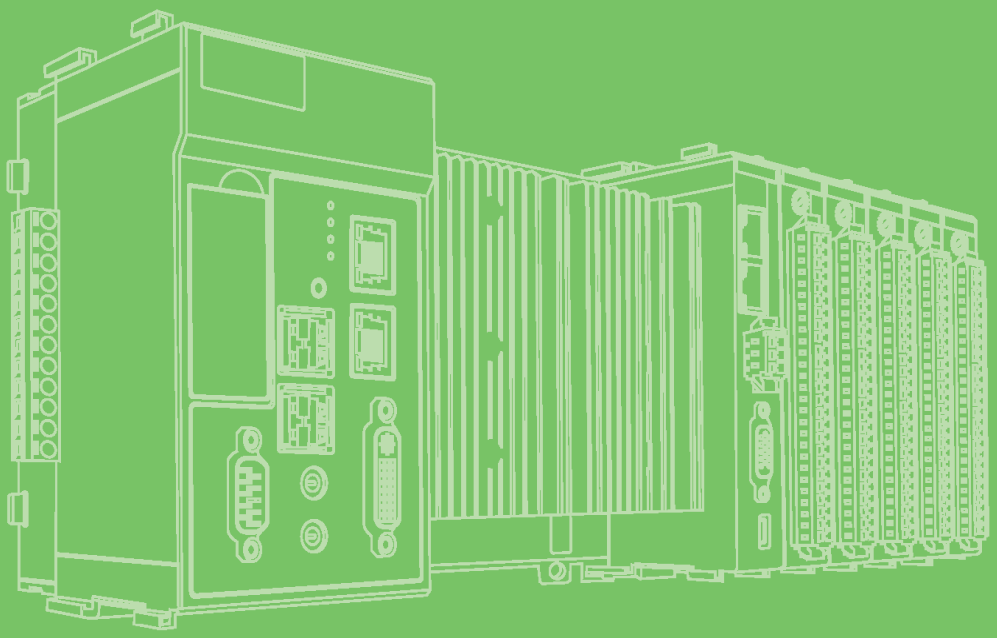
User Manual



APAX-5520CE

Software Manual

ADVANTECH
eAutomation

Copyright

Acknowledgements

Intel and Pentium are trademarks of Intel Corporation.

Microsoft Windows and MS-DOS are registered trademarks of Microsoft Corp.

MULTIPROG and ProConOS are registered trademarks of KW-Software GmbH Lemgo (Germany)

All other product names or trademarks are properties of their respective owners.

Notes on the Manual

This is the Software Manual for the Advantech APAX-5520 product. This manual will help guide the end user through implementation and use of the software portion of this product.

What is covered in this manual:

This manual will give a general overview of the Windows CE operating system, most of the applications that are included with Windows CE as well as the applications added and/or created by Advantech Corporation in the Windows CE image. This manual will also cover installation and use of development and utility software that is needed. It will also reference optional software that can be used by the end user with the Windows CE Operating system.

What is not covered in this manual:

This manual will reference the hardware but does not contain hardware setup information, wiring information, electrical specifications or any detailed hardware information. Please refer to the hardware manual for this information.

Edition 1

September 2009

Contents

Chapter 1

Windows CE.NET

1.1 Overview

APAX-5520CE is part of Advantech's PC-based Controller offering complete open platform with Windows CE .NET operating system. Leveraging powerful PC technology, APAX-5520CE delivers excellent integration ability with rich interface connectivity. Programmers can develop their applications under Microsoft Visual Studio .NET (by Advantech Apax.NET class library) or eVC (by Advantech C/C++ API) environment, and copy the executives to APAX-5520CE to run the control process.

1.2 WinCE Image

Advantech has engineered the Windows CE.NET embedded image exclusively for this hardware. It contains specific drivers for the APAX-5520 and is designed and licensed only for this hardware.

1.3 Modification of Standard Image

While the WinCE image is considered an embedded image, it is possible for the developer to add their own developed software to the image if done properly. This is possible through the Microsoft Visual Studio .NET programming environment. Users can create and deploy their own applications through this tool along with the libraries distributed by Advantech (see below for more information). Users can also make changes to an image and have that image deployed on subsequent purchased images through a Configure To Order specification (CTO). There may be a situation where a user needs modification of the standard image. Since the image is created by Advantech, this may be possible depending on the user requirements. A non reoccurring engineering fee (NRE) would most likely be required to create a custom image. Please check with your sales person for more information about the CTO and NRE services.

1.4 Connecting to the Device

1.4.1 DiagAnywhere

DiagAnywhere, an abbreviation of "Diagnostic Anywhere", is a networking solution for remotely monitoring and controlling other Windows based devices. It is very similar to a remote desktop application with some additional features. Currently, DiagAnywhere includes the utility on client side, and the server on the other. The main technology is based on Microsoft .NET Framework for the client. For this reason, the PCs using this solution must have the Microsoft .NET Framework installed for Win32 platform. You can find the .NET Framework and DiagAnywhere client trial version on the CD that comes with the APAX-5520.

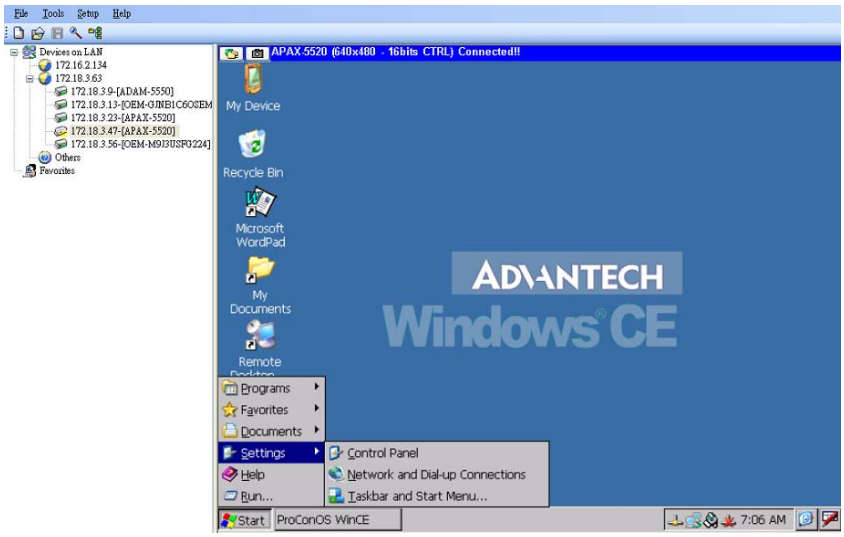
DiagAnywhere server can only run on Advantech's **TPC, UNO, AMAX, APAX** and **ADAM** Windows based devices. The supported platforms include Windows XP, Windows XPe and Windows CE.

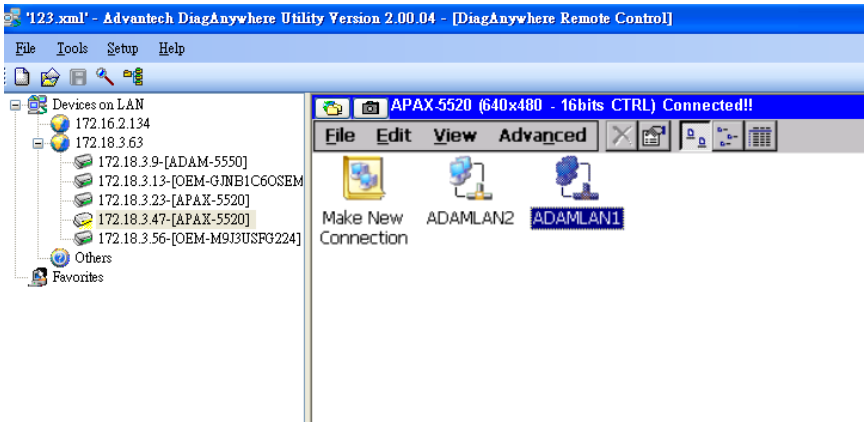
However, the server can accept only one connection from the utility at a time, and other connection attempts will be rejected if there is a live connection. This server is set up to automatically start when Windows CE starts. APAX-5520 has built-in DiaAnywhere server and the server will launch automatically after the system boots. You can use DiagAnywhere client (The trial version is provided in the CD) to connect to the APAX-5520. You need to type password when you connect to the APAX-5520. There is no default password. Users can remotely control the APAX-5520 through Ethernet, including file transferring.

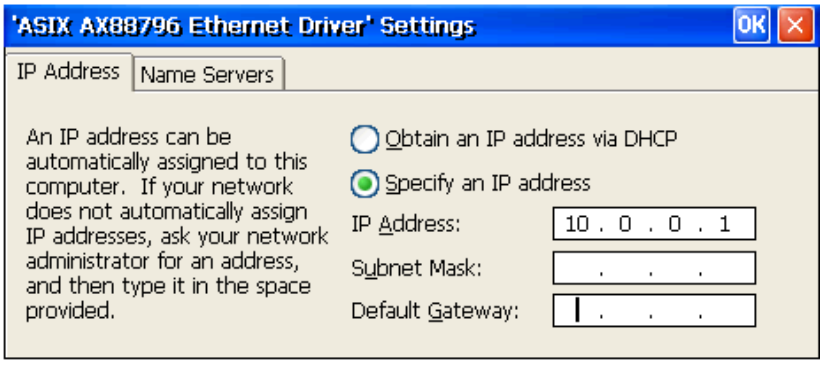
1.4.2 IP Address

The APAX-5520 will come with a default IP address set to 10.0.0.1 and 10.0.0.2. This IP address can be changed through or local VGA display to suit the users specific requirements. Refer to figure below. Double click the LAN port icon you want to change IP through Start>>Setting>>Control Panel>>Network and Dial-up Connection. You will see the configuration window as shown below. It is not recommended to use DHCP for the APAX-5520 because the project and other items connecting to the APAX-5520 will be programmed to specific IP address'.

Note! *You must save the registry after you update the IP address or your changes will be discarded upon reboot. This can be done from the start menu at Start | Programs | Advantech | Registry Saver.*







1.5 WinCE Remote Tools

WinCE Remote tools are a set of Microsoft administration tools provided via web server on the APAX-5520. The remote tools are accessed by a web browser. The IP address of the APAX-5520 must be known in order to use the remote tools. It is important that you find your IP address either by setting a static address or getting the DHCP assigned address.

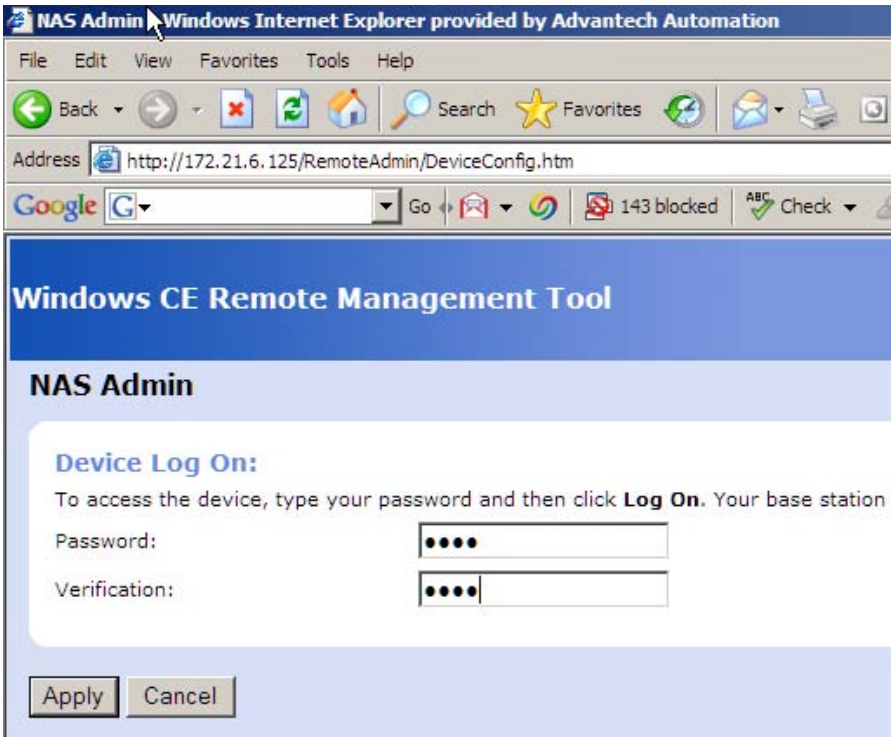
1.5.1 Remote Admin

Setup Administrator Password

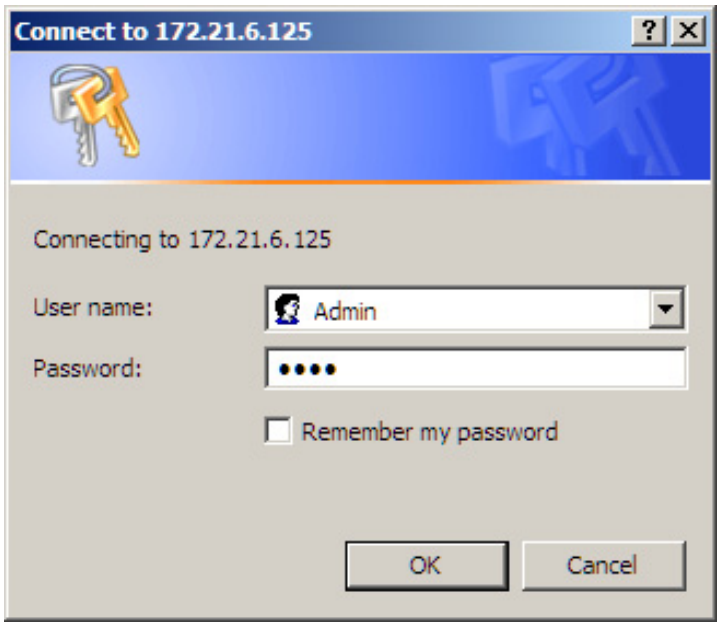
The first time remote admin is connected to, the user must enter an administrator password. It is important that this step is done to keep the APAX-5520 protected. If the registry is lost or if the defaults are loaded, then this step must be done again.

Connect to the APAX-5520 via a web browser with the IP address that was previously set. Using the path xxx.xxx.xxx.xxx/RemoteAdmin

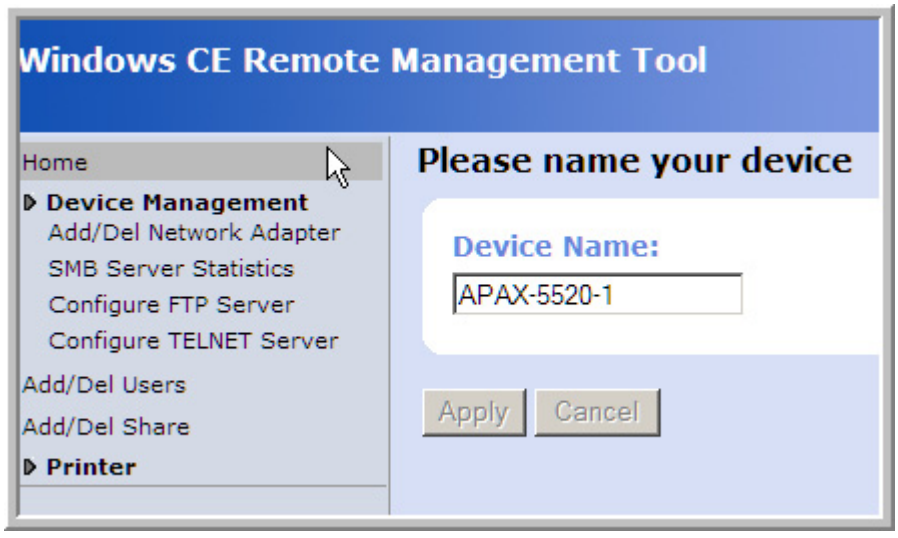
For the first time log on, the page will be redirected to the “DeviceConfig.htm” to set up the Admin password. Enter the Admin password and click the apply button.



When the Apply button is clicked, the gateway will reset and the user is then prompted to log in with the new password.



Once logged in, the user must change the device name. The device name box may have a sample such as APAX-5520. A suggestion is to change the name to “APAX-5520-1”. Other controllers on the same network can have subsequent numbers or different names but all controller names on the same network must be unique.



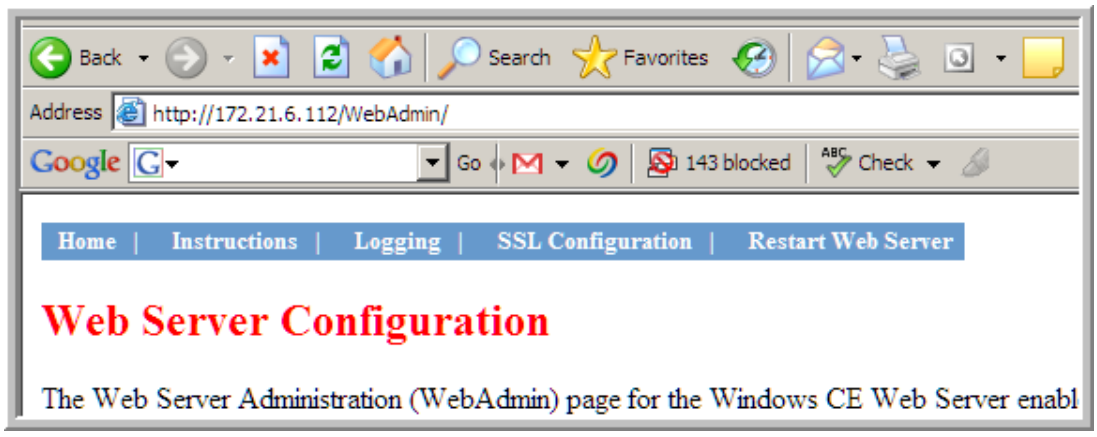
Once the device name is saved, the remote admin page will be displayed. From this page the following functions can be managed:

- Enable Network Adaptors for file share
- Configure FTP Server
- Configure TELNET Server
- Add/Delete Users
- Add/Delete file shares
- Add/Delete Printers

1.5.2 Remote Web Admin

Windows CE provides remote web server administration. This is located on a virtual root by typing in the address xxx.xxx.xxx.xxx/webadmin. The login and password will be the same for Remote Web Admin as the Remote Admin that was set in the previous section. The Web Server Administration (WebAdmin) page for the Windows CE Web Server enables you to remotely administer your Web server using your Web browser. Use WebAdmin to manage the accessibility, security, and file sharing features of your Web server, including the following tasks:

- Configure which files are shared and how they are accessed.
- Configure which users have access to which files.
- Configure the authentication protocols the Web server will use.
- View and configure the Web server log.



The web server configuration comes with its own instructions and help files. Please see these documents for further information.

1.5.3 Remote System Admin

Windows CE provides a remote system administration. This is located on a virtual root by typing in the address xxx.xxx.xxx.xxx/sysadmin. The login and password will be the same as the Remote Admin login and password. This interface includes the following tools:

- System Information viewer
- Process Management view and control
- File browser
- Registry editor

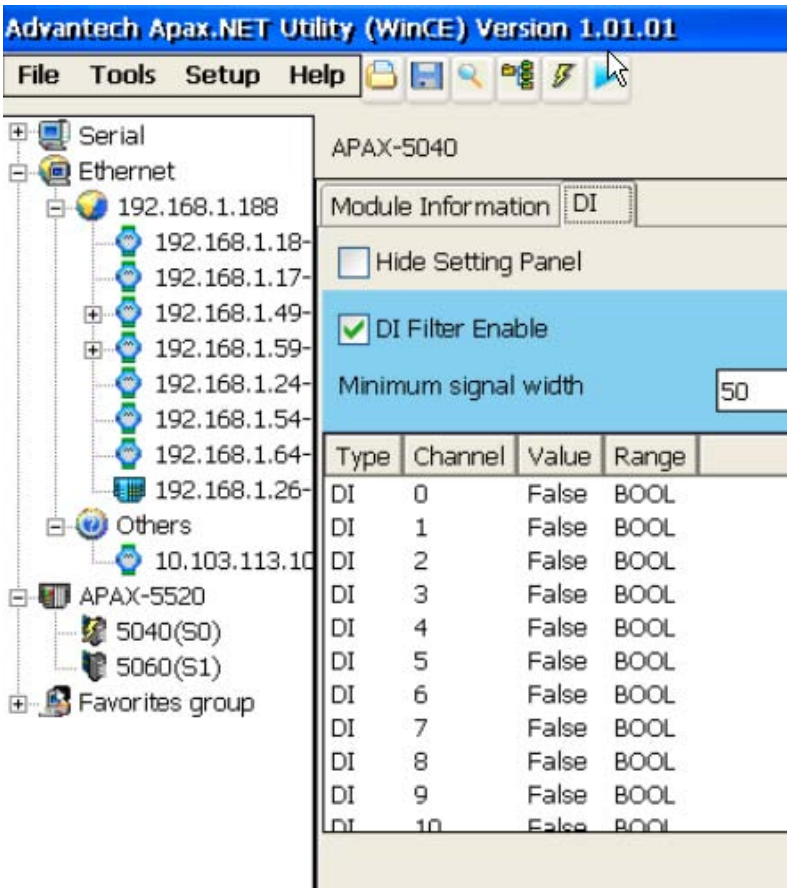
1.6 WinCE Applications

1.6.1 APAX.NET Utility

Advantech provides the APAX.NET utility which allows the developer/end user to interrogate the APAX bus, see connected modules and do simple testing of the I/O. This software can be helpful when checking wiring inputs prior to installing the runtime project. It is also able to detect and test other Advantech supported hardware for this product.

The installation file is contained in the CD and on our website at: *http://www.advantech.com* in the download area under the support page.

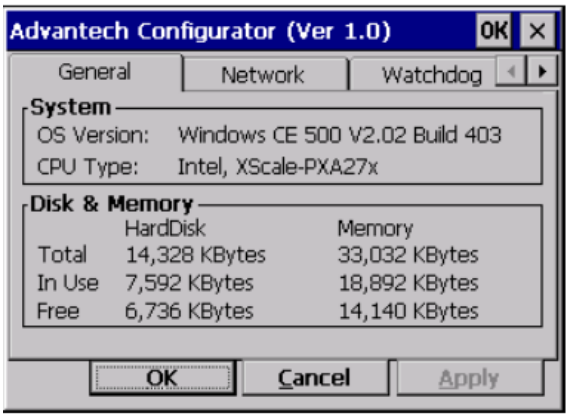
Detailed operation for APAX.NET utility can be found in Appendix A.



1.6.2 Advantech Configuration Utility

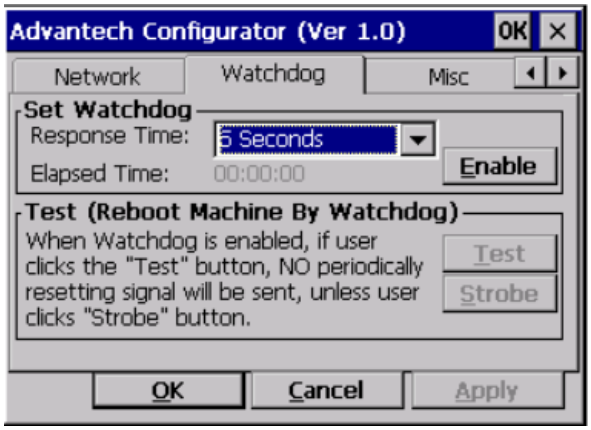
Advantech provides a tool called the Configuration Utility which can be accessed from the start menu by selecting Start >> Programs >> Advantech >> Configuration Utility. This tool provides the following items:

- General: System and disk information is available here.

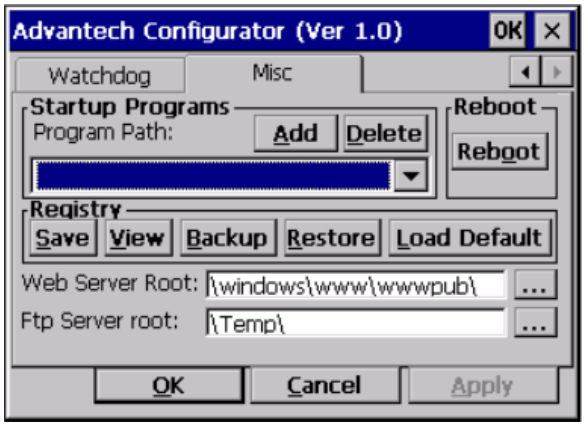


- Network: The two LAN port information (such as MAC address, IP address, Subnet Mask, etc) is available here. If you configure the LAN port as DHCP, click the **Renew** button to get another ID. Click the **Ping** button to ping another device in the same network. Click the **Advanced** button for further information such as DHCP server or DNS server.

- Watchdog: APAX-5520 offers built-in watchdog timer. It will continuously check the system and automatically reset the system if the system fails. Choose the periodical checking time for watchdog timer by the **Response Time** combo box and then enable the watchdog timer by the **Enable** button. Here, you also can test the watchdog timer.



- Miscellaneous: You can define which program application should execute automatically when system boot-up by including it in the Startup Program. Use **Add** and **Delete** to decide which programs become startup programs. There are other configuration for system such as Register, Web Server Root and FTP Server root. Click the **Reboot** button can help to reboot the system without power-off the system.



1.6.3 Advantech Version InformationTool

Advantech provides a simple reporting tool that will provide necessary version information for the Windows CE operating system as well as any post OS Build installations from Advantech. This is an important tool for determining what versions of Advantech Added software are on the APAX-5520 and may help during troubleshooting. Launch the Version Information Tool by selecting Start >> Programs >> Advantech >> Version Information

1.6.4 DiagAnywhere Server

APAX-5520 provides the DiagAnywhere Server to allow a connection from the DiagAnywhere client. The application is automatically started. If you choose not to use this program, you can disable the startup by using the Configuration Utility to remove it from startup. (Refer to Section 2.5.2)

Chapter 2

API Programming

2.1 C/C++ API

Advantech provides C/C++ API for eVC development environment to control APAX-5000 I/O modules. Remember to install eVC SDK to the eVC environment on your development computer. The eVC SDK is included in CD. After you launch the CD, select the **APAX Software** button and click the **VC++ Example** button, then you can find it under **SDK** folder. On APAX-5520CE, the runtime DLL has been installed already. You also can upgrade the runtime DLL (ApaxSys.cab) by execute a new cab file on APAX-5520CE. You always can find the latest runtime DLL by linking to Advantech website at http://www.advantech.com (in the download area under Support page). The following sections will explain these API functions.

In order to save your development time, Advantech provides several examples that you can use it as reference to build your own eVC application program. These examples can be found in the CD offered by APAX-5520CE, or from Advantech website at http://www.advantech.com (in the download area under Support page.) When you launch the CD, select the **APAX Software** button and click the **VC++ Example** button to find these examples.

2.1.1 ADAMDrvOpen

LONG ADS_API ADAMDrvOpen(LONG* handle);

Purpose:

Initialize the driver

Parameters:

handle = driver handle

Return

1. ERR_SUCCESS, Driver initialization succeeded, the handle will be valid for function use until closed.
2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.2 ADAMDrvClose

LONG ADS_API ADAMDrvClose(LONG* handle);

Purpose:

Initialize the driver

Parameters:

handle = driver handle

Return

1. ERR_SUCCESS, Driver termination succeeded
2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.3 SYS_SetInnerTimeout

LONG ADS_API SYS_SetInnerTimeout(LONG handle, WORD i_wTimeout);

Purpose:

Set the inner-timeout of the configuration functions that use internal communication channel. All functions with the exception of Get/Set values, use the internal communication network. When using any of those functions, they must wait for completion before returning. This sets the timeout value for returning.

Parameters:

handle = driver handle

i_wTimeout = inner-timeout, in millisecond. Default is 50 milliseconds.

Return

1. ERR_SUCCESS, Set timer succeeded
2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.4 SYS_GetModuleID

LONG ADS_API SYS_GetModuleID (LONG handle, WORD i_wSlot, DWORD* o_dwModuleID);

Purpose:

Get the module ID of the indicated slot.

Parameters:

handle = driver handle

i_wSlot = the slot ID which is ranged from 0 to 31.

o_dwModuleID = returned module ID

Return

1. ERR_SUCCESS, Module ID was found and returned
2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.5 SYS_GetSlotInfo

LONG ADS_API SYS_GetSlotInfo (LONG handle, WORD i_wSlot, struct SlotInfo* o_stSlotInfo);

Purpose:

Get the module information of the indicated slot.

Parameters:

handle = driver handle

i_wSlot = the slot ID which is ranged from 0 to 31.

o_stSlotInfo = returned SlotInfo structure.

Return

1. ERR_SUCCESS, o_stSlotInfo contains slot information.
2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.6 SYS_GetAllSlotErrorFlag

LONG ADS_API SYS_GetAllSlotErrorFlag(LONG handle,

DWORD* o_wError);

Purpose:

Get the presence of a module for each slot.

Parameters:

handle = driver handle

o_wError = Return value for all slots status. From LSB to MSB of the value indicates the slot-0 to slot-31 status. If the bit is 1, it means that the slot has no module present.

Return

1. ERR_SUCCESS,
2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.7 AIO_GetValue

LONG ADS_API AIO_GetValue(LONG handle,
WORD i_wSlot,
WORD i_wChannel,
WORD* o_wValue);

Purpose:

Get a single analog input or output value from the indicated slot and channel.

Parameters:

handle = driver handle

i_wSlot = the slot ID which has a range of 0 to 31.

i_wChannel = the channel ID which has a range of 0 to 31.

o_wValue = the variable to hold the returned AIO value.

Return

1. ERR_SUCCESS, o_wValue contains AIO value.
2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.8 AIO_GetValues

LONG ADS_API AIO_GetValues(LONG handle,
WORD i_wSlot,
WORD* o_wValues);

Purpose:

Get the all analog input or output values of the indicated slot.

Parameters:

handle = driver handle

i_wSlot = the slot ID which is ranged from 0 to 31.

o_wValues = the variables array to hold the returned AIO values. The size of this array must be at least 32 WORD's.

Return

1. ERR_SUCCESS, o_wValue contains AIO values from channel-0 to the last channel.
2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.9 AIO_SetRanges

LONG ADS_API AIO_SetRanges(LONG handle,
WORD i_wSlot,
WORD i_wChannelTotal,
WORD* i_wRanges);

Purpose:

Set the channel ranges of the indicated slot.

Parameters:

handle = driver handle

i_wSlot = the slot ID which is ranged from 0 to 31.

i_wChannelTotal = the channel total of the module in the indicated slot.

i_wRanges = the ranges to be set. The size of this array must be i_wChannelTotal WORDs. See APPENDIX A for valid range settings.

Return

1. ERR_SUCCESS, setting ranges succeeded.
2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.10 AIO_SetZeroCalibration

LONG ADS_API AIO_SetZeroCalibration(LONG handle,
WORD i_wSlot,
WORD i_wChannel,
WORD i_wType);

Purpose:

Run the zero calibraion of the indicated slot and channel.

Parameters:

handle = driver handle

i_wSlot = the slot ID which is ranged from 0 to 31.

i_wChannel = the channel ID which is ranged from 0 to 31.

i_wType = the type value to be set. Currently, it is ingnored.

Return

1. ERR_SUCCESS, setting zero calibration succeeded.
2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.11 AIO_SetSpanCalibration

LONG ADS_API AIO_SetSpanCalibration(LONG handle,
WORD i_wSlot,
WORD i_wChannel,
WORD i_wType);

Purpose:

Run the span calibraion of the indicated slot and channel.

Parameters:

handle = driver handle

i_wSlot = the slot ID which is ranged from 0 to 31.

i_wChannel = the channel ID which is ranged from 0 to 31.

i_wType = the type value to be set. Currently, it is ingnored.

Return

1. ERR_SUCCESS, setting span calibration succeeded.
2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.12 AIO_GetChannelStatus

LONG ADS_API AIO_GetChannelStatus(LONG handle,
WORD i_wSlot,
BYTE* o_byStatus);

Purpose:

Get all channels status of the indicated slot.

Parameters:

handle = driver handle

i_wSlot = the slot ID which is ranged from 0 to 31.

o_byStatus = the array to hold the returned channels status. The size of this array must be at least 32 BYTEs.

Return

1. ERR_SUCCESS, channel status succeeded.

The value of o_byStatus indicates:

0: None

1: Normal

2: Over current

3: Under current

4: Burn out

5: Open loop

6: Not ready

2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.13 AI_SetChannelMask

LONG ADS_API AI_SetChannelMask(LONG handle,
WORD i_wSlot,
DWORD i_dwMask);

Purpose:

Set enabled AI channel mask of the indicated slot.

Parameters:

handle = driver handle

i_wSlot = the slot ID which is ranged from 0 to 31.

i_dwMask = the enabled AI channel mask to be set. From LSB to MSB of the value indicate the slot-0 to slot-31 enabled mask. If the bit is 1, it means that the channel is enabled.

Return

1. ERR_SUCCESS, setting channel mask succeeded.
2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.14 AI_SetIntegrationTime

LONG ADS_API AI_SetIntegrationTime(LONG handle,
WORD i_wSlot,
DWORD i_dwIntegration);

Purpose:

Set AI integration time of the indicated slot.

Parameters:

handle = driver handle

i_wSlot = the slot ID which is ranged from 0 to 31.

i_dwIntegration = the AI integration time to be set. Two settings are availabled

0 = 60Hz

1 = 50Hz

Return

1. ERR_SUCCESS, setting integration time succeeded.
2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.15 AI_SetAutoCalibration

LONG ADS_API AI_SetAutoCalibration(LONG handle,
WORD i_wSlot);

Purpose:

Set to run the auto calibraion of the indicated slot.

Parameters:

handle = driver handle

i_wSlot = the slot ID which is ranged from 0 to 31.

Return

1. ERR_SUCCESS, setting auto calibration succeeded.
2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.16 AO_SetCalibrationMode

LONG ADS_API AO_SetCalibrationMode(LONG handle,

WORD i_wSlot);

Purpose:

Set to switch to the AO calibraion mode of the indicated slot.

Parameters:

handle = driver handle

i_wSlot = the slot ID which is ranged from 0 to 31.

Return

1. ERR_SUCCESS, setting calibration mode succeeded.
2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.17 AO_GetStartupValues

LONG ADS_API AO_GetStartupValues(LONG handle,

WORD i_wSlot,

WORD i_wChannelTotal,

WORD* o_wValues);

Purpose:

Get the AO startup values of the indicated slot.

Parameters:

handle = driver handle

i_wSlot = the slot ID which is ranged from 0 to 31.

i_wChannelTotal = the channel total of the module in the indicated slot.

o_wValues = the variables array to hold the AO startup values. The size of this array must be at least 32 WORDs.

Return

1. ERR_SUCCESS, Geting values succeeded. o_wValues contains AO startup values from channel-0 to the last channel.
2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.18 AO_SetStartupValues

LONG ADS_API AO_SetStartupValues(LONG handle,

WORD i_wSlot,

WORD i_wChannelTotal,

WORD* i_wValues);

Purpose:

Set the AO startup values of the indicated slot. These values are stored in onboard flash and are initialized to the slot upon boot up of the hardware.

Parameters:

handle = driver handle

i_wSlot = the slot ID which is ranged from 0 to 31.

i_wChannelTotal = the channel total of the module in the indicated slot.

i_wValues = the values array to be set. The size of this array must be i_wChannelTotal WORDs.

Return

1. ERR_SUCCESS, setting values succeeded.
2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.19 AO_SetValue

LONG ADS_API AO_SetValue(LONG handle,

WORD i_wSlot,

WORD i_wChannel,

WORD i_wValue);

Purpose:

Set a single AO value of the indicated slot and channel.

Parameters:

handle = driver handle

i_wSlot = the slot ID which is ranged from 0 to 31.

i_wChannel = the channel ID which is ranged from 0 to 31.

i_wValue = the AO value to be set.

Return

1. ERR_SUCCESS,
2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.20 AO_SetValues

LONG ADS_API AO_SetValues(LONG handle,

WORD i_wSlot,

DWORD i_dwMask,

WORD* i_wValues);

Purpose:

Set multiple AO values of the indicated slot.

Parameters:

handle = driver handle

i_wSlot = the slot ID which is ranged from 0 to 31.

i_dwMask = the channels mask. From LSB to MSB of the value indicate the slot-0 to slot-31 mask. If the bit is 1, it means that the channel must change value.

i_wValues = the AO values to be set. This is a pointer to an array of 32 words.

Return

1. ERR_SUCCESS, setting values succeeded.
2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.21 AO_BufValues

LONG ADS_API AO_BufValues(LONG handle,

WORD i_wSlot,

DWORD i_dwMask,

WORD* i_wValues);

Purpose:

Buffer the AO values of the indicated slot. This function is used along with OUT_FlushBufValues for a synchronized write Output. Once all slots are buffered, then OUT_FlushBufValues function triggers the synchronized buffer write of all masked slots.

Parameters:

handle = driver handle

i_wSlot = the slot ID which is ranged from 0 to 31.

i_dwMask = the channels mask. From LSB to MSB of the value indicate the slot-0 to slot-31 mask. If the bit is 1, it means that the channel must buffer value.

i_wValues = the AO values to be bufferred.

Return

1. ERR_SUCCESS, bufferring values succeeded.
2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.22 DIO_GetValue

LONG ADS_API DIO_GetValue(LONG handle,
WORD i_wSlot,
WORD i_wChannel,
BOOL* o_bValue);

Purpose:

Get a single DIO value of the indicated slot and channel.

Parameters:

handle = driver handle

i_wSlot = the slot ID which is ranged from 0 to 31.

i_wChannel = the channel ID which is ranged from 0 to 31.

o_bValue = the variable to hold the DIO value.

Return

1. ERR_SUCCESS, geting value succeeded o_wValue contains DIO value.
2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.23 DIO_GetValues

LONG ADS_API DIO_GetValues(LONG handle,
WORD i_wSlot,
DWORD* o_dwHighValue,
DWORD* o_dwLowValue);

Purpose:

Get the all DIO values of the indicated slot.

Parameters:

handle = driver handle

i_wSlot = the slot ID which is ranged from 0 to 31.

o_dwHighValue = the variable to hold the returned DIO values from channel 32 to 63. The LSB indicates the channel-32.

o_dwLowValue = the variable to hold the returned DIO values from channel 0 to 31. The LSB indicates the channel-0.

Return

1. ERR_SUCCESS, Get values succeeded. The o_dwHighValue and o_dwLowValue contain DIO values from channel-0 to the last channel.
2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.24 DI_GetFilters

LONG ADS_API DI_GetFilters(LONG handle,

WORD i_wSlot,

DWORD* o_dwHighMask,

DWORD* o_dwLowMask,

DWORD* o_dwWidth);

Purpose:

Get the DI filter mask and width of the indicated slot. All channels use the same filter.

Parameters:

handle = driver handle

i_wSlot = the slot ID which is ranged from 0 to 31.

o_dwHighMask = RESERVED

o_dwLowMask = If set to zero, filter is disabled. Non-zero indicates that filter is applied.

o_dwWidth = the variable to hold the DI filter width. Filter is in .1msec units and value of filter width must be in multiples of 5.

Return

1. ERR_SUCCESS, get filters succeeded.
2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.25 DI_SetFilters

LONG ADS_API DI_SetFilters(LONG handle,

WORD i_wSlot,

DWORD i_dwHighMask,

DWORD i_dwLowMask,

DWORD i_dwWidth);

Purpose:

Set the DI filter mask and width of the indicated slot. Filter is amount of time needed to verify a change of state. This is to reduce noise.

Parameters:

handle = driver handle

i_wSlot = the slot ID which is ranged from 0 to 31.

i_dwHighMask = RESERVED

i_dwLowMask = If set to zero, filter is disabled. Non-zero indicates that filter is applied.

i_dwWidth = the variable to hold the DI filter width. Filter is in .1msec units and value of filter width must be in multiples of 5.

Return

1. ERR_SUCCESS, setting filter succeeded.
2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.26 DO_SetValue

LONG ADS_API DO_SetValue(LONG handle,
WORD i_wSlot,
WORD i_wChannel,
BOOL i_bValue);

Purpose:

Set a single DO value of the indicated slot and channel.

Parameters:

handle = driver handle
i_wSlot = the slot ID which is ranged from 0 to 31.
i_wChannel = the channel ID which is ranged from 0 to 31.
i_bValue = the DO value to be set.

Return

1) ERR_SUCCESS, setting value succeeded.
2) ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.27 DO_SetValues

LONG ADS_API DO_SetValues(LONG handle,
WORD i_wSlot,
DWORD i_dwHighValue,
DWORD i_dwLowValue);

Purpose:

Set all DO values of the indicated slot.

Parameters:

handle = driver handle

i_wSlot = the slot ID which is ranged from 0 to 31.

i_dwHighValue = the DI values from channel 32 to 63 to be set. The LSB indicates the channel-32.

i_dwLowValue = the DI values from channel 0 to 31 to be set. The LSB indicates the channel-0.

Return

1. ERR_SUCCESS, setting values succeeded.
2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.28 DO_BufValues

LONG ADS_API DO_BufValues(LONG handle,

WORD i_wSlot,

DWORD i_dwHighValue,

DWORD i_dwLowValue);

Purpose:

Buffer the DO values of the indicated slot.

Parameters:

handle = driver handle

i_wSlot = the slot ID which is ranged from 0 to 31.

i_dwHighValue = the DI values from channel 32 to 63 to be bufferred. The LSB indicates the channel-32.

i_dwLowValue = the DI values from channel 0 to 31 to be bufferred. The LSB indicates the channel-0.

Return

1. ERR_SUCCESS, bufferring values succeeded.
2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.29 OUT_FlushBufValues

LONG ADS_API OUT_FlushBufValues(LONG handle,

DWORD i_dwSlotMask);

Purpose:

Flush the bufferred values. This triggers all buffered values to write simultaniously.

Parameters:

handle = driver handle

i_dwSlotMask = the flush slot enable mask. The LSB indicates the slot-0.

Return

1. ERR_SUCCESS, flushing values succeeded.
2. ERR_INTERNAL_FAILED, Call GetLastError to get extended error information.

2.1.30 Modbus Function

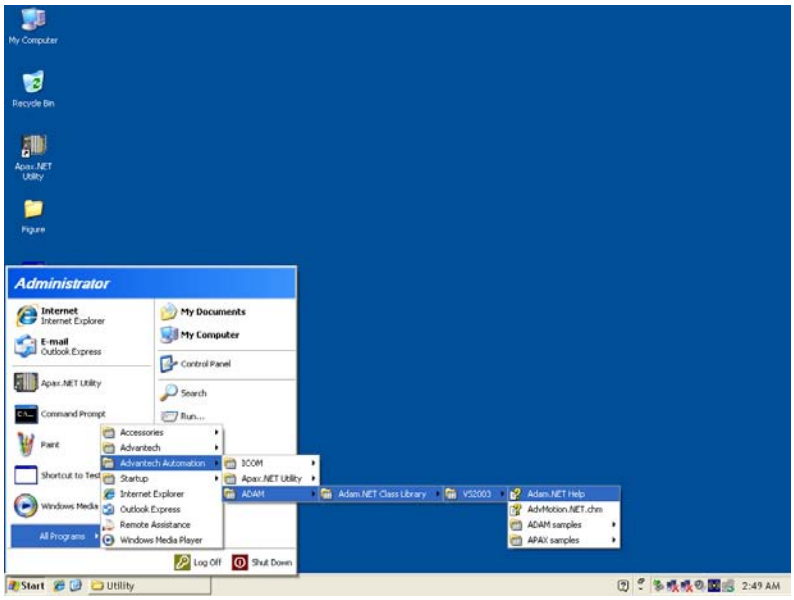
All Modbus functions' reference manual is located in the CD offered by APAX-557X. When you launch the CD, click the **Browser Manual** button and the you can see the document **APAX Modbus Library Manual.pdf** there.

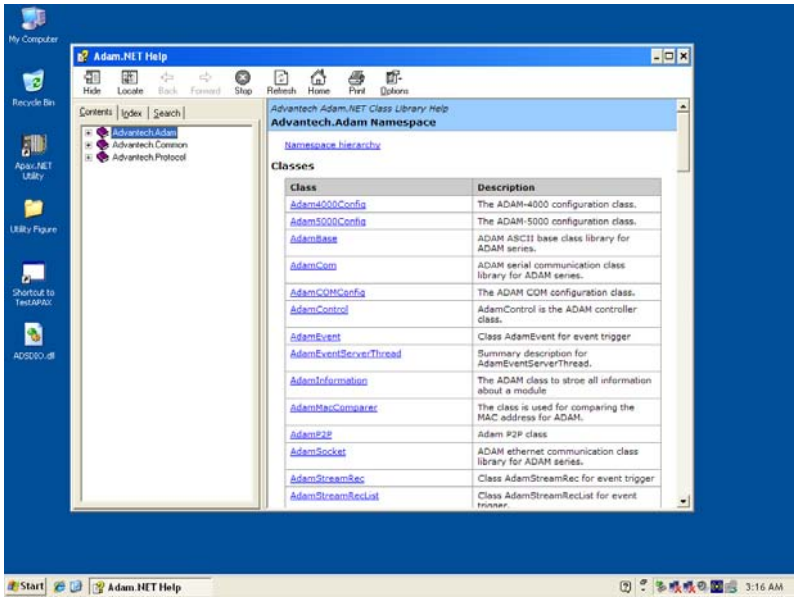
2.2 .NET API (Adam .NET Class Library)

Advantech provides a .NET API for developing .NET applications for many Advantech products. This API interface is called Adam .NET Class library. All the functions described in Section 3.1 are supported by Adam .NET class library. You can leverage Advantech Adam .NET class library to develop application controlling APAX-5000 I/O modules under Microsoft Visual Studio .NET environment such as VB .NET or C#.

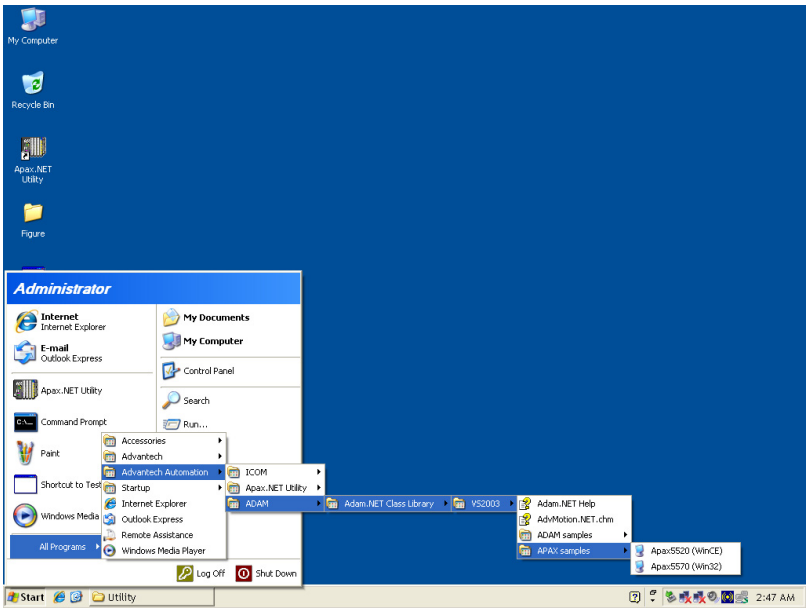
The installation file is contained in the CD. When you launch the CD, select the **APAX Software** button and click the **.NET Class Library** button to find the installation file. Besides, you also can link to the website: http://www.advantech.com and click into the Download Area under the Support website to get the latest version of the Adam.NET class library.

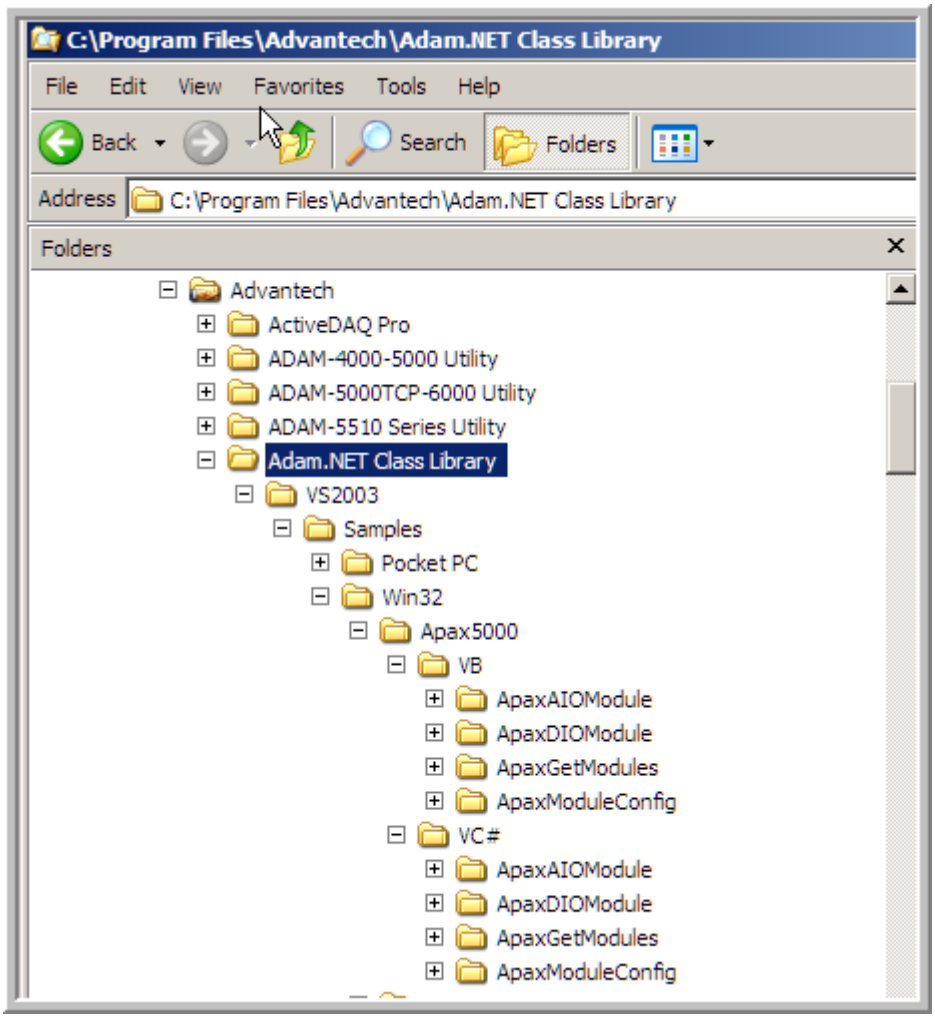
After you complete the installation, you can find Adam .NET class library help document by selecting Start >> All Programs >> Advantech Automation >> ADAM >> Adam.NET Class Library >> VS2003 >> Adam.NET Help.





Besides, there are many examples offered that you can use it as reference to build your own application program. These examples can also be found by selecting Start >> All Programs >> Advantech Automation >> ADAM >> Adam.NET Class Library >> VS2003 >> APAX samples >> Apax5520 (WinCE) after you have installed Adam.NET Class library. Or you can find these examples by C:\Program Files\Advantech\Adam.NET Class Library\.



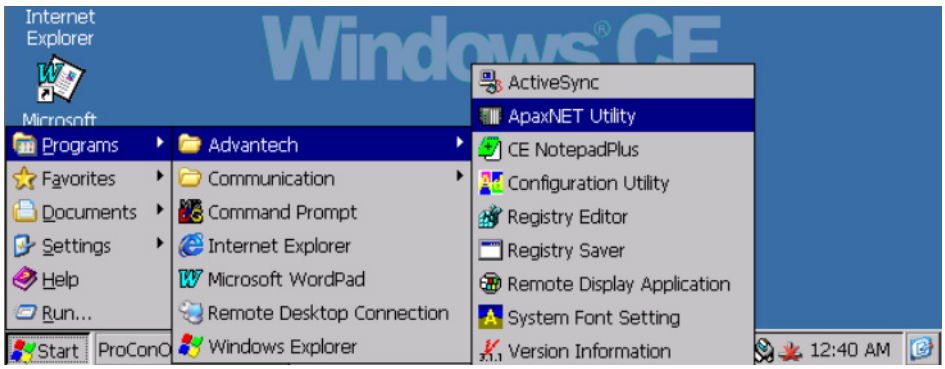


Appendix A

APAX.NET Utility Operation

A.1 APAX.NET Utility General Window

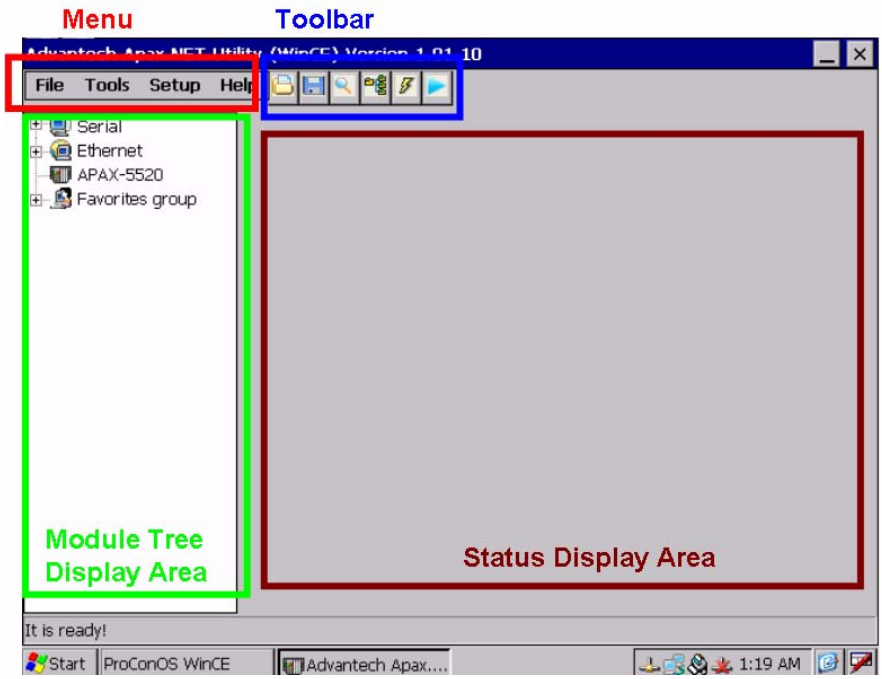
After you install the APAX.NET utility, you can launch it through Start>>Programs>>Advantech>>ApaxNET Utility. Refer to Section 1.6.1 for installation information. Or you can click the shortcut in the HarDisk folder under My Device.



Note! *It may take around 15 ~ 20 second to launch the utility.*

Note! *We suggest to close APAX.NET utility after you complete your configuration to release system memory for other applications.*

After you launch the utility, you should see the operation window as figure below. Except APXA-5000 I/O modules, other devices such as ADAM-4000, ADAM-5000 and ADAM-6000 modules can also be searched and configured in this utility.



The operation window consists of four areas --- the **Menu**, the **Toolbar**, the **Module Tree Display Area** and the **Status Display Area**.

A.1.1 Menu

The menu at the top of the operation window contains:

- The **File** menu

1. **Open Favorite Group** - You can configure your favorite group and save the configuration into one file. Using this option, you can load your configuration file for favorite group.
2. **Save Favorite Group** - You can configure your favorite group and save the configuration into one file. Using this option, you can save your favorite group into one configuration file.
3. **Auto-Initial Group** - If you want to have the same favorite group configuration when you exit APAX.NET utility and launch it again, you need to check this option.
4. **Exit** - Exit APAX.NET Utility.

- The **Tools** menu

1. **Search** - Search if there are any remote I/O modules connected. For I/O modules communicated by serial (such as ADAM-4000 modules), click the **COM1** item (COM 2 is an internal COM port) under **Serial** item in the **Module Tree Display Area** first before you click this button. For I/O modules communicated by Ethernet (such as ADAM-6000 modules), click the **Ethernet** item in the **Module Tree Display Area** first before you click this button.
2. **Add Devices to Group** - You can add any I/O modules to your favorite group by this option. You need to select the device you want to add in the **Module Tree Display Area** (it will be described below) first, and then select this option to add.
3. **Terminal for Command Testing** - ADAM modules support ASCII commands and Modbus as communication protocol. You can launch the terminal to communicate with remote module by these two kinds of protocols directly. Refer to ADAM-4000, ADAM-5000 and ADAM-6000 manual for ASCII and Modbus command.
4. **Monitor Stream/ Event Data** - ADAM-6000 modules support Data Stream function. You can use this to configure related setting for the connected ADAM-6000 modules connected. Refer to ADAM-6000 manual for more detail.

- The **Setup** menu

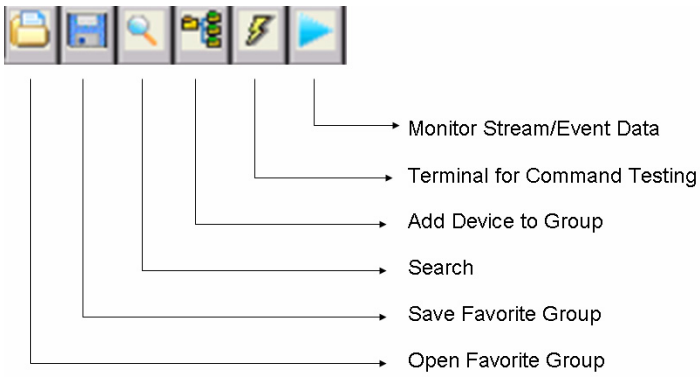
1. **Favorite Group** - You can configure your favorite group including add one new device (only for remote device), modify or delete one current device, sort current devices and diagnose connection to one device.
2. **Refresh COM and LAN node** - APAX.NET utility will refresh the serial and LAN network connection situation.
3. **ShowTreeView** - Check this option to display the **Module Tree Display Area** or not.
4. **Add COM Port Tree Nodes** - This option is used to add serial COM ports in APAX.NET Utility.
5. **Delete the COM Port** - This option is used to delete serial COM ports in APAX.NET Utility.

- The **Help** menu

1. **Check Up-to-Date on the Web** - Choose this option, it will automatically connect to Advantech download website. You can download the latest utility there.
2. **About Apax.NET Utility** - Choose this option, you can see version of APAX.NET Utility installed on your computer.

A.1.2 Toolbar

The six buttons on the toolbar represent the six commonly used items from the **Menu**. Refer to figure below for the definition of each button:



A.1.3 Module Tree Display Area

APAX.NET Utility is one complete software tool that all APAX and ADAM I/O module can be configure and operated in this utility. The **Module Tree Display Area** is on the left part of the utility operation window. There are four categories in the **Module Tree Display Area**:

- **Serial**

All serial remote I/O Modules connected to APAX-5520 will be listed in this category. You also can configure COM port parameter (such as baud rate, parity, stop bit, etc.) here.

- **Ethernet**

All Ethernet remote I/O Modules connected to APAX-5520 will be listed in this category.

- **APAX-5520**

All APAX-5000 local I/O modules in the same system will be listed in this category. Simply click this item all related modules will be displayed automatically.

- **Favorite Group**

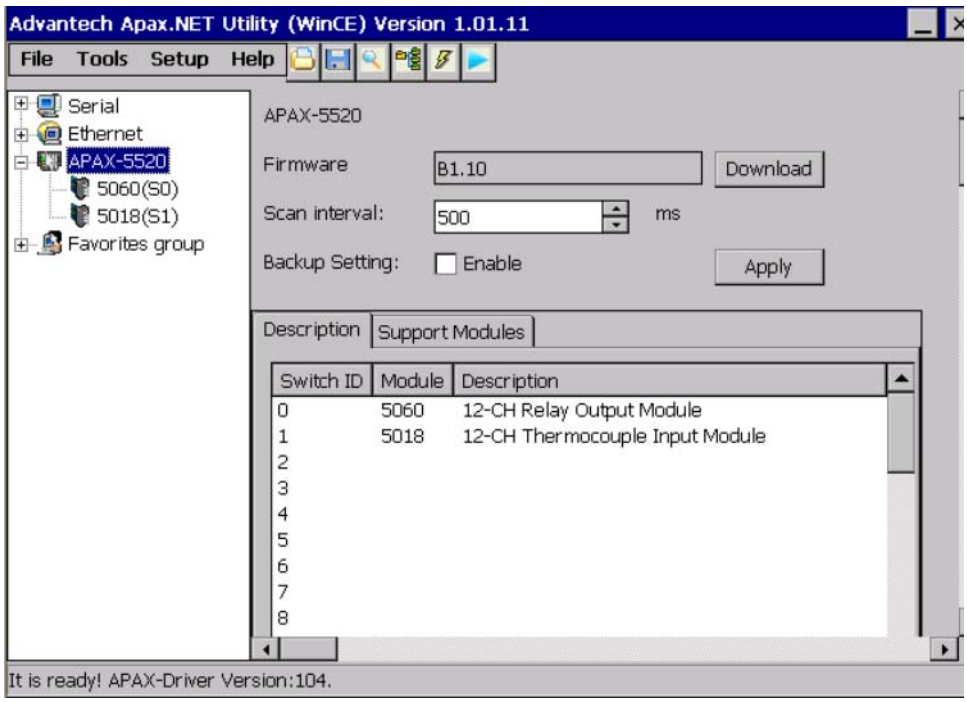
You can define which devices listed in **Serial** or **Ethernet** categories above into your personal favorite group. This will make you easier to find your interested modules. Click on the **ADAM device** item under **Favorite group** item, and select **Favorite** >> **New** in Setup menu to create a new group. After you create your own group, click on your group and select **Favorite >> New** in **Setup** menu to add any remote devices into your group. You can also select **Diagnose connection** to check the communication.

A.1.4 Status Display Area

Status Display Area, on the right part of utility operation window, is the main screen for operation. When you select different items in **Modules Tree Display Area**, **Status Display Area** will change dependently. You can do all configurations and tests on this area.

A.2 General Configuration

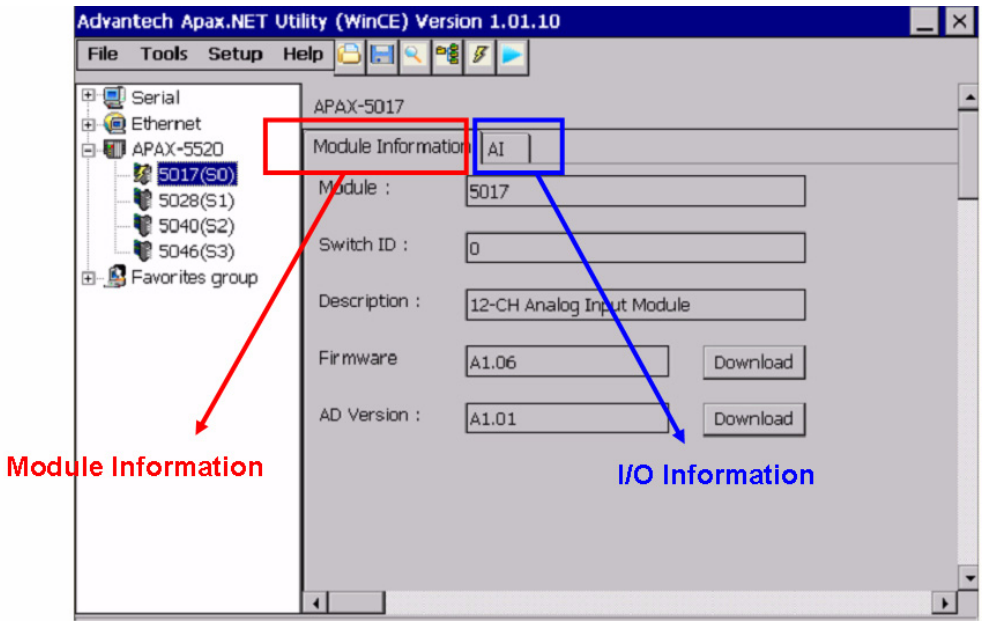
If you click the **APAX-5520** item in the **Module Tree Display Area**, the **Status Display Area** should look similar to the figure below:



All I/O modules with its ID number are listed in the **Description** tab in the **Module Tree Display Area** (the left tab) and **Description** tab on **Status Display Area** (the right tab). You can see all I/O modules supported by APAX-5520 by the **Support Modules** tab on **Status Display Area**. The **Backup Setting** check box is used to enable or disable APAX-5520 backup function. Refer to Appendix B for more detail about backup functionality.

A.3 I/O Modules Configuration

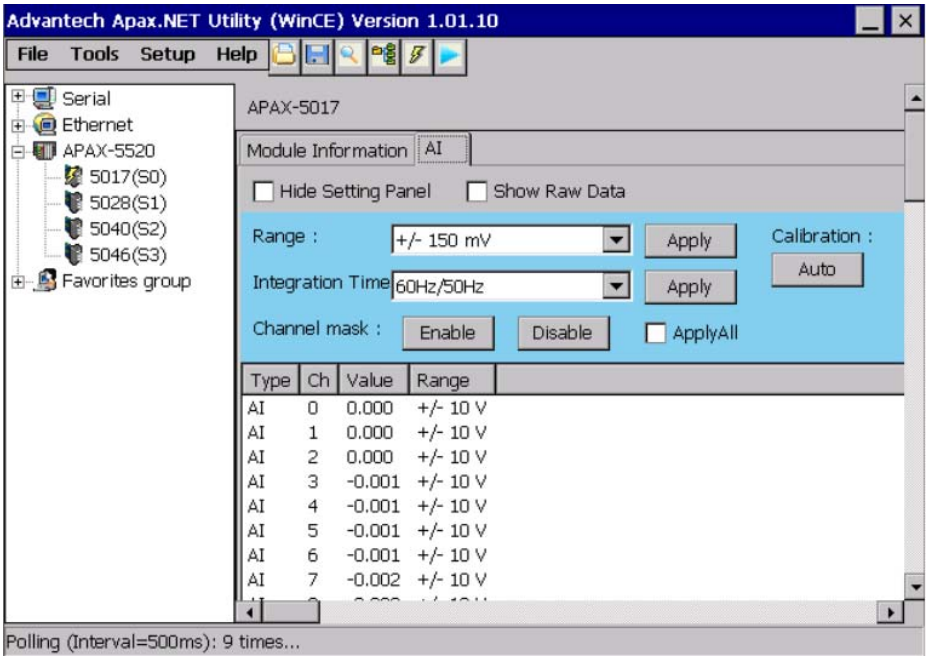
When you click any I/O module in the **Module Tree Display Area**, the **Status Display Area** at the right side will automatically change to show the module's information. There will be two tabs displayed: **Module Information** and **I/O Information**. (Refer to the figure below)



On the **Module Information** tab, information such as module name, switch ID, module description, and firmware version is displayed. You also can update related firmware to the specific module by the **Download** button.

On the **I/O Information** tab, you can write or read all channels' status and perform related configuration and calibration. Refer to the sections below for more detail.

A.3.1 Analog Input Modules

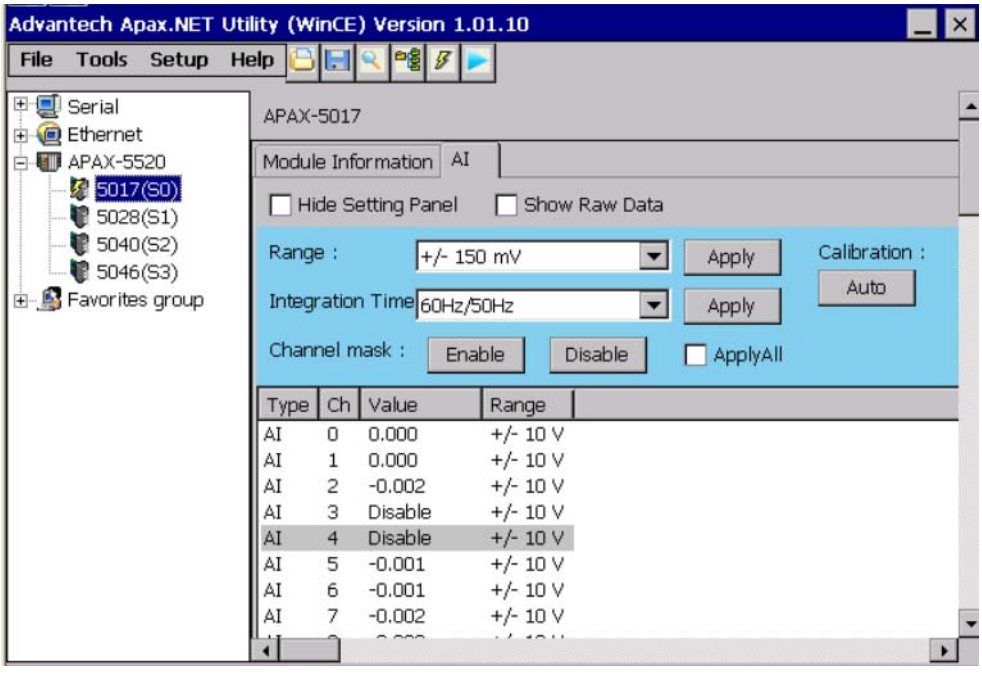


There are two parts for the **I/O Information** tab of APAX-5000 AI module. At the bottom is the **Channel Status** Area. You can see all channels' type, value, and range. Above the **Channel Status** Area is the **Setting Panel** Area. If you don't want see the **Setting Panel** Area, you can click the **Hide Setting Panel** check box to hide the **Setting Panel** Area. If you want to see the raw data (presented in Hexadecimal format) from the input channels, click the **Show Raw Data** check box.

If you want to configure specific input channels' range or integration time, select the channels in the **Channel Status** Area. Choose appropriate range and integration time by the **Range** and **Integration Time** combo boxes in the **Setting Panel** Area and then click the **Apply** button to save the configuration. If you want to save the same range setting for all channels, click the **ApplyAll** check box before you click the **Apply** button.

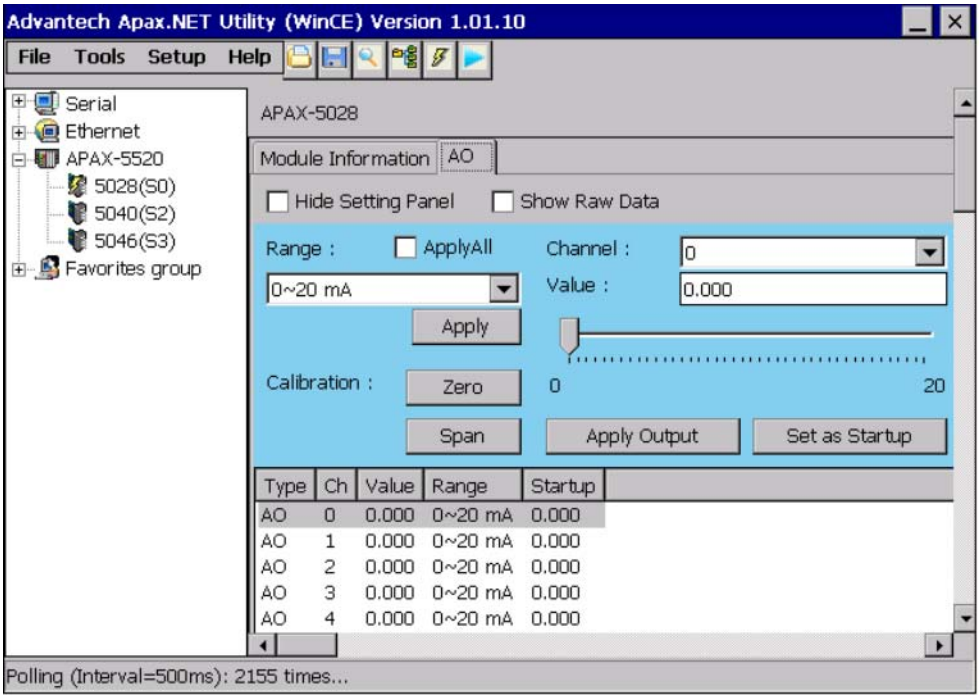
Note! *In order to remove the noise from the power supply, APAX AI modules feature built-in filter. Filters are used to remove noise generated from environment. The integration time is used to configure the filter frequency.*

You can define specific channels reading or not by the **Enable** and **Disable** button. Refer to figure below, channel 3 and channel 4 are disabled that no data will be read.



By clicking on the **Auto** button, you can perform auto calibration to the AI module. The module will automatically calibrate itself.

A.3.2 Analog Output Module

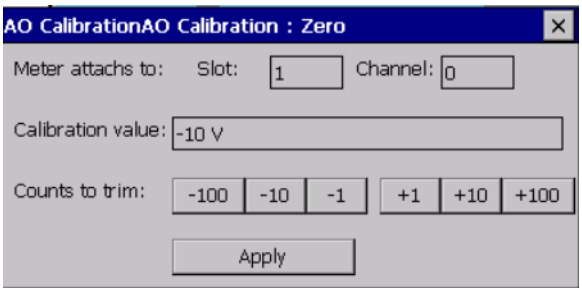


There are two parts for the **I/O Information** tab of APAX-5000 AO module. At the bottom is the **Channel Status** Area. You can see all channels' type, value, range and startup value (the initial value when the AO module is power-on). Above the **Channel Status** Area is the **Setting Panel** Area. If you don't want see the **Setting Panel** Area, you can click the **Hide Setting Panel** check box to hide the **Setting Panel** Area. If you want to see the raw data (presented in Hexadecimal format) from the output channels, click the **Show Raw Data** check box.

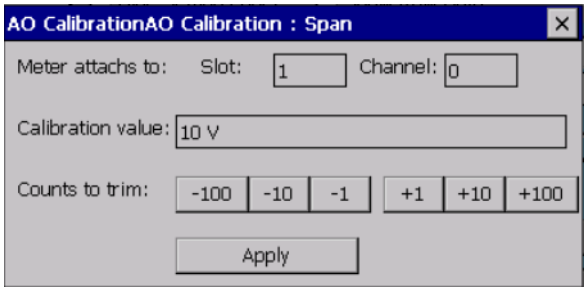
If you want to configure specific output channels' range, select the channels in the **Channel Status** Area. Choose appropriate range by the **Range** combo box in the **Setting Panel** Area and then click the **Apply** button to save the configuration. If you want to save the same range setting for all channels, click the **ApplyAll** check box before you click the **Apply** button.

If you want to change specific output channel' output value, select that channel by clicking the channel in the **Channel Status** Area or choosing it from **Channel** combo box in the **Setting Panel** Area. Then define the output value by the **Value** text box or the horizontal slide below in the **Setting Panel** Area. Then, click the **Apply** button to save the configuration. You can see the channel output value changed in the **Channel Status** Area. Similarly, you can save the value in the **Value** text box to become the startup value by the **Set as Startup** button. And you also can see the startup value changed in the **Channel Status** Area.

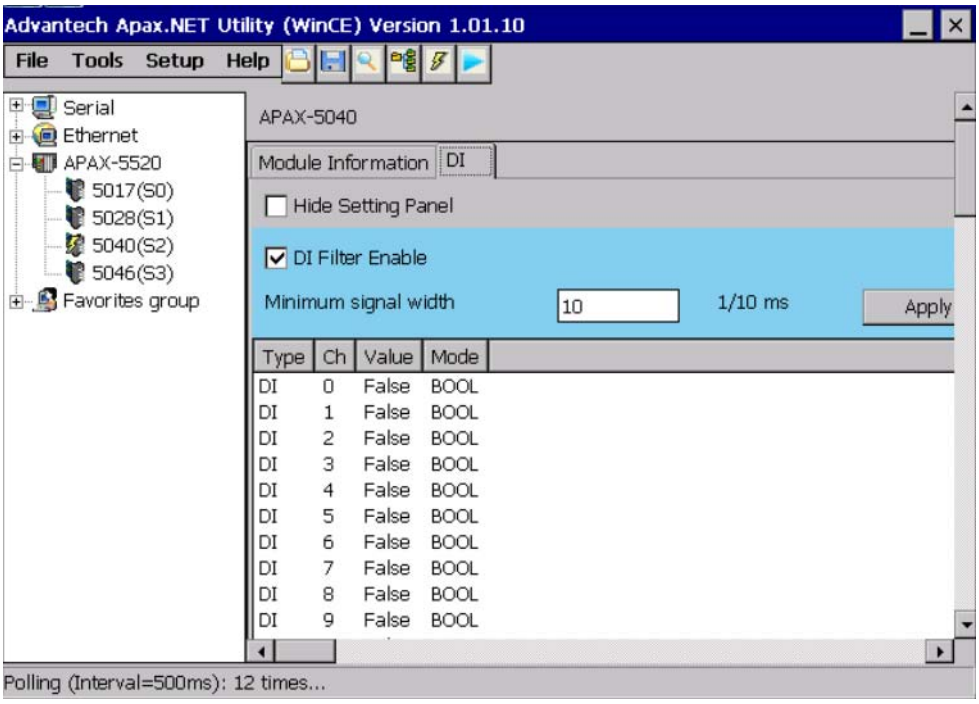
By clicking the **Span** button and **Zero** button, you can perform span calibration and zero calibration, separately. When you click the **Zero** button, you will see a dialog popping-up as figure below. The specific channel will generate output signal using the minimum value within range which is shown in the **Calibration Value** text box. Connect that channel to an external accurate instrument and measure the output signal. Using the **Counts to trim** buttons to adjust until the output value real matches the value in the **Calibration Value** text box. Then click the **Apply** button to save the calibration configuration.



When you click the **Span** button, you will see a dialog popping-up as figure below. The specific channel will generate output signal using the maximum value within range which is shown in the **Calibration Value** text box. Connect that channel to an external accurate instrument and measure the output signal. Using the **Counts to trim** buttons to adjust until the output value real matches the value in the **Calibration Value** text box. Then click the **Apply** button to save the calibration configuration.



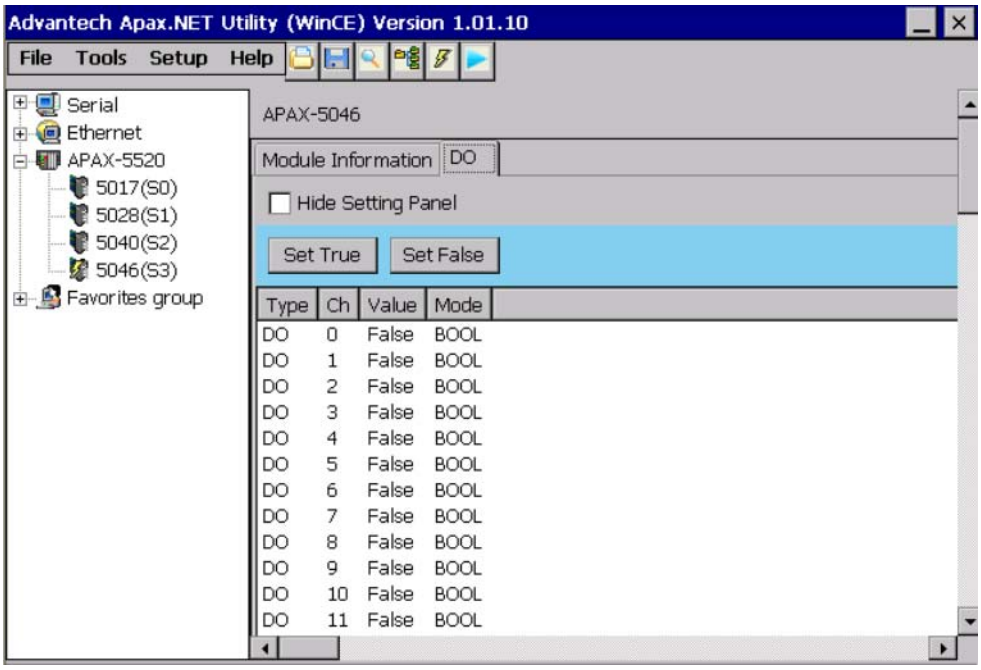
A.3.3 Digital Input Module



There are two parts for the **I/O Information** tab of APAX-5000 DI module. At the bottom is the **Channel Status** Area. You can see all channels' type, value, and mode. Above the **Channel Status** Area is the **Setting Panel** Area. If you don't want see the **Setting Panel** Area, you can click the **Hide Setting Panel** check box to hide the **Setting Panel** Area.

If you want to configure specific output channels' filter status or minimum acceptable pulse width, select the channels in the **Channel Status** Area. Click the **DI Filter Enable** check box in the **Setting Panel** Area to enable filter for that channel. Type the appropriate value (unit: 0.1 ms) into the **Minimum signal width** text box to configure acceptable minimum pulse width in the **Setting Panel** Area. After you complete the configuration, click the **Apply** button to save the configuration.

A.3.4 Digital Output Module



There are two parts for the **I/O Information** tab of APAX-5000 DO module. At the bottom is the **Channel Status** Area. You can see all channels' type, value, and mode. Above the **Channel Status** Area is the **Setting Panel** Area. If you don't want see the **Setting Panel** Area, you can click the **Hide Setting Panel** check box to hide the **Setting Panel** Area.

If you want to change specific output channels' output value, select those channels by clicking the channel in the **Channel Status** Area. Then define the output value by the **Set True** button or **Set False** button in the **Setting Panel** Area. Then, click the **Apply** button to save the configuration. You can see the channel output value changed in the **Channel Status** Area.

Appendix B

System Backup Functionality

B.1 Introduction

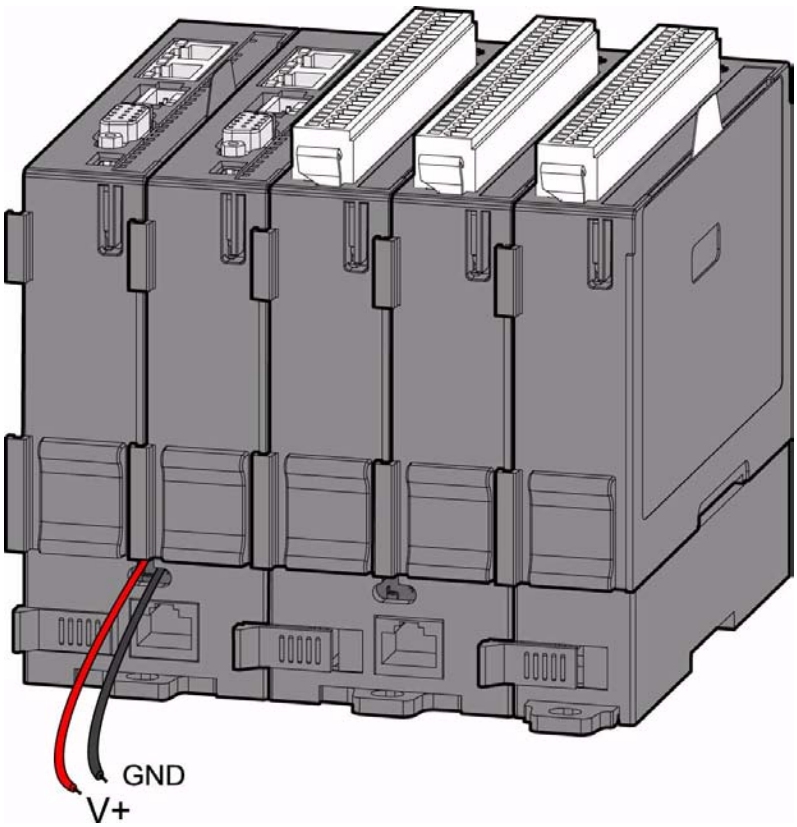
APAX-5520 series delivers system backup functionality. To leverage this functionality, two CPU modules (controllers), with the same control program, are installed in one system. After both controllers' backup function is enabled, the APAX-5000 system will automatically delegate one of the two controllers as the master controller.

The master controller will run the control program to execute the control process, while another controller (the backup controller) is put on standby. The master controller will periodically send living message to the backup controller. If the backup controller dose not receive living message from master controller over 500 milliseconds, it will automatically become master controller and take the control responsibility and restarts the control process execution. The maximum operation time for the backup controller to become master controller (the take over time) won't be greater than 1.5 second.

Changing master controller means there is something wrong for the previous master controller. Therefore, engineers can check or change the previous master controller with a new one and enable it to have backup functionality, becoming second backup controller. Then if the new master controller fails again, the second backup controller will automatically take the control responsibility.

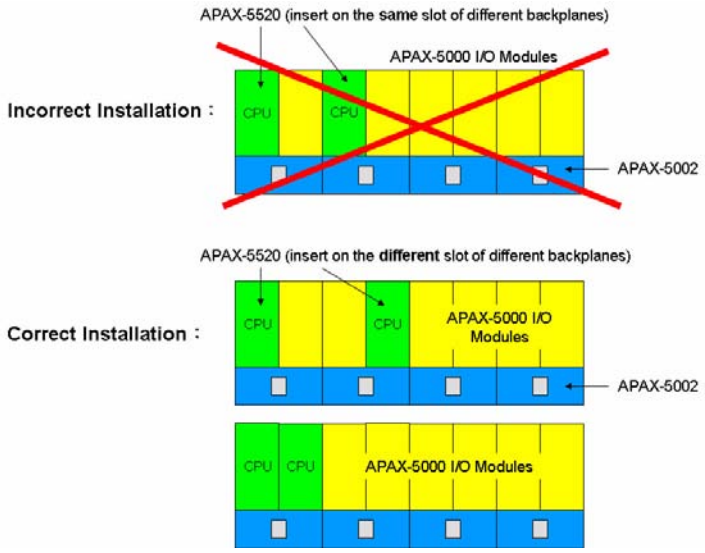
This mechanism ensures the control system will continuously run the control process. And the system won't be stopped even if controller fails.

B.2 Configuration

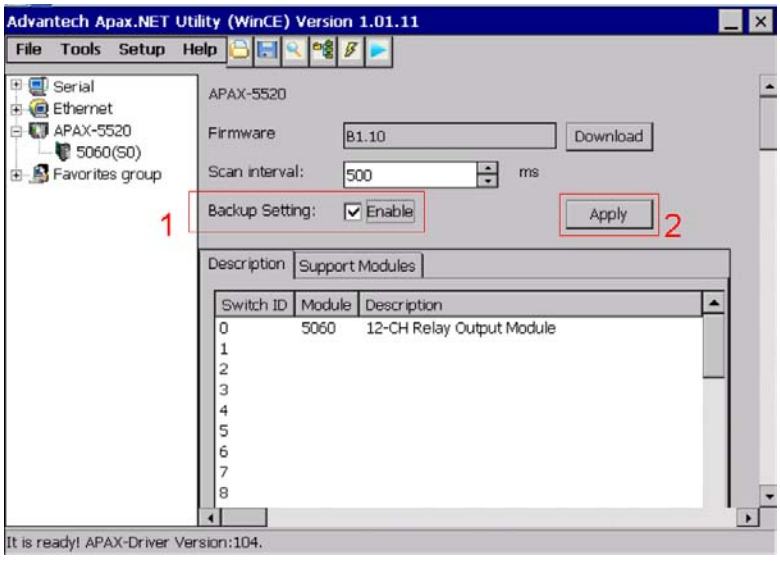


As you can see in the figure above, two APAX-5520 CPU modules are installed in one system. APAX-5000 series will automatically decide which one is the master controller.

Warning! *The controller ID of the APAX-5520 module is auto-identified by the location where the module is inserted on the APAX-5002 backplane (Slot 1 or Slot 2). Thus, be sure NOT to insert two APAX-5520 modules in the same location on two backplanes. For example, if you insert one APAX-5520 on slot 1 of one APAX-5002 backplane and insert the second APAX-5520 on slot 1 of another APAX-5002 backplane in the same system, APAX-5000 series cannot distinguish the two modules.*



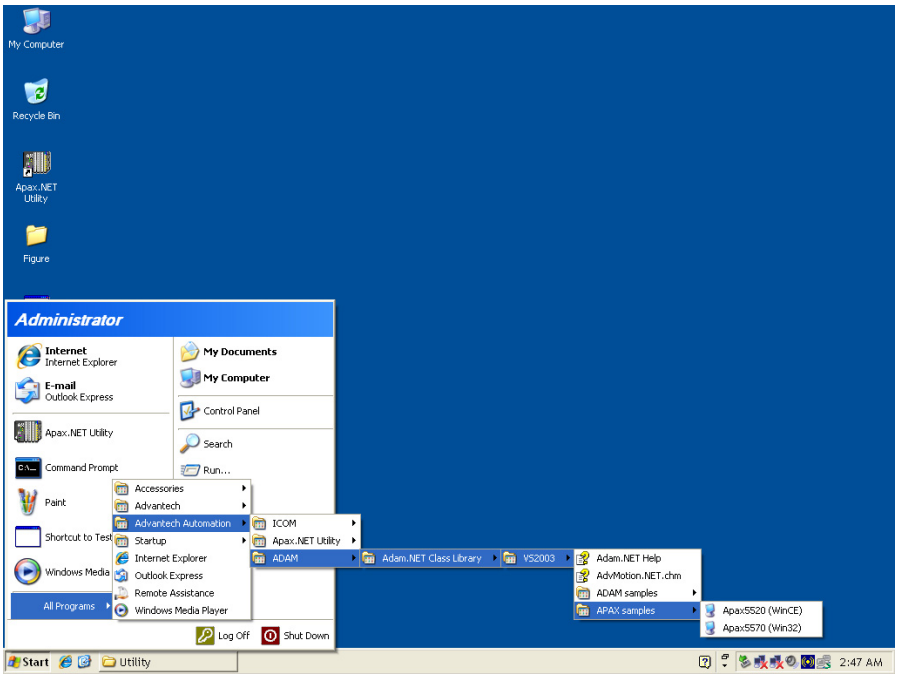
Backup functionality needs to be enabled for both the two APAX-5520 modules, in the APAX utility. Refer to figure below. Click the **Backup Setting** check box in **Setting Panel** Area and then click the **Apply** button to enable backup functionality for APAX-5520.



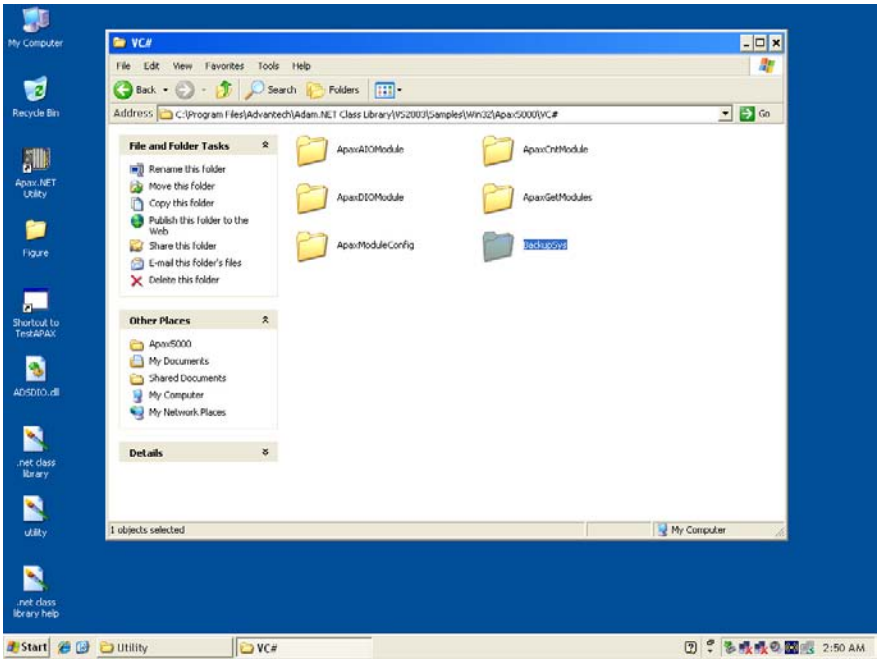
Note! *After applying the configuration for the backup system, remember to power cycle the whole system to run the backup functionality.*

B.3 Programming in Visual Studio .NET

After you enable backup functionality by utility, you can leverage the backup functionality in your application written in Microsoft Visual Studio .NET program. Related libraries are provided with Advantech Adam .NET class libraries. After you have installed the Advantech Adam .NET class libraries (Refer to Section 2.2 for the installation procedure), you can find related example programs by selecting Start >> All Programs >> Advantech Automation >> ADAM >> Adam.NET Class Library >> VS2003 >> APAX samples >> Apax5520 (WinCE).



Double click the VC# folder, you can find related example code in BackupSys folder.



ADVANTECH

eAutomation

www.advantech.com