AcroPack Windows Driver Software
User's Manual

ACROMAG INCORPORATED
30765 South Wixom Road
Wixom, MI 48393-2417 U.S.A.

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

8501-070E

The information in this document is subject to change without notice. Acromag, Inc., makes no warranty of any kind with regard to this material and accompanying software, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Further, Acromag, Inc., assumes no responsibility for any errors that may appear in this document and accompanying software and makes no commitment to update, or keep current, the information contained in this document. No part of this document may be copied or reproduced in any form, without the prior written consent of Acromag, Inc.

Contents

Contents 3
Introduction 4
Supported Hardware 5
Supported Programming Languages 5
Getting Started 6
Software Installation 6
Installed Software 6
Program Menu Shortcuts 6
Installed Files 6
Hardware Installation 7
AcroPack Enumeration Utility 8
Demonstration Programs 8
Where to Go From Here 8
Software Overview 9
Function Format 9
Status Codes 10
Sequence of Operations 12
Interrupts 13
Callback Functions 13
Synchronization 14
Base Address Pointers 14
Using the Library with Visual C++ 15
32-bit application 15
64-bit application 16
Deploying Applications to Target Systems 17
Kernel Driver Installation 17
Application and DLL Installation 17
DLL Location Notes 17
Modifying the PATH setting 18
Windows 7, 8 and 10 18
Revision History 19

Introduction

AcroPack Windows Driver Software consists of low-level drivers and Windows Dynamic Link Libraries (DLLs) that facilitate the development of Windows applications accessing Acromag AcroPack I/O modules.

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 APSW-API-WIN on a 32-bit Windows system and use it to develop an 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 hardware and software installation, programming concepts, application development and deployment issues. Also included in the AcroPack Window Driver documentation is a set of Function Reference documents for each AcroPack module DLL. After reviewing this user's manual, readers will next want to consult the Function Reference document specific to their hardware.

AP500, AP512, AP513, AP520, AP521, AP522 Serial Communication Modules Note:

AcroPack AP500, AP512, AP513, AP520, AP521 and AP522 modules use a device driver that allows them to be seen as standard COM ports by Windows. This driver is different than that used by all other AcroPack modules. Subsequently, the information presented in this manual does not apply to AP5xx boards. For information on installing these modules please see the included **AP5xx Windows Driver Installation** document.

Supported Hardware

The list of supported AcroPack modules is shown in Table1.

Table 1: Acromag AcroPack I/O Modules

Model	Description
AP220	12-Bit Analog Output Module
AP225	12-Bit, 16 Channel Analog Output Module
AP226	Isolated 12-Bit Analog Output Module
AP231	16-Bit High Density Analog Output Module
AP235	16-Bit, 16 Channel Analog Output Module
AP236	Isolated 16-Bit Analog Output Module
AP323	16-Bit High Density Analog Input Module
AP341	Simultaneous Sample and Hold Analog to Digital Converter Module
AP342	Isolated Simultaneous Sample and Hold Analog to Digital Converter Module
AP408	32-Channel Digital I/O Module
AP418	16-Channel Digital I/O Module
AP441-1/2/3	32-Channel Isolated Digital Input Module
AP445	32-Channel Isolated Digital Output Module
AP471	48-Channel Digital I/O Module
AP482	10 32-bit counters – TTL
AP483	5 16-bit counters – TTL and 3 32-bit counters – RS485
AP484	6 32-bit counters – RS485
AP560	Four Channel CAN Module
AP560A	Four Channel CAN Module
AP730	Multifunction I/O Module
APA7-201/202/203/204	Xilinx® Artix7® 7 series FPGA Based AcroPack Modules
APA7-501/502/503/504	Xilinx® 50T Artix7® 7 series FPGA Based AcroPack Modules
APZU-301/303/304	Xilinx® Zynq® Ultrascale+™ based AcroPack Modules

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 APSW-API DLL functions in a similar manner.

Getting Started

Software Installation

To install the AcroPack Windows Driver software on the development system, insert the software disk into the optical drive and run **Setup.exe**. Note that 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 directory containing executable demonstration programs (optional)

Installed Files

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

Subdirectory	
c_examples	Microsoft Visual C++ example projects with source code.
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 FPGA I/O Modules
demos_bin	Executable demo programs (segregated into Win32 and Win64 subdirectories). The source code can be found in the c_examples folder.
docs	User's manual, DLL references, revision history, application notes
redist\AP5xx	Driver packages for AP500, AP512, AP513, AP520, AP521 and AP522
redist\DLLs	32-bit and 64-bit DLLs (segregated into Win32 and Win64 subdirectories)
redist\Driver	INF and catalog files for each board, 32-bit and 64-bit device drivers and co-installer DLLs (segregated into Vista, Win7_8, Win10 and WinAll subdirectories)
utility	32-bit and 64-bit AcroPackEnum utilities (segregated into Win32 and Win64 subdirectories)

Hardware Installation

1. Configure any jumpers or switches on the AcroPack module as necessary.
2. With power off, install the module into an available slot on the target system. Connect any field wiring at this time.
3. Turn on the system.
4. Identify the folder that contains the correct driver files for your board and version of Windows. These folders are under the "redist\Driver" subdirectory (see table above).

   First look for your driver package in the WinAll folder. If it's not there choose the folder that matches your version of Window.
5. Install the driver package

   Windows 7, 8 and 10:
   a. Launch Device Manager from the Control Panel (or type devmgmt.msc in the Search Box)
   b. Locate and right-click the "PCI Data Acquisition and Signal Processing Controller," select Update Driver, and browse to the driver files in the folder identified in Step 4.

Notes

- The files in the Vista, Win7_8, Win10 and WinAll folders only differ in the way that they were digitally signed. The files in the WinAll folder were signed with an older certificate that is recognized by all current versions of Windows. When this certificate expired and was replaced, a new signing procedure needed to be adopted. With this procedure, signatures required for newer versions of Windows are not recognized by earlier versions of Windows.
- Windows 7 may not recognize the SHA2 catalog file signatures used in the Win7_8 folder if it is not current on system updates. If this problem is encountered either install the update to enable SHA2 support (recommended) or install the AcroPack driver package from the Vista folder.

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

  1. The INF file for the board (e.g.AP408.inf). The same INF file is used to install both the 32-bit and the 64-bit drivers.
  2. The platform specific catalog file for the board (e.g. Ap408_x64.cat). It should be placed in the same directory as the INF file.
  3. The platform specific kernel driver (acrmgpci.sys) and co-installer (acrmgcls.dll). The kernel driver and co-installer must be placed in a subdirectory relative to the INF and catalog files. The 32-bit files must be in a subdirectory named "i386" and the 64-bit files must be in subdirectory named "amd64."

AcroPack Enumeration Utility

AcroPack Windows Driver Software includes a command line utility, **AcroPackEnum.exe** which may be run to display basic information about all installed AcroPack modules. Running this utility is a quick way to verify that the device driver and boards are properly installed. The displayed information includes the board name, board number, location ID, and the address and length of each memory range present. The board number is the value passed to the `APXXX_Open` function to open a connection to the device. (See the **Sequence of Operations** section below.)

Demonstration Programs

Console demonstration programs (source code provided) are included for each Acromag board. Each demo project can be built as both a 32-bit and a 64-bit application. Use the demo program to test your board and to become familiar with how it operates.

Tip:
Some of the more complicated demos include instructions on the main menu to walk you through some basic operations. Rather than jump back and forth between the instructions and the other menus, open two instances of the program and use one to display the instructions and the other to communicate with the board.

In some cases the main demo source file (APxxxDemo.c) will include comments with additional information necessary for running the demo (e.g. how channels should be wired together).

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 board. You can use this code as a starting point for your custom application.

Software Overview

The software includes 32-bit and 64-bit DLLs for each AcroPack module. In most cases the name of the DLL matches the name of the AcroPack module. For example the AP408 is used with AP408.dll. There are a few exceptions, however, where groups of similar AcroPack modules are supported by a single DLL. These include:

AcroPack Modules	**Shared DLL**
AP220, AP226, AP231 and AP236	AP231.dll
AP225 and AP235	AP225.dll
AP341 and AP342	AP341.dll
Series AP441	AP441.dll
AP482, AP483 and AP484	AP482.dll
APA7-201, APA7-202, APA7-203 and APA7-204 APA7-501, APA7-502, APA7-503 and APA7-504	APA72.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. The 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 AcroPack DLL functions have the following form:

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

The "APXXX" portion of the function name indicates the AcroPack module the function is used with (e.g. AP464).

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 AcroPack module. (See the **Sequence of Operations** section below.)

STATUS CODES

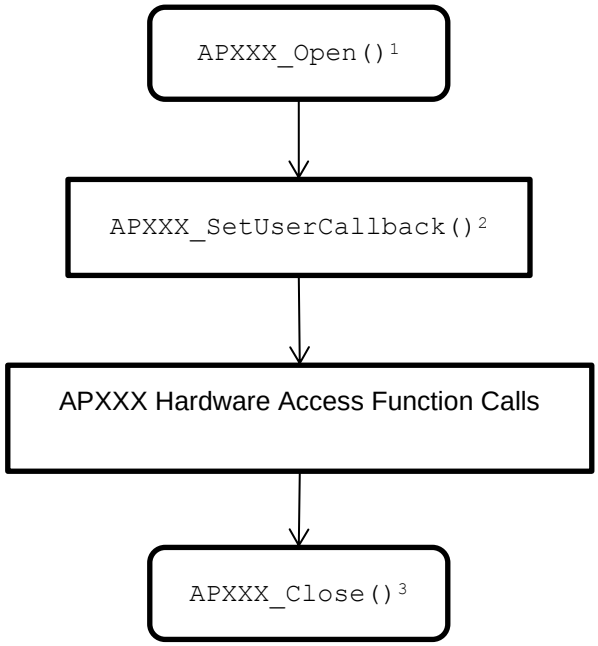
The table below summarizes the 32-bit status codes that may be returned from the DLL functions. Please note the return code of any failing functions if you should need to contact Acromag for technical support.

Value	Mnemonic	Description
0	ERR_OK	Operation Successful
-1	ERR_INVALID_HNDL	Returned if no board is associated with the specified handle. Applies to most functions.
-2	ERR_BOARD_IN_USE	Returned by *APXXX_Open()* if board is already open. This can occur if the board is in use by another application.
-4	ERR_CONNECT	Returned by *APXXX_Open()* if an error occurred connecting to the device. This will occur if the specified card number is invalid or if the kernel mode drivers are not properly installed or configured.
-5	ERR_MAPMEM	Returned by *APXXX_Open()* if an error occurred mapping the devices physical memory into the virtual address space.
-8	ERR_OUTOFHANDLES	Returned by *APXXX_Open()* if the maximum number of handles have already been allocated.
-9	ERR_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	ERR_INSUF_ RESOURCES	Returned by a function if there were insufficient resources to create a required data structure or carry out an operation.
-13	ERR_CONFIG_READ	Returned by *APXXX_ReadPCIReg* if an error occurred while reading data from the device's PCI configuration space.
-14	ERR_TIMEOUT	Returned by a function if it timed out before completing.
-15	ERR_CONFIG_SET	Returned by a Configuration function if the current settings used by this function do not represent a valid configuration
-16	ERR_CALIB	Indicates an error generating or using calibration data.
-17	ERR_BUFFER	Indicates an error occurred accessing a user defined data buffer
-18	ERR_CONFIG_WRITE	Returned by *APXXX_WritePCIReg* if an error occurred while writing data to the device's PCI configuration space.
-19	ERR_DMA_MAP	Returned by a DMA function if a DMA buffer is not mapped into the process
-20	ERR_EEPROM_ACK	Returned by EEPROM access functions if an Acknowledge is expected but not received
-21	ERR_READBACK	Returned by a write function if the value read back from the hardware does not match value written
-22	ERR_FILE_OPEN	Returned if a file cannot be opened
-23	ERR_FILE_FORMAT	Returned if file contents are not in the expected format
-24	ERR_FILE_READ	Returned if a file cannot be read
-25	ERR_CONFIG_DONE	Returned by functions if the configuration of a re-configurable board is not complete
-26	ERR_EX_DESIGN	Returned by functions if a re-configurable board is not configured with the Acromag supplied example design

-27	ERR_HARDWARE	Returned if the hardware indicates an error or malfunction occurred
-28	ERR_FLASH_BUSY	Returned if a flash operation cannot be carried out because the flash chip is busy
-29	ERR_UNSUPPORTED	Returned if the device does not support the requested action
-30	ERR_CHECKSUM	Returned if a checksum mismatch is detected
-31	ERR_HANDLER	Returned if the function requires an interrupt handler and one has not yet been attached
-35	ERR_INVALID_ CTRLHNDL	Returned if the kernel driver control handle is invalid
-36	ERR_EMPTY	Returned if an operation cannot be carried out because a hardware of software container (such as a FIFO or buffer) is empty
-37	ERR_FULL	Returned if an operation cannot be carried out because a hardware of software container (such as a FIFO or buffer) is full
-38	ERR_BUSY	Returned if an operation cannot be carried out because the device is busy

Sequence of Operations

Although each AcroPack module has its own DLL with unique functions, all AcroPack modules are accessed using the calling sequence shown below.



Notes:

1. APXXX_Open provides an integer "handle" used to access the specific APXXX module in all subsequent calls.
2. Not all Acromag AcroPack modules support interrupts. User callbacks are used to implement custom interrupt service routine logic for reconfigurable boards.
3. APXXX_Close should always be called for each "open" AcroPack module prior to terminating the application.

Interrupts

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

Each AcroPack DLL that supports interrupts has its own predefined internal interrupt service routine. (DLLs for reconfigurable boards are a special case. See note below.) The specifics of each routine are outlined in the AcroPack 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 board's Interrupt Enable bit and signals the DLL's internal interrupt service routine (ISR).
2. At this point three things can happen
    a. If no callback was configured, the ISR simply processes the interrupt and then re-enables the board's Interrupt Enable bit.
    b. If a callback function was configured but should not override the internal ISR, the internal ISR processes the interrupt, re-enables the board's Interrupt Enable bit and then invokes the callback.
    c. If a callback function was configured to override the internal ISR, the ISR invokes the callback and then immediately returns without further processing. It is then the responsibility of the callback function to process the interrupt and re-enable the Interrupt Enable bit.

Note
The DLLs for user programmable FPGA boards include only a bare-bones ISR. When using interrupts, applications for these boards must include a callback function to implement the custom ISR logic. In other word, overriding the internal ISR is mandatory.

CALLBACK FUNCTIONS

When using the callback mechanism your application defines a function that the 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 **APXXX_SetUserCallback** topic in the AcroPack 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 board 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. *APXXX_StartIsrSynch* and *APXXX_EndIsrSynch* can be used to synchronize other application threads with the ISR thread and to synchronize multiple threads within an application with each other.

Bracketing a section of code between calls of *APXXX_StartIsrSynch* and *APXXX_EndIsrSynch* defines that code as a device critical section. Two threads within a single process cannot execute critical section code simultaneously. *APXXX_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 *APXXX_EndIsrSynch* when finished accessing these shared resources.

Base Address Pointers

Each DLL provides a function that returns the base address of the user mode mapping of the AcroPack module's memory space.

C and C++ programmers can cast the returned value to a 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 AP408 Location ID Register */

UINT64 base_address;
volatile BYTE* pbase_addr;
BYTE locationID;

if (AP408_GetBaseAddress(Handle, &base_address) == 0)
{
   pbase_addr = (BYTE*)base_address;
   locationID = *(pbase_addr + 0x4);
}
```

Using the Library with Visual C++

This section describes the basic steps to add an AcroPack 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. Open a new or 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 (APXXX.h, PCIErrorCodes.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 library (APXXX.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 library to the **Additional Dependencies** property

   e.g. **AP408.lib**

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

   e.g.
   ```
   #include <windows.h>
   #include "AP408.h"
   #include "PCIErrorCodes.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 (APXXX.h, PCIErrorCodes.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 library (APXXX.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. **AP408.lib**

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

   e.g.
   ```
   #include <windows.h>
   #include "AP408.h"
   #include "PCIErrorCodes.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 AcroPack 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
- All DLL's corresponding to the AcroPack 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 AcroPack DLLs are dependent upon the Microsoft C Runtime Library (msvcr120.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 an AcroPack 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 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 AcroPack Windows Driver Software installer is to append the common DLL directory (typically C:\Acromag\APSW_API_WIN\redist\DLLs\Win32 or

C:\Acromag\APSW_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 7, 8 and 10

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

Revision History

The following table shows the revision history for this document.

Release Date	Version	EGR/DOC	Description of Revision
6 NOV 20	E	DGC/MJO	Corresponds to software revision 9500-488E. Added AP560A board to supported hardware list
24 OCT 19	D	DGC/MJO	Corresponds to software revision 9500-488D. Added AP730 and APZU-30x boards to supported hardware list
04 JAN 19	C	DGC/MJO	Corresponds to software revision 9500-488C. Added APA7-50x boards to supported hardware list and shared DLL table.
21 MAR 18	B	DGC/MJO	Corresponds to software revision 9500-488B. • Added ERR_BUSY to Status Codes section • Updated supported hardware list • Updated Hardware Installation instructions • Removed some text specific to Windows Vista
22 SEP 16	A	DGC/MJO	Initial Release. Corresponds to software revision 9500-488A.