USING YOUR NI-DAQ 4.9.0 SOFTWARE WITH YOUR VXI-DATA ACQUISITION MODULE

Introduction

This note describes your options when using NI-DAQ version 4.9.0 with your VXI-data acquisition (VXI-DAQ) module.

Note: ***The information in these release notes applies only to configurations using National Instruments VXIbus controllers.***

NI-DAQ 4.9.0 supports the following VXI-DAQ products:

- VXI-DIO-128
- VXI-MIO-64XE-10
- VXI-MIO-64E-1

on the following operating systems:

- Windows 3.1/3.11
- Windows 95

At this time, neither your printed nor online NI-DAQ 4.9.0 documentation describes the configuration and use of VXI-DAQ products. This document is your primary source of software information.

July 1996

Software Installation

A typical installation includes these National Instruments driver software products:

- NI-VXI
- NI-VISA
- NI-DAQ
- VXI*plug&play* Instrument Driver and Soft Front Panels

and one or more of these National Instruments application software packages and documentation:

- LabVIEW
- LabWindows®/CVI
- ComponentWorks
- VirtualBench

Perform the following steps to install your software:

1. Install NI-VXI on your computer. See your NI-VXI documentation for specific installation instructions. The necessary version number will vary with your configuration; Table 1 lists the possible controller and operating system combinations and the oldest version of NI-VXI you can use to create VXI-DAQ applications.

Table 1. NI-VXI Version Options

Controller	Win 3.1	Win 95
VXIpc-850	1.1	1.1
VXIpc-740	1.1	1.1
PCI-MXI-2	1.1	1.0
AT-MXI-2	1.0	1.0
AT-MXI-1	3.3	3.3
VXIpc-486, Model 500 Series	1.3	1.4

2. Verify that you already have NI-VISA 1.0 or later installed on your computer:

   a. Look for the VISA Interactive Control program in your **VXIPnP** folder.

      – If you find the program, run it and open the **About...** dialog box to determine the NI-VISA version number.

   b. If you don't find the program, look in your `\WINDOWS\SYSTEM` directory for `VISA.DLL` or `VISA32.DLL`.

      – If you find one of these files and the file has a 1996 or newer date, you have NI-VISA 1.0 or later.

   If you don't have NI-VISA 1.0 or later installed, install it at this time. Refer to your NI-VISA documentation for specific instructions. Contact National Instruments if you need the latest version of NI-VISA.

3. Install your application software using the appropriate documentation.

4. Install NI-DAQ version 4.9.0 on your computer, if it's not already installed. Use the diskettes that were included with your VXI-DAQ hardware and your NI-DAQ release notes to install it.

   If NI-DAQ is already installed, run the NI-DAQ Configuration Utility (formerly `WDAQCONF`) and check the title bar for the version number. If you have an earlier version than 4.9.0, replace it with version 4.9.0. See your NI-DAQ release notes for installation instructions.

5. Install your VXI*plug&play* Instrument Driver. Use the instructions in the Setup Utility on the diskettes that were included with your VXI-DAQ hardware.

Your software installation is now complete.

Configuration

After you have installed all your software, run the following configuration programs to configure your VXIbus system with NI-DAQ:

- Choose the icon and run `VXIINIT` to initialize your VXIbus controller. `VXIINIT` is not an interactive utility. To initialize your VXIbus controller, you must run it each time your computer reboots. You should place `VXIINIT` in your startup folder.
- Choose the icon and run the Resource Manager, `RESMAN`, to initialize the VXIbus modules in your chassis. `RESMAN` is also not an

interactive utility. You must run it every time your chassis is reset or turned on. `RESMAN` assigns resources to the VXIbus modules in your chassis. Run `RESMAN` after you run `VXIINIT`.

- Choose the icon and run `VXIEDIT` to configure your VXIbus controller. Follow the appropriate steps in the following sections to configure your controller to use VXI-DAQ products.
- Choose the icon and run the NI-DAQ Configuration Utility to configure your VXI-DAQ modules. More detailed information is in the *Configuring Your VXI-DAQ Module with NI-DAQ* section later in these release notes.

Configuring VXIpc-850 and VXIpc-740 Embedded Controllers

1. Launch `VXIEDIT`.
2. Choose the **VXIpc Configuration Editor**.
3. Select **Load Configuration From File**.
4. If you have a VXIpc-850 embedded controller, find `\NIVXI\TBL\VDAQ850.CFG` and select it.

   If you have a VXIpc-740 embedded controller, find `\NIVXI\TBL\VDAQ740.CFG` and select it.
5. Click on **Load**.
6. You should see a message that your **Configuration Restored Successfully**.
7. Choose the **Logical Address Configuration Editor**.
8. Set the **VXI Shared RAM Size** to **Share All of System Memory** and click **OK**.
9. Select the **Bus Configuration Editor**.
10. Depending on the number of VXI-DIO-128 and VXI-MIO modules in your system, choose a value for **User Window Size** from Table 2 and select that value in `VXIEDIT`. Click **OK**.

Table 2. User Window Size Options

Number of VXI-DIO-128 Modules	Number of VXI-MIO Modules	User Window Size Values
1 or more	0	64 KB
1 or more	1	128 KB
1 or more	2 or more	256 KB
0	1	64 KB
0	2	128 KB
0	3 or more	256 KB

11. Select **Update Current Configuration**. After updating the EEPROM, return to the main menu.
12. Quit `VXIEDIT` and reboot your computer.

Configuring PCI-MXI-2 Controllers

1. Launch `VXIEDIT`.
2. Choose the **PCI-MXI-2 Configuration Editor**.
3. Select **Load Configuration From File**.
4. Select `\NIVXI\TBL\VXIDAQ.CFG`.
5. Click on **Load**.
6. You should see a message that your **Configuration Restored Successfully**.
7. Choose the **Logical Address Configuration Editor**.
8. Set the **VXI Shared RAM Size** to **Share All of System Memory** and click **OK**.
9. Select the **Bus Configuration Editor**.
10. Depending on the number of VXI-DIO-128 and VXI-MIO modules you have, choose a value in the User Window Size column from Table 2 and select that value in the Window Size control in the PCI»User Window»Window Size section of `VXIEDIT`. Click **OK**.
11. Select **Update Current Configuration**. After updating the EEPROM, return to the main menu.
12. Select the VXI-MXI-2 Configuration Editor for each VXI-MXI-2 in your system by selecting a particular VXI-MXI-2 and clicking **OK**.

13. Set the **Interlocked** field to **Enable** and click **OK**. Repeat this step for each VXI-MXI-2 in your system.
14. Quit `VXIEDIT` and reboot your computer. Then turn your VXIbus chassis power off, then on, and run `VXIINIT` and `RESMAN`.

Configuring VXIpc-486 Model 500 Series Embedded Controllers

1. Launch `VXIEDIT`.
2. Choose the **Configuration Editor**.
3. Select the **A16/A32** address space.
4. Set **VXI Shared RAM (Byte Order)** to **MOT INT** (Motorola and Intel byte order) and click **OK**.
5. Set **VXI Shared RAM (MB)** to **All, 0**.
6. Press the **Save** button.
7. Quit `VXIEDIT` and reboot your computer.

Configuring AT-MXI-2 and AT-MXI-1 Controllers

No additional configuration steps are necessary for these controllers.

Configuring the GPIB-VXI/C Controller

This release of NI-DAQ does not support the use of VXI-DAQ modules with the GPIB-VXI/C controller.

Configuring Your VXI-DAQ Module with NI-DAQ

The NI-DAQ Configuration Utility identifies your VXI-DAQ module and configures its various operating settings. Among these settings are:

- VXIbus logical address

  Your VXIbus logical address is determined by the DIP switch on your VXI-DAQ module. `RESMAN` reports your logical address as part of its output.
- VXIbus interrupt level (VXI-MIO modules only)
- Either **A24** or **A32** address space (VXI-MIO modules only)

  `RESMAN` reports the address space assigned to your module in the **Configuring Address Map** portion of its output. The address space assigned is the address space in which your VXI-DAQ module requests memory.
- Amount of onboard RAM (VXI-MIO modules only)

In Windows 95, you must go through the **Add New Hardware** procedure before your VXI-DAQ module will be visible to the NI-DAQ Configuration Utility. Windows 3.*x* has no such requirement.

Select the **A32** address space for your VXI-MIO modules unless you are using a VXIpc-486 Model 500 Series controller in Windows 95; for the VXIpc-486 Model 500 Series controller, select the **A24** address space.

Your VXI-DAQ configuration is complete.

Programming Your VXI-DAQ Module

Current releases of your National Instruments software documentation do not specify VXI-DAQ hardware VIs and functions. However, most of the VIs and functions you need are already listed in the documentation for the DAQ hardware that is functionally equivalent to your VXI-DAQ module.

VXI-MIO Series Modules

The VXI-MIO-64XE-10 is functionally equivalent to the AT-MIO-16XE-10. Any VI or function valid for the AT-MIO-16XE-10 is also valid for the VXI-MIO-64XE-10. The same relationship exists between the VXI-MIO-64E-1 and the AT-MIO-16E-1. The channel numbering scheme for both VXI-MIO Series modules is identical to that of the AT-MIO-64E-3. The few differences that do exist, such as onboard memory, are discussed in the *Common Questions* section of this document.

VXI-DIO-128 Module

The VXI-DIO-128 currently has no functionally equivalent DAQ hardware. The following sections list the LabWindows/CVI functions and LabVIEW VIs you will use to program your VXI-DIO-128.

☞ **Note:** ***You cannot change the direction of the digital lines on the VXI-DIO-128. Sixty-four of the lines are always input and 64 are always output.***

LabWindows/CVI Functions

Use the following functions to program the VXI-DIO-128 module.

DIG_Prt_Config

- **Ports** 0–7 are always input ports and **ports** 8–15 are always output ports.
- **latchMode** is not supported and must be set to 0.
- You must set **Direction** to **input** for ports 0–7 and to **output** for ports 8–15.

DIG_In_Port

- When the software reads output ports, the returned data is simply the last data written, and the returned status is the **badChanDirError** warning.

DIG_Out_Port

- **Ports** must be 8–15.

DIG_Line_Config

- Lines in ports 0–7 are always input lines and lines in ports 8–15 are always output lines.
- There are eight lines per port numbered 0–7.
- You must set **Direction** to **input** for lines in ports 0–7 and to **output** for lines in ports 8–15.

DIG_In_Line

- When the software reads output lines, the returned data is simply the last data written, and the returned status is the **badDirOnSomeLinesError** warning.

DIG_Out_Line

- **Ports** must be 8–15.

Set_DAQ_Device_Info

The VXI-DIO-128 supports an adjustable threshold for the input ports. The logical 0 to logical 1 transition can occur anywhere in the range of -32.00 to +31.75 V in 250 mV steps. Set this threshold with the `Set_DAQ_Device_Info` function as shown in the following example, which sets the threshold of input port 3 to TTL level (1.5 V).

```
status = Set_DAQ_Device_Info(device,
ND_DIO128_SELECT_INPUT_PORT, 3);
status = Set_DAQ_Device_Info(device,
ND_DIO128_SET_PORT_THRESHOLD, 1500);
```

The first `Set_DAQ_Device_Info` call selects which input port threshold to modify. The definitions for `ND_DIO128_SELECT_INPUT_PORT` and `ND_DIO128_SET_PORT_THRESHOLD` are in the `NIDAQCNS include` file. The value for `ND_DIO128_SET_PORT_THRESHOLD` is in millivolts. The input port thresholds of input ports 0–7 are automatically initialized to the TTL level (1.5 V) whenever the NI-DAQ driver is loaded. Any changes made will endure only for the life of the program.

Get_DAQ_Device_Info

In addition to its other functions, you can use `Get_DAQ_Device_Info` to obtain the current input port threshold as shown in the following example:

```
err = Set_DAQ_Device_Info (device,
ND_DIO128_SELECT_INPUT_PORT, 3);
err = Get_DAQ_Device_Info (device,
ND_DIO128_GET_PORT_THRESHOLD, &port3threshold);
```

The value returned will be in millivolts.

LabVIEW VIs

Use the following LabVIEW VIs—DIO Port Config, DIO Port Read, and DIO Port Write—to program the VXI-DIO-128 module.

DIO Port Config

- **Ports** 0–7 are input ports and **ports** 8–15 are output ports.
- Every line within a port must have the same direction (for instance, be all input or all output).
- The physical port width is 8 bits. You can combine up to four consecutive ports into a 32-bit port.

DIO Port Read

- When the software reads the output lines, the returned data is simply the last data written, and the returned status is the **badDirOnSomeLinesError** warning.

DIO Port Write

You can write only to ports 8–15.

Setting the Input Threshold

The only way to change the input logical 0 to logical 1 threshold is by calling the `Set_DAQ_Device_Info` C language function. Use the LabVIEW **Call Library Node** to call this function. Use the nidaq32.dll for Windows 95, and the nidaq.dll for Windows 3.*x*. See the *Set_DAQ_Device_Info* section earlier in this document for more information on setting the input threshold.

Common Questions

1. **When I attempt to save my configuration in the NI-DAQ Configuration Utility, I receive a dialog box that says** *The device is not responding to the selected IRQ levels*. **What does this mean?**

   If you followed the steps in the *Installation* and *Configuration* sections of these release notes and you get this message, most likely it means that NI-DAQ does not have kernel mode access to your module. If you disable **Auto Test** (under the **Options** menu in the **Main** window) or set `VXI-IRQ` to **Disabled**, you should be able to save with no errors.

   Additionally, it may mean that you have forgotten to run `VXIINIT` or `RESMAN`. In Windows 3.1, the NI-DAQ Configuration Utility tells you that your device is not responding to the selected base address when you forget to run `VXIINIT` or `RESMAN`. In Windows 95, the message says that your device is not responding to the selected IRQ level(s).

2. **How do I know my VXI-DAQ module is installed and working?**

   After following the installation and configuration steps, run the NI-DAQ Configuration Utility. After opening the program, click on the entry for your VXI-DAQ module. When the second window appears, click on the **Test** menu and choose one of the available tests. The configuration test verifies communication with the VXIbus module and, if you have kernel mode access, confirms whether

VXIbus interrupts and DMA are working. The other tests are interactive and test the A/D converter, the D/A converter, the digital I/O lines, and the counter/timers. Additionally, if you installed the VXI*plug&play* instrument driver, the Soft Front Panels included with the instrument driver are also useful in verifying that your module is working.

3. **What is meant by** *kernel mode access* **and what do I need to know about it?**

   NI-DAQ handles all interrupts in kernel mode. However, certain resource limitations can prevent NI-DAQ from communicating with your VXI-DAQ module in kernel mode. When NI-DAQ is denied kernel mode access, it is unable to perform some buffered operations, such as waveform capture. It can still perform immediate operations, such as `AI_VRead` in LabWindows/CVI, or AI Single Scan and timed, nonbuffered operations in LabVIEW. The VXI-DIO-128 does not require kernel mode access and is completely functional without it.

4. **What are the resource limitations that deny NI-DAQ kernel mode access?**

   Each VXIbus controller has a limited number of *windows* into the VXIbus address space where your VXI-DAQ module resides.

   - Each of the VXIpc-850, VXIpc-740, and PCI-MXI-2 controllers has four of these windows. You have kernel mode access on up to four modules simultaneously. Using any A16 devices will use up one of your four windows. The actual number of VXI-DAQ modules to which you have simultaneous kernel mode access will vary on the VXIpc-486 Model 500 Series embedded controller.
   - On the AT-MXI-2 and AT-MXI-1 controllers, there are no windows available for kernel mode access.

   If you attempt an operation on a VXI-DAQ module that requires NI-DAQ to communicate with your module at interrupt time, and NI-DAQ does not have kernel mode access to the module, your application will receive the **deviceSupportError**. See *Question 5* for more information.

5. Is there anything I can do about a kernel mode access problem?

Yes. If you have simply run out of windows, you can close the window on one VXI-DAQ module and reopen it on another. You can also use onboard memory.

Use this pair of functions to open and close a window:

```
short _stdcall forceVXIDAQWindowOpen (short
deviceNumber);

short _stdcall forceVXIDAQWindowClosed (short
deviceNumber);
```

Your board must be idle when you close its VISA window. There are no LabVIEW VIs for these calls so you must use the **Call Library Node** if you are using LabVIEW. Use the nidaq32.dll for Windows 95, and the nidaq.dll for Windows 3.*x*. If you are using LabWindows/CVI, follow this example:

```
short _stdcall forceVXIDAQWindowOpen (short
deviceNumber);

short _stdcall forceVXIDAQWindowClosed (short
deviceNumber);

void main ()

{

   short err;

   err = forceVXIDAQWindowClosed (5);

   if (err == noError)

          printf("\nVISA Window successfully
          closed for device number 5.");

   err = forceVXIDAQWindowOpen (6);

   if (err == noError)

          err = Init_DA_Brds (6, &boardCode);

   if (err == noError)

          printf("\nVISA Window successfully
          opened for device number 6.");

}
```

Another way to solve a kernel mode access problem is to use your onboard memory. Timed, buffered analog input (waveform capture) without kernel mode access is possible only if you have onboard memory.

Timed, buffered analog output (waveform generation) is possible even when kernel mode access is denied. However, the iteration count will be unavailable and the regeneration modes will not be supported. You can use either host or onboard memory for your waveform generation, even when NI-DAQ does not have kernel mode access.

Timed, buffered counter/timer input is not available without kernel mode access.

Only one operation at a time can use onboard memory. That is, you can acquire analog input data into the onboard memory, but you cannot simultaneously generate a waveform from data in that onboard memory. Onboard memory on the VXI-MIO Series modules is only supported in LabVIEW. See *Question 6* for more information about onboard memory.

6. How do I specify the use of onboard memory in my program?

First, you must tell NI-DAQ via the NI-DAQ Configuration Utility that you have installed onboard memory. See the *Installation* chapter of your VXI-DAQ user manual for more information. The onboard memory is on the VXI-DAQ module itself, and not on the VXIbus controller.

Select the amount of onboard memory you intend to use via the **Onboard Memory Size** dialog box in the NI-DAQ Configuration Utility. Select **A32** address space for your module if your controller allows it, also in the NI-DAQ Configuration Utility. After making these changes in the NI-DAQ Configuration Utility, reset or turn your VXIbus chassis off, then on, and run `RESMAN`.

Next, set your LabVIEW application to use onboard memory at run time. For analog input, set the **allocation mode** control in the **AI Buffer Config VI** to **3**; allocate **DSP Memory**. For analog output, set the **allocate mode** control in **AO Buffer Config VI** to **4**; allocate **DSP Memory**. Ignore the text referring to DSP Memory and the AT-DSP2200 module.

A constraint is placed on analog input operations that use onboard memory when kernel mode access is denied—you cannot acquire more data than the buffer that you allocated in the onboard memory can hold. For example, if your buffer is 1,000 scans in size, you can acquire no more than 1,000 scans. Continuous acquisition is not allowed.

7. **What exactly does the VXI-DAQ configuration file (`VXIDAQ.CFG`, `VDAQ850.CFG`, `VDAQ740.CFG`) do to my `VXIEDIT` configuration?**

   For all controllers, the VXI-DAQ configuration file sets your controller address space to **A32**, sets the **Byte Order** to **Non-Swapped**, and sets **Slave Write Posting** to **Enable**. The PCI-MXI-2 VXI-DAQ configuration file will also point retries caused by CPU-MXI collisions towards the MXIbus, and set **MXI Auto Retry** to **disable**. The VXIpc700 Series VXI-DAQ configuration file will point retries caused by CPU-VXI collisions toward the VXIbus and set **VXI Auto Retry** to **Disable**.

8. **How do I use SCXI with my VXI-MIO module?**

   Simply follow the procedures outlined in your NI-DAQ documentation for MIO E Series devices. Using SCXI with your VXI-MIO module is no different than using SCXI with any other MIO E Series device.

9. **Are there any module features described in my VXI-DAQ user manuals that are not supported in the NI-DAQ 4.9.0 release?**

   Yes. For the VXI-MIO Series modules, NI-DAQ 4.9.0 does *not* support driving signals onto, or receiving signals from, the eight TTL VXIbus trigger lines and the two ECL VXIbus trigger lines.

   For the VXI-DIO-128 module, NI-DAQ 4.9.0 does *not* support Serial Number EPROM or Temperature Sensor features.

10. **When I attempt to use the regeneration modes for waveform generation, why do I receive the** *-10004 (valueConflictError)* **error?**

    Waveform regeneration modes (**regeneration mode** values 2 and 3 in the AO Buffer Write VI in LabVIEW and **oldDataStop** value 1 and **partialTransferStop** value 1 in `WFM_DB_Config`) are supported on the VXI-MIO modules only when you use interrupts to send the waveform data to the module. Since the default transfer mode for waveform generation uses DMA, use the Set DAQ Device Information VI in LabVIEW or the `Set_DAQ_Device_Info` function in the C language interface to switch to using interrupts. Also notice that if kernel mode access is denied, the regeneration modes will never be available, since you can only use DMA in this case.

11. **I had just finished running a LabVIEW data acquisition VI that used my VXI-DAQ module when I launched the NI-DAQ Configuration Utility. The Configuration test on my VXI-DAQ module informed me that the module was busy and that any tests would interrupt the current data acquisition, even though the LabVIEW VI had finished. Then the Configuration test informed me that the module was not responding to the base address, even though the Configuration test had passed on previous occasions. The other tests in the NI-DAQ Configuration Utility return the** *-10403 (deviceSupportError)* **error, even though these tests have worked on previous occasions.**

    This behavior occurs when a second application attempts to use NI-DAQ to communicate with a VXI-DAQ module in Windows 95 after another application has already done so and is still open. NI-DAQ version 4.9.0 does not allow two applications to perform operations on VXI-DAQ modules at the same time in Windows 95. You will have to close the application that ran first (LabVIEW in the question above), before launching the second application. This behavior does not occur in Windows 3.1.

321252A-01
July 1996