long gdwSize;
short MajorVersion,MinorVersion,Revision;

gdwSize = sizeof(BiosVersion);

if (AuGetNumInformation(0x03, BiosVersion, &gdwSize, &gdwStatus)){
  for(int i=0; i<8; i++){
    if(0 != BiosVersion [i] ){
      MajorVersion = (short) (BiosVersion [i]>>16 & 0xFF);

      MinorVersion = (short) (BiosVersion [i]>>8 & 0xFF);
      Revision = (short) (BiosVersion [i] & 0xFF);
      printf("Board %d : Bios Version %d.%d sp %d",i+1 ,MajorVersion, MinorVersion,
Revision);
    }
  }
}

else {
  printf(" Status = %d ", gdwStatus);
}
```

4.4 AU_TYPE_CARD (0x04)

Returns the type and the revision for each applicom interface present in the computer.
pvGlobalInfo format: Table of 8 unsigned 32-bit integers

index	32	28	24	20	16	12	8	4	0
0	Board 1	Reserved	Reserved	Interface type				Revision	
N	Board n			...				...	
7	Board 8	Reserved	Reserved	Interface type				Revision	

Notes:
If the value of **pvGlobalInfo[n]** is 0 the applicom interface "**n**" is not present.

Example of use:

```
unsigned long InterfaceType[8];
unsigned long gdwStatus;
unsigned long gdwSize;
short wInterfaceType;
char InterfaceRevision;

gdwSize = sizeof(InterfaceType);

if (AuGetNumInformation(0x04, InterfaceType, &gdwSize, &gdwStatus)){
  for(int i=0;i<8;i++){
    if(0 != InterfaceType [i] ){
      wInterfaceType = (short ) (InterfaceType[i] >>8 & 0xFFFF);

      InterfaceRevision = (char )(InterfaceType[i] & 0xFF);

      printf("Board %d : Type = %d Rev %c", i+1, wInterfaceType, InterfaceRevision);
    }
  }
}
else {
  printf(" Status = %d ", gdwStatus);
}
```

4.5 AU_EMBEDDED_DATE (0x05)

Returns the date when the configuration was made for each applicom interface present in the computer.
pvGlobalInfo format: Table of 8 unsigned 32-bit integers

index	32	28	24	20	16	12	8	4	0
0	Board 1	Day		Month		Year			
N	Board n					...			
7	Board 8	Day		Month		Year			

Notes:
If the value of **pvGlobalInfo[n]** is 0 or the applicom interface "**n**" has not been initialized or it is not present.

Example of use:

```
unsigned long EmbeddedDate[8];
unsigned long gdwStatus;
unsigned long gdwSize;
short Day,Month,Year;

gdwSize = sizeof(EmbeddedDate);

if (AuGetNumInformation(0x05, EmbeddedDate,&gdwSize,&gdwStatus)){
  for(int i=0; i<8; i++){
    if(0 != EmbeddedDate[i] ){
      Day = (EmbeddedDate [i] >> 24) & 0xFF;
      Month = (EmbeddedDate [i] >> 16) & 0xFF;
      Year = (EmbeddedDate [i] ) & 0xFFFF;

      printf("Board %d : %d/%d/%d ",i+1,Day, Month, Year);
    }
  }
}else {
  printf(" Status = %d ", gdwStatus);
}
```

4.6 AU_EMBEDDED_TIME (0x06)

Returns the time when the configuration was made for each applicom interface present in the computer.
pvGlobalInfo format: Table of 8 unsigned 32-bit integers

index	32	28	24	20	16	12	8	4	0
0	Board 1	Reserved	Reserved	seconds		minutes		hours	
N	Board n					<P< p>			
7	Board 8	Reserved	Reserved						

Notes:
If the value of **pvGlobalInfo[n]** is 0 or the applicom interface "**n** has not been initialized or it is not present.

Example of use:

```
unsigned long
EmbeddedTime[8];
unsigned long gdwStatus;
unsigned long gdwSize;
short Hour,minute,second;

gdwSize = sizeof(EmbeddedTime);

if (AuGetNumInformation(0x06, EmbeddedTime,&gdwSize,&gdwStatus)){
  for(int i=0;i<8;i++){
    if(0 != EmbeddedTime [i] ){
      second = (EmbeddedTime [i] >> 16) & 0xFF;
      minute = (EmbeddedTime [i] >> 8) & 0xFF;
      hour = (EmbeddedTime [i] ) & 0xFF;
      printf("Board %d : %d:%d:%d " ,i+1, hour, minute, second);
    }
  }
}
else{
  printf(" Status = %d ", gdwStatus);
}
```

4.7 AU_PLY_VERSION (0x07)

Returns the version of the applicom library which was used to generate the ".ply" file specified.
pvGlobalInfo format:

> *In input:* string array containing the path and name of the ".ply" file for which you want to obtain the version of the applicom library used to generate it.
>
> *In output:* unsigned 32-bit integers

32	28	24	20	16	12	8	4	0
	Reserved	Reserved	Reserved	Reserved	Major Version	Minor Version	Reserved	Service pack

Notes:
If the value of ***pvGlobalInfo** is 0, this value cannot be used.

Example of use:

```
char szBuffer[255];
unsigned long gdwStatus;
unsigned long gdwSize;
short MajorVersion,MinorVersion,spVersion;

gdwSize = sizeof(szBuffer);

/* The path and name of the ply are indicated */
szBuffer = "c:\\applicom.ply";
 bgcolor=gray
if (AuGetNumInformation(0x01, szBuffer,&gdwSize,&gdwStatus)){
  if (0 != LibraryVersion ){
    MajorVersion = (short) *((unsigned long * )szBuffer) >>12 ;
    MinorVersion = (short) *((unsigned long * )szBuffer) >>8 & 0xF;
    spVersion = (short) *((unsigned long * )szBuffer) & 0xF;
    printf("Ply Version : %d.%d sp %d", MajorVersion, MinorVersion, spVersion);
  }
}
else{
  printf(« Status = %d », gdwStatus);
}
```

4.8 AU_PLY_DATETIME (0x08)

Returns the date and time when the configuration contained in the specified ".ply" file was made.
pvGlobalInfo format:

> *In input:* string array containing the path and name of the ".ply" file for which you want to obtain the version of the applicom library used to generate it.
>
> *In output:* Table of 2 unsigned 32-bit integers

index	32	28	24	20	16	12	8	4	0
0	Date	Day		Month		Year			
1	Time			Seconds		Minutes		Hours	

Notes:
If the value of **pvGlobalInfo[n]** is 0, this value cannot be used.

Example of use:

```
char szBuffer[255];
unsigned long gdwStatus;
unsigned long gdwSize;
short Day, Month, Year;
unsigned long dwPlyDate;
short second, minute, hour;
unsigned long dwPlyTime;
gdwSize = sizeof(szBuffer);

/* The path and name of the ply are indicated */
szBuffer = “c:\\applicom.ply”;
if (AuGetNumInformation(0x01, szBuffer, &gdwSize, &gdwStatus)){
  if (0 != LibraryVersion ){
    dwPlyDate = (unsigned long) *((unsigned long * )szBuffer);
    dwPlyTime = (unsigned long) *((unsigned long * )&szBuffer[4]);
    second = (dwPlyTime>>16) & 0xFF;
    minute = (dwPlyTime>>8) & 0xFF;
    hour = (dwPlyTime) & 0xFF;
    printf("ply Time = %d :%d : %d \n" hours,minutes, seconds);
    Day = (dwPlyDate>>24) & 0xFF;
    Month = (dwPlyDate>>16) & 0xFF;
    Year = (dwPlyDate) & 0xFFFF;
    printf("Ply date : %d/%d/%d", Day, Month, Year);

  }
}
else{
  printf(" Status = %d ", gdwStatus);
}
```

4.9 Interface type according to interface ID

Interface name	ID	Interface name	ID
PC1000	**1**	**PCI1000**	**>21**
PC4000	**2**	**PCI2000FIP**	**22**
VME400	**3**	**PCI1500S7**	**23**
GT4000	4	**PCI1500PFB**	**24**
IBM_RIC	**5**	**PC104PFB**	**25**
PC1000PFB_CP5480	**6**	**PCI_DPIO**	**26**
PC2000ETH	**7**	**PCI_IBSIO**	**27**
PC1500PFB	**8**	**PC104_DPIO**	**28**
PC2000FIP	**9**	**PCI2000**	**29**
PC2000	**10**	**PCI2000MBP**	**30**
GT4010	**11**	**PCI_ETHIO**	**31**
GT2000PFB	**12**	**PCI_DVNIO**	**32**
GT2000	**13**	**PCI_CANIO**	**33**
PCI2000ETH	**14**	**PC104_DVNIO**	**34**
PCI2000PFB	**15**	**PC104_CANIO**	**35**
PCI2000IBS	**16**	**CPCI1000PFB**	**36**
PC1500S7	**17**	**CPCI_DPIO**	**37**
PCI2000CAN>	**18**	**SW1000ETH**	**38**
PCI2000ASI	**19**	**CPCI_DVNIO**	**39**
PCI4000	**20**	**CPCI_CANIO**	**40**