IP Windows Driver Software
User's Manual

ACROMAG INCORPORATED
30765 South Wixom Road
P.O. BOX 437
Wixom, MI 48393-7037 U.S.A.

Tel: (248) 295-0310
Fax: (248) 624-9234

9500-330H

Contents

Contents 3
Introduction 4
- **Hardware Support** 5
- **Supported Programming Languages** 6

Getting Started 7
- **Software Installation** 7
  - **Installed Software** 7
    - **Program Menu Shortcuts** 7
    - **Installed Files** 7
- **Hardware Installation** 8
- **IP Enumeration Utility** 9
- **Demonstration Programs** 9
- **Where to Go From Here** 9

Software Overview 10
- **Function Format** 10
  - **Status Codes** 11
- **Sequence of Operations** 13
- **Interrupts** 14
  - **Callback Functions** 14
  - **Synchronization** 15
  - **Base Address Pointers** 15

Using the Library with Visual C++ 16
- **32-bit application** 16
- **64-bit application** 17

Deploying Applications to Target Systems 18
- **Kernel Driver Installation** 18
- **Application and DLL Installation** 18
- **DLL Location Notes** 18
  - **Modifying the PATH setting** 19
    - **Windows XP, Vista, 7 and 8** 19

Introduction

IP Windows Driver Software consists of kernel drivers and Windows Dynamic Link Libraries (DLLs) that facilitate the development of Windows applications interfacing with Industry Pack modules installed on Acromag PCI carrier cards (e.g. APC8620A), PCI Express carrier cards (e.g. APCe8650) and CompactPCI Carrier Cards (e.g. AcPC8625). The software provides custom support for all Acromag Industry Pack modules (e.g. IP220) as well as general read/write access and interrupt support for IP modules from other vendors.

Notes:

- **Most carrier control software functions are used identically with all Acromag PCI, PCI Express and CompactPCI Carrier Cards. The convention of this document is to refer to all these cards using the generic term "carrier."**
- **Although the software provides support for IP modules from other vendors, it <u>does not</u> support carrier cards from other vendors.**

The software includes kernel drivers and DLLs for both 32 and 64-bit versions of Windows. Applications developed to run on 32-bit versions of Windows use the 32-bit kernel drivers and 32-bit DLLs. Applications developed to run on 64-bit versions of Windows use the 64-bit kernel drivers and either the 32-bit or 64-bit DLLs. Except where noted otherwise, the usage of the 32-bit and 64-bit DLLs is identical.

Notes:

- **64-bit versions of Windows support running both 32-bit and 64-bit applications. 32-bit applications run within the WoW64 subsystem. A 32-bit application must use the 32-bit DLLs and a 64-bit application must use the 64-bit DLLs. The 64-bit kernel driver is always used on 64-bit versions of Windows.**

- **The "bitness" of the Target system is important. The bitness of the Development system is not. For example, you can install IPSW-API-WIN on a 32-bit Windows system and use it to develop a 64-bit application that will run on a 64-bit Windows target.**

DLL functions use the Windows _stdcall calling convention and can be accessed from a number of programming languages.

This document covers general information on software installation, programming concepts, application development and redistribution issues. The following documents are also included in the IP Windows Driver Software documentation set:

- Function Reference document for the IP Carrier DLL
- Function Reference documents for each Acromag IP module DLL.
- Function Reference document for the "IGeneric" DLL (which can be used to access IPs from other vendors).

After reviewing this user's manual, readers will next want to consult the Function Reference documents specific to their hardware.

The older, 32-bit only versions of **IPSW-API-WIN** were based on a third party, 32-bit only kernel driver. To support 64-bit versions of Windows, new kernel drivers were developed and all the DLLs were updated. For a time **IPSW-API-WIN** was split into separate 32-bit and 64-bit products (**IPSW-API-WIN32** and **IPSW-API-WIN64**). If you are updating from -**WIN32**, **-WIN64**, or an earlier version of **IPSW-API-WIN**, please review the **Migrating from previous software**

versions document. This document describes the differences between the various versions of the software, and outlines how to upgrade applications written against the old libraries.

Hardware Support

The list of supported Acromag Industry Pack carriers and modules is shown in Table1.

Table 1: Acromag IP Carriers and Modules

Model	Description	Interrupts
AcPC8625	4 slot non-intelligent CompactPCI bus carrier board	N.A.
AcPC8630	2 slot non-intelligent CompactPCI bus carrier board	N.A.
AcPC8635	2 slot, rear I/O, non-intelligent CompactPCI bus carrier board	N.A.
APC8620	5 slot non-intelligent PCI bus carrier board	N.A.
APC8621	3 slot non-intelligent PCI bus carrier board	N.A.
APC8620A	5 slot, 32MHz capable non-intelligent PCI bus carrier board	N.A.
APC8621A	3 slot, 32MHz capable non-intelligent PCI bus carrier board	N.A.
APC8640	5 slot, 32MHz capable non-intelligent PCI bus carrier board	N.A.
APCe8650	4 slot, 32MHz capable non-intelligent PCI Express bus carrier board	N.A.
IP-1K100	Reconfigurable FPGA module via IP bus	Yes
IP-1K110	Reconfigurable FPGA module via IP bus	Yes
IP-1K125	Reconfigurable FPGA module via JTAG	Yes
IP-EP2	Reconfigurable FPGA module via JTAG or IP bus	Yes
IP220(A)-X	8/16 Non-Isolated 12-Bit DAC Outputs	No
IP230-X	4/8 Ch., 16-Bit DAC Outputs (No RAM or interrupt)	No
IP231-X	8/16 Non-Isolated 16-Bit DAC Outputs	No
IP235-X	4/8 Ch., 16-Bit DAC Outputs with RAM and interrupt	Yes
IP236-X	4/8 Ch., 16-Bit DAC Outputs with FIFO and interrupt	Yes
IP320(A)	40SE/20DE Non-Iso. 12-Bit ADC Inputs	No
IP330	16 Bit (16DE/32SE) Analog Input Module	Yes
IP340/1	12/14 Bit (16DE) Simultaneous Analog Input Module	Yes
IP400	40 Non-Iso. Digital Inputs	Yes
IP405	40 Non-Iso. Low Side Digital Output Switches	No
IP408	32 Non-Iso. Digital Inputs/Outputs (Low Side Sw.)	Yes
IP409	24 Non-Iso. Digital Inputs/Outputs (Differential)	Yes
IP440-X	32 Ch., Isolated Digital Input Module with Interrupts	Yes
IP445	32 Ch., Isolated SSR Output Module	No
IP470	48 Ch., Digital I/O Module with Interrupts	Yes
IP480-X	2/6, 16-Bit Counter/Timer Mod. with Int. & Watchdog	Yes
IP482	10 16-Bit Counters - TTL	Yes
IP483	5 16-Bit Counters – TTL, 2 16-Bit Counters – RS422	Yes
IP484	5 16-Bit Counters –RS422	Yes
IP500	4 Port, Serial 232 Communication	Yes
IP501	4 Port, Serial 422/485 Communication	Yes
IP502	4 Port, Serial 485 Communication	Yes
IP511	4 Port, Isolated Serial 422 Communication	Yes
IP512	4 Port, Isolated Serial 485 Communication	Yes
IP520	8 Port, Serial 232 Communication	Yes
IP521	8 Port, Serial 422/485 Communication	Yes
IP560	Two Controller Area Network (CAN) 2.0B channels	Yes

N.A. = Not Applicable

Supported Programming Languages

The demonstration program source code as well as the function descriptions and example code provided in the documentation, are all written in the C programming language. However, the library functions are also callable from numerous other languages. The DLL functions use the Windows _stdcall calling convention. Any programming language capable of calling the Windows API DLL functions should be able to call the IPSW-API DLL functions in a similar manner.

Getting Started

Software Installation

To install the IP Windows Driver software on the development system, insert the software disk into the CD drive and run **Setup.exe**. Administrative rights are required to perform the installation.

Note:
The installation CD is used to install the complete driver software package on the development system (the system where your application will be written and built). Often, the target system (the system where your application will run) will be different than the development system. If your development and target systems are different, you will also want to read the **Deploying Applications to Target Systems** section of this document.

INSTALLED SOFTWARE

Program Menu Shortcuts

- Shortcut to installation directory (see below)
- Shortcut to this manual
- Shortcut to Enumeration utility (optional)
- Shortcut to executable demonstration programs (optional)

Installed Files

The default installation directory is `C:\Acromag\IPSW_API_WIN`.

Subdirectory	
c_examples	Microsoft Visual C++ example projects with source code. **Prebuilt 32-bit and 64-bit executables are in each project's Release directories**
c_include	Header files
c_lib	32-bit and 64-bit COFF format import libraries (segregated into Win32 and Win64 subdirectories)
config_files	Example VHDL object code for reconfigurable I-Packs
docs	User's manual, DLL references, revision history, application notes
redist\DLLs	32-bit and 64-bit DLLs (segregated into Win32 and Win64 subdirectories)
redist\Driver	INF and catalog files for each board
redist\Driver\i386	32-bit device driver and co-installer DLL
redist\Driver\amd64	64-bit device driver and co-installer DLL
utility	32-bit and 64-bit IP module enumeration utilities (segregated into Win32 and Win64 subdirectories)

Hardware Installation

1. Plug the necessary I-Packs into the carrier. Make sure to configure any jumpers as necessary.
2. With power off, install the carrier into an available slot on the PC. Connect any field wiring at this time.
3. Turn on the system.

   XP and Vista:
   You will receive a dialog box shortly after boot-up asking if you want to install a driver for the new device. Answer yes and direct the New Hardware Wizard to the "redist\Driver" subdirectory (see table above) or to the optical drive containing the IP Windows Driver disk or to some other directory with the required files (see note below). The New Hardware Wizard will copy and install the kernel mode driver.

   Windows 7 and 8:
   a. Launch Device Manager from the Control Panel (or type devmgmt.msc in the Search Box)
   b. Locate and right-click the "Other PCI Bridge Device," select Update Driver, and browse to the driver files as described for XP and Vista.

Note
If you want to install the driver from a custom location (e.g. a flash drive) the following files must be present.

- The INF file for the carrier (e.g. Apc8620.inf). The same INF file is used to install both the 32-bit and the 64-bit drivers.
- The platform specific catalog file for the board (e.g. Apc8620_x64.cat). The catalog file is digitally signed by Acromag. It should be placed in the same directory as the INF file.
- The platform specific kernel driver. The kernel driver must be placed in a subdirectory relative to the INF and catalog files. The 32-bit driver must be in a subdirectory named "i386" and the 64-bit driver must be in subdirectory named "amd64."

IP Enumeration Utility

IP Windows Driver Software includes a command line utility, **IEnum.exe,** that may be run to display basic information about all installed Acromag IP carriers. Running this utility is a quick way to verify that the device driver and boards are properly installed. The displayed information includes the carrier number, carrier type (PCI or PCI Express), user assigned carrier ID (not supported on all carriers), the address and length of each memory range present,.and the names of all installed IP modules. In addition, the utility indicates if each carrier supports 32MHz operation. The kernel driver and carrier DLL (APC86xx.dll) must be installed to use this utility. Note that the carrier number is the value passed to the `A86_Open` function to open a connection to the carrier. (See the **Sequence of Operations** section below.)

Demonstration Programs

Console demonstration programs (source code provided) are included for each Acromag IP module. Use the demo program to test your board and to become familiar with how it operates.

Tip:
In many cases the main demo source file (IxxxDemo.c) will include comments with information on how to run the demo (sequences of steps, channel wiring information, etc.).

Where to Go From Here

1. Read the remainder of this document. It covers general concepts for using the software.

2. Read the function reference document for the library you will be developing with.

3. Study the source code for the demonstration program corresponding to your module. You can use this code as a starting point for your custom application.

Software Overview

The software includes a 32-bit and a 64-bit DLL for carrier access and 32-bit and 64-bit DLLs for each Acromag IP module. In most cases the name of the DLL corresponds to the name of the IP module. There are a few exceptions, however, where groups of similar IP modules are supported by a single DLL. These include:

IP Modules	Shared DLL(s)
IP1K100, IP1K110, IP1K125 and IP-EP2	I1K100.dll
IP340 and IP341	I340.dll
IP482, IP483 and IP484	I482.dll
IP502 and IP512	I502.dll

The DLLs provide the Application Programming Interface (API) used to access the hardware. Each DLL is written in C and contains functions using the _stdcall calling convention. A DLL is loaded and linked at runtime when its functions are called by an executable application. Multiple applications can access the functions of a single copy of a DLL in memory.

Tip:
The 32-bit and 64-bit DLLs share the same names and are installed into separate Win32 and Win64 directories on the development system. If you ever have a "loose" DLL and need to determine its type, right-click the file and view its Properties. The Version Description indicates whether the DLL is 32 or 64 bit.

Function Format

All IP carrier DLL functions have the following form:

```
status = A86_FunctionName(arg1, arg2, ... argn)
```

The format of IP module DLL functions is similar:

```
status = IXXX_FunctionName(arg1, arg2, ... argn)
```

The "IXXX" portion of the function name indicates the IP module the function is used with (e.g. I470).

Every function returns a 32-bit status value. This value is set to 0 when a function completes successfully or to a negative error code if a problem occurred. The following **Status Codes** section describes the values that may be returned from the DLL functions.

For most functions, *arg1* is an integer "handle" used to reference a specific carrier or IP module. (See the **Sequence of Operations** section below.)

STATUS CODES

The table below summarizes the 32-bit status codes that may be returned from the DLL. Please note the return code of any failing functions if you should need to contact Acromag for technical support.

Value	Mnemonic	Description
0	E_OK	Operation Successful
-1	E_INVALID_MODHNDL	Returned if no IP module is associated with the specified handle. Applies to most IP DLL functions.
-2	E_BOARD_IN_USE	Returned by *A86_Open* if the carrier is already open. This can occur if the carrier is in use by another application.
-4	E_CONNECT	Returned by *A86_Open* if an error occurred connecting to the carrier. This will occur if the specified card number is invalid or if the kernel mode drivers are not properly installed or configured.
-5	E_MAPMEM	Returned by *A86_Open* or if an error occurred mapping the devices physical memory into the virtual address space.
-8	E_OUTOFHANDLES	Returned by *A86_Open* or *IXXX_Open* if the maximum number of handles have already been allocated.
-9	E_BAD_PARAM	Returned by a function if it received an invalid parameter. This typically means the parameter was outside of the expected range or the function received a NULL pointer. Consult the individual function descriptions for other possible reasons for this error.
-10	E_INSUF_RESOURCES	Returned by a function if there were insufficient resources to create a required data structure or carry out an operation.
-13	E_CONFIG_READ	Returned by *A86_ReadConfigReg* if an error occurred while reading data from the device's PCI configuration space.
-14	E_TIMEOUT	Returned by a function if it timed out before completing.
-15	E_CONFIG_SET	Returned by a Configuration function if the current settings used by this function do not represent a valid configuration
-16	E_CALIB	Indicates an error generating or using calibration data.
-17	E_BUFFER	Indicates an error occurred accessing a user defined data buffer
-21	E_EEPROM_READBACK	The value read from the EEPROM doesn't match the value written
-22	E_FILE_OPEN	Returned if a file cannot be opened
-23	E_FILE_FORMAT	Returned if file contents are not in the expected format
-24	E_FILE_READ	Returned if a file cannot be read
-25	E_CONFIG_DONE	Returned by functions if the configuration of a re-configurable board is not complete
-26	E_EX_DESIGN	Some DLL functions for reconfigurable Acromag IP modules will return this error if the IP is not configured with Acromag example design.
-27	E_FLASH_BUSY	Returned if a flash memory operation failed because the flash chip is busy

-29	E_UNSUPPORTED	Returned if the hardware does not support the function. For example not all carriers support 32MHz operation.
-30	E_CHECKSUM	Returned if a checksum mismatch is detected
-31	E_HANDLER	The function requires that an interrupt handler be attached
-32	E_READY	The device is not ready to execute this command
-33	E_EXT_LIBRARY	An external library returned an error
-34	E_RESOURCE_IN_USE	The specified resource is already in use
-35	ERR_INVALID_ CTRLHNDL	Returned if the kernel driver control handle is invalid
-40	E_INVALID_CRHNDL	Returned if no IP carrier is associated with the specified handle. Applies to most IP carrier DLL functions
-41	E_EMPTY_SLOT	Returned by carrier and IP module functions if the specified carrier slot does not contain an I-Pack. Note that this error is also returned if a specified slot letter falls in the range C – E but exceeds the number of slots found on the model of carrier being accessed.
-42	E_SLOT_IN_USE	Returned by *IXXX_Open* if the IP in the specified carrier slot is already open
-43	E_MODULE_MODEL	Returned by *IXXX_Open* if the IP in the specified carrier slot is not a model supported by this DLL
-44	E_HANDSHAKE	Returned by serial communication functions if an expected handshake signal was not received

Sequence of Operations

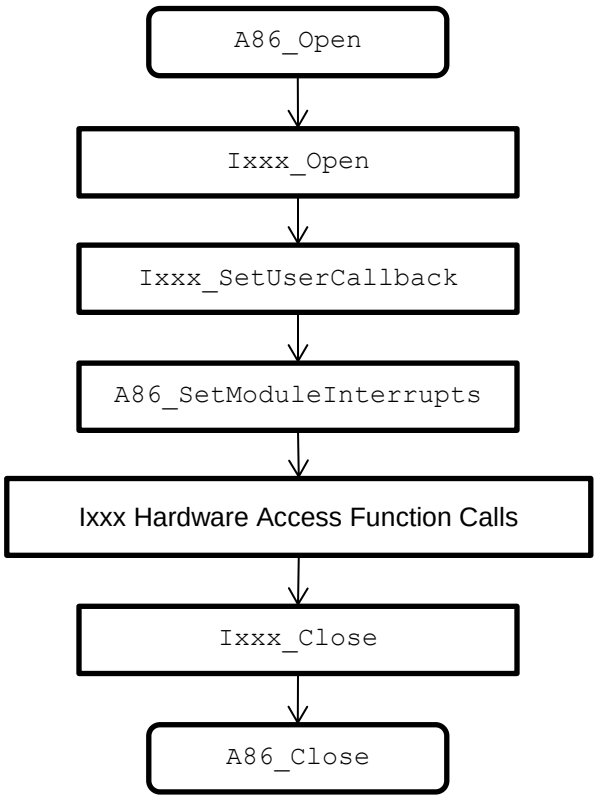
Although each IP module has its own DLL with unique functions, all IP modules are accessed in the following manner:

1. Open a connection to the carrier where the IP module is installed by calling A86_Open. This function provides an integer "carrier handle" that is used in all subsequent carrier function calls.
2. Open connections to one or more IP modules on the carrier by calling the corresponding IXXX_Open functions. IXXX_Open receives the carrier handle and carrier slot letter and provides an IP module handle that is used in subsequent function calls to that module.
3. (Optional*) Setup a callback function for each IP that supports interrupts.
4. (Optional) Enable the generation of IP module interrupts by calling A86_SetModuleInterrupts.
5. Implement your application logic using the carrier and IP module hardware access function calls.

Prior to terminating the application:

6. Close each IP connection by calling IXXX_Close.
7. Close the carrier connection by calling A86_Close.

These steps are summarized in the diagram below



* A callback function or event handler is required when using an IP560 with interrupts.

Interrupts

IP Windows Driver Software supports callback functions for allowing your application to respond to interrupts generated by an IP module. Callback functions execute synchronously with the internal interrupt service routines (see below).

Each IP DLL that supports interrupts has its own predefined internal interrupt service routine. The specifics of each routine are outlined in the IP module's corresponding Function Reference document. If you choose to implement a callback function, you have the option of overriding this routine. This is done by setting a "*Replace*" parameter when designating the callback. (See **Callback Functions** below.)

When an interrupt occurs the following sequence of events takes place:

1. The kernel level driver disables the carrier's IP Module Interrupt Enable bit and signals the carrier DLL's internal interrupt service routine (ISR).
2. The carrier ISR identifies the IP with pending interrupts and calls the device specific ISR in the corresponding IXXX DLL.
3. At this point three things can happen
   - If no callback was configured, the IXXX ISR simply processes the interrupt and returns a TRUE value to the carrier ISR.
   - If a callback function was configured but should not override the internal ISR, the internal IXXX ISR processes the interrupt, invokes the callback and returns a TRUE value to the carrier ISR.
   - If a callback function was configured to override the internal IXXX ISR, the ISR invokes the callback rather than processing the interrupt and then returns a TRUE value to the carrier ISR. It is the responsibility of the callback function to process the IP interrupt.
4. The carrier ISR identifies pending IP interrupts in slot order (A – E). Steps 2 and 3 are repeated for each interrupting IP module. If all IXXX ISRs have returned TRUE, the carrier ISR re-enables the IP Module Interrupt Enable bit.

CALLBACK FUNCTIONS

When using the callback mechanism your application defines a function that the IP DLL will call from its internal interrupt service routine. The format of this function must exactly match that expected by the DLL. This format is hardware specific and is given in the **IXXX_SetUserCallback** topic in the IP module's Function Reference document.

This format, however, will be some variation of the following:

```
C:   void _stdcall YourIsrName (int Handle, BYTE Status)
```

The *Handle* argument identifies the IP module that caused the interrupt. If the function is not overriding the internal ISR, the Status variable(s) will contain data allowing you to determine the cause of the interrupt (e.g. the value read from a status register by the internal ISR). If the function is overriding the internal ISR, the Status variable(s) will be zero since the internal ISR did not read any registers prior to invoking the callback function.

SYNCHRONIZATION

The DLL's interrupt service routine (ISR) executes on a different thread than that of your application. Within the DLL the ISR (which includes the call to any callback function) is delimited as a device critical section. *IXXX_StartIsrSynch* and *IXXX_EndIsrSynch* can be used to synchronize other application threads with the ISR thread. DLLs for Acromag IP modules that do support interrupts also provide these functions to support the synchronization of hardware access by multiple threads within an application.

Bracketing a section of code between calls of *IXXX_StartIsrSynch* and *IXXX_EndIsrSynch* defines that code as a device critical section. Two threads within a single process cannot execute critical section code simultaneously. *IXXX_StartIsrSynch* should be called by your application before it attempts to access data or device memory that can be accessed by another thread. Remember to call *IXXX_EndIsrSynch* when finished accessing these shared resources.

BASE ADDRESS POINTERS

Each IP DLL provides a function that returns the base address of the user mode mapping of the IP module's I/O space.

C and C++ programmers can cast the returned value to a byte pointer and access memory using normal pointer mechanisms. This method can be used to write additional functions that complement those provided through the DLL.

Example

```
/* Read IP408 Digital Input Channel Register B */

UINT64 base_address;
volatile BYTE* pbase_addr;
WORD chan_val;

if (I408_GetBaseAddress(Handle, &base_address) == 0)
{
   pbase_addr = (BYTE*)base_address;
   chan_val = *(PWORD)(pbase_addr + 0x2);
}
```

Using the Library with Visual C++

This section describes the basic steps to add a IP Windows Driver dynamic link library to a Visual C++ project. These instructions are based on the Microsoft Visual Studio 2010 development environment.

32-bit application

1. Create a new or open an existing Visual C++ project.
2. Open the project's property pages
3. Confirm the Platform drop-down is set to "Win32" and the Configuration drop-down is set to "All Configurations."
4. Add the path to the necessary header files (APC86xx.h, IXXX.h, IErrorCodes.h) to the project. To do this, open the **Configuration Properties | C/C++ | General** property page and modify the **Additional Include Directories** property. This can be a relative path.

   e.g. **..\..\..\c_include;**

   Add the path to the 32-bit import libraries (APC86xx.lib, IXXX.lib) to the project.
   To do this, open the **Configuration Properties | Linker | General** property page and modify the **Additional Library Directories** property. This can be a relative path.

   e.g. **..\..\..\c_lib\Win32**

5. Select the **Configuration Properties | Linker | Input** property page and add the import libraries to the **Additional Dependencies** property

   e.g. **APC86xx.lib;I408.lib;**

6. Include the windows.h, APC86xx.h, IXXX.h and IErrorCodes.h header files at the beginning of your .c (C source code) or .cpp (C++ source code) files.

   e.g.
   ```
   #include <windows.h>
   #include "APC86xx.h"
   #include "I408.h"
   #include "IErrorCodes.h"
   ```

64-bit application

1. Open a new or existing Visual C++ project.
2. Open the project's property pages
3. Confirm the Platform drop-down is set to "x64" and the Configuration drop-down is set to "All Configurations." (If x64 is not available, you will need to create the new platform in Configuration Manager.)
4. Add the path to the necessary header files (APC86xx.h, IXXX.h, IErrorCodes.h) to the project. To do this, open the **Configuration Properties | C/C++ | General** property page and modify the **Additional Include Directories** property. This can be a relative path.

   e.g. **..\..\..\c_include;**

   Add the path to the 64-bit import libraries (APC86xx.lib, IXXX.lib) to the project.
   To do this, open the **Configuration Properties | Linker | General** property page and modify the **Additional Library Directories** property. This can be a relative path.

   e.g. **..\..\..\c_lib\Win64**

5. Select the **Configuration Properties | Linker | Input** property page and add the import library to the **Additional Dependencies** property

   e.g. **APC86xx.lib;I408.lib;**

6. Include the windows.h, APC86xx.h, IXXX.h and IErrorCodes.h header files at the beginning of your .c (C source code) or .cpp (C++ source code) files.

   e.g.
   ```
   #include <windows.h>
   #include "APC86xx.h"
   #include "I408.h"
   #include "IErrorCodes.h"
   ```

Deploying Applications to Target Systems

This section outlines how to move your custom application from the development system to a target system where the driver software is not currently installed.

Kernel Driver Installation

The kernel driver for the Acromag board is installed on the target using the procedure outlined in the **Hardware Installation** section. The complete driver software package does not need to be installed on the system. Direct the New Hardware or Update Driver Software wizard to the IP Windows Driver disk or to some other directory where the required files can be found.

The acrmgpci.sys driver will be installed to the \windows\system32\drivers directory.

Application and DLL Installation

The following files must be installed on the target machine.

- Your application program
- The carrier DLL and all DLL's corresponding to the IP modules you are using. Copy these from the \redist\DLLs folder on the development system. These are typically installed in your application's directory.
- The DLLs are dependent upon the Microsoft C Runtime Library (msvcr100.dll). If this file is not already present on the target, it can be copied from the \redist\DLLs folder on the development system Place the file in your application's directory.
- If your application program is dependent on additional components, those will need to be installed on the target system as well.

DLL Location Notes

To reduce the likelihood of "DLL Conflict" issues Microsoft recommends that DLLs be installed to the application directory with the program executable. This is the preferred location when running a single executable. However, if several applications will be simultaneously sharing a DLL it is recommended that the DLL be placed in a common directory.

In order for the operating system to find a DLL, its location must be part of the Windows search order. The typical search order is as follows:

1. The directory from which the application is loaded
2. The current directory
3. The Windows system directory (e.g., C:\Windows\system32 or C:\Windows\SysWow64)
4. The Windows directory (e.g., C\Windows)
5. The directories listed in the PATH environment variable

The easiest solution to sharing a DLL is to place it in the Windows or Windows system directory. However, many applications store DLLs in these directories so using these locations creates the most risk for DLL conflict issues.

The technique used by the IP Windows Driver Software installer is to append the common DLL directory (typically C:\Acromag\IPSW_API_WIN\redist\DLLs\Win32 or C:\Acromag\IPSW_API_WIN\redist\DLLs\Win64) to the PATH environment variable. This allows the appropriate DLL to be located when running each example project.

MODIFYING THE PATH SETTING

Use the following steps if you wish to modify the PATH setting on a target machine.

Windows XP, Vista, 7 and 8

1. Launch the System applet from the Windows Control Panel.
2. Select the Advanced link (or tab) and then click the Environment Variables button.
3. Locate "Path" in the User Variables or System Variables. The PATH is a series of one or more directories separated by semicolons.
4. Edit the variable by appending the path to the common DLL directory to the right of the existing value.
5. Click OK