I^2C/SPI

NI-845*x* Software User Manual

October 2005
371746A-01

NATIONAL INSTRUMENTS™

Important Information

Compliance

Compliance with FCC/Canada Radio Frequency Interference Regulations

Determining FCC Class

The Federal Communications Commission (FCC) has rules to protect wireless communications from interference. The FCC places digital electronics into two classes. These classes are known as Class A (for use in industrial-commercial locations only) or Class B (for use in residential or commercial locations). All National Instruments (NI) products are FCC Class A products.

Depending on where it is operated, this Class A product could be subject to restrictions in the FCC rules. (In Canada, the Department of Communications (DOC), of Industry Canada, regulates wireless interference in much the same way.) Digital electronics emit weak signals during normal operation that can affect radio, television, or other wireless products.

All Class A products display a simple warning statement of one paragraph in length regarding interference and undesired operation. The FCC rules have restrictions regarding the locations where FCC Class A products can be operated.

Consult the FCC Web site at `www.fcc.gov` for more information.

FCC/DOC Warnings

This equipment generates and uses radio frequency energy and, if not installed and used in strict accordance with the instructions in this manual and the CE marking Declaration of Conformity*, may cause interference to radio and television reception. Classification requirements are the same for the Federal Communications Commission (FCC) and the Canadian Department of Communications (DOC).

Changes or modifications not expressly approved by NI could void the user's authority to operate the equipment under the FCC Rules.

Class A

Federal Communications Commission

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user is required to correct the interference at their own expense.

Canadian Department of Communications

This Class A digital apparatus meets all requirements of the Canadian Interference-Causing Equipment Regulations.

Cet appareil numérique de la classe A respecte toutes les exigences du Règlement sur le matériel brouilleur du Canada.

Compliance with EU Directives

Users in the European Union (EU) should refer to the Declaration of Conformity (DoC) for information* pertaining to the CE marking. Refer to the Declaration of Conformity (DoC) for this product for any additional regulatory compliance information. To obtain the DoC for this product, visit `ni.com/certification`, search by model number or product line, and click the appropriate link in the Certification column.

* The CE marking Declaration of Conformity contains important supplementary information and instructions for the user or installer.

Contents

Chapter 8
Using the NI-845*x* DIO API

Chapter 9
NI-845*x* DIO API for LabVIEW

Appendix A
Technical Support and Professional Services

Glossary

Index

About This Manual

This manual explains how to use the NI-845*x* software. It contains installation and configuration information and function reference for a LabVIEW-based API.

Use this manual to learn the basics of I^{2}C and SPI communication with NI-845*x*, as well as how to develop an application.

Conventions

The following conventions appear in this manual:

»	The » symbol leads you through nested menu items and dialog box options to a final action. The sequence **File»Page Setup»Options** directs you to pull down the **File** menu, select the **Page Setup** item, and select **Options** from the last dialog box.
✎	This icon denotes a note, which alerts you to important information.
bold	Bold text denotes items that you must select or click in the software, such as menu items and dialog box options. Bold text also denotes parameter names.
italic	Italic text denotes variables, emphasis, a cross reference, or an introduction to a key concept. Italic text also denotes text that is a placeholder for a word or value that you must supply.
`monospace`	Text in this font denotes text or characters that you should enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames, and extensions.

Related Documentation

The following documents contain information that you might find helpful as you read this manual:

- *USB-8451 User Guide and Specifications*

1 Introduction

This chapter introduces the Inter-IC (I^2C) and Serial Peripheral Interface (SPI) buses.

I^2C Bus

Philips Semiconductors developed the I^2C bus in the early 1980s to connect a CPU to peripheral chips in televisions. I^2C is also used to communicate with temperature sensors, EEPROMs, LCD displays, and other embedded peripheral devices.

I^2C Terminology

This manual uses the following I^2C bus terms:

I^2C	Inter-IC.
SMBus	System Management Bus.
Transmitter	Device transmitting data on the bus.
Receiver	Device receiving data from the bus.
Master	Device that can initiate and terminate a transfer on the bus. The master is responsible for generating the clock (SCL) signal.
Slave	Device addressed by the master.
Multimaster	The ability for more than one master to co-exist on the bus concurrently without data loss.
Arbitration	The procedure to allow multiple masters to determine which single master controls the bus for a particular transfer time.

Synchronization	The defined procedure to allow the clock signals provided by two or more masters to be synchronized.
SDA	Serial DAta (data signal line).
SCL	Serial CLock (clock signal line).

I^{2}C Bus

The I^{2}C bus is a two-wire half-duplex serial interface. The two wires, SDA and SCL, are both bidirectional. The I^{2}C specification defines three speeds, standard at 100 kHz, fast at 400 kHz, and high speed at 3.4 MHz.

Each device connected to the I^{2}C bus has a unique 7-bit I^{2}C address to facilitate identification and communication by the master. Typically, the upper four bits are fixed and assigned to specific categories of devices (for example, 1010 is assigned to serial EEPROMs). The three lower bits are programmable through hardware address pins, allowing up to eight devices of the same type to be connected to a single I^{2}C bus.

Each device on the bus (both master and slave) can be a receiver and/or transmitter. For example, an LCD is typically only a receiver, while an EEPROM is both a transmitter and receiver.

The I^{2}C is a multimaster bus, meaning that multiple masters can be connected to the bus at the same time. While a master is initiating a transfer on the bus, all other devices, including other masters, are acting like slaves. However, if another master is trying to control the bus at the same time, I^{2}C defines an arbitration mechanism to determine which master gets control of the bus.

I^{2}C Arbitration

When two masters are trying to control the bus simultaneously, or if a second master joins the bus in the middle of a transfer and wants to control the bus, the I^{2}C bus has an arbitration scheme to guarantee no data corruption.

With I^{2}C, a line (both SDA and SCL) is either driven low or allowed to be pulled high. When a master changes a line state to high, it must sample the line afterwards to make sure it really has been pulled high. If the master samples the SDA bus after setting it high, and the sample shows that the line is low, it knows another master is driving it low. The master assumes it has lost arbitration and waits until it detects a stop condition before making another attempt to start transmitting.

I²C Transfers

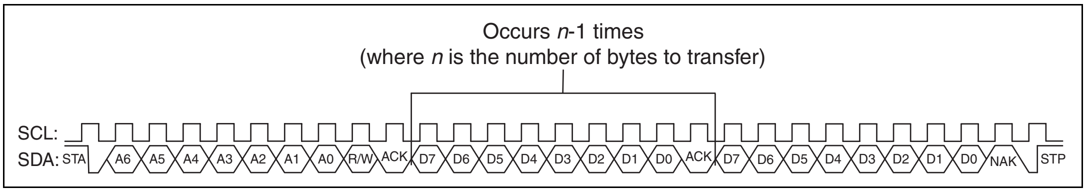


Figure 1-1. I²C Transfers

To initiate a transfer, the master issues a start condition by changing the SDA line level from high to low while keeping the SCL clock line high. When this occurs, the bus is considered busy, and all devices on the bus get ready to listen for incoming data.

Next, the master sends the 7-bit address and 1-bit data transfer direction on the bus to configure for the appropriate data transfer. All slaves compare the address with their own address. If the address matches, the slave produces an acknowledge signal.

If the master detects an acknowledge signal, it starts transmitting or receiving data. To transmit data to a device, the master places the first bit onto the SDA line and generates a clock pulse to transmit the bit across the bus to the slave. To receive data from a device, the master releases the SDA line, allowing the slave to take control of it. The master generates a clock pulse on the SCL line for each bit, reading the data while the SCL line is high. The device is not allowed to change the SDA line state while the SCL line is high.

After the data transmission, the master issues the stop condition by changing the SDA line from low to high while keeping the SCL clock line high. When this occurs, the bus is considered free again for another master to initiate a data transfer.

I²C Clock Stretching

Because the master controls the clock, the I²C specification provides a mechanism to allow the slave to slow down the bus traffic when it is not ready. This mechanism is known as clock stretching. During any SCL low phase, a slave may additionally hold down SCL to prevent it from rising high again to slow down the SCL clock rate or pause I²C communication.

When the master attempts to make SCL high to complete the current clock pulse, it must verify that it has really gone high. If it is still low, it knows a slave is holding it low and must wait until it goes high before continuing.

I^{2}C Extended (10-Bit) Addressing

Typical I^{2}C devices use a 7-bit addressing scheme. I^{2}C also defines a 10-bit addressing scheme that allows up to 1024 additional addresses to be connected to the I^{2}C bus. This 10-bit addressing scheme does not affect the existing 7-bit addressing, allowing both 7-bit and 10-bit addressed devices to share the bus. A device that supports 10-bit addressing receives the address across two bytes. The first byte consists of the Philips-designated 10-bit slave addressing mode code (11110), the 2 MSBs of the device address, and the Read/Write bit. The next data byte sent across the bus contains the eight LSBs of the address.

I^{2}C vs. SMBus

Intel defined the System Management Bus (SMBus) in 1995. This bus is used primarily in personal computers and servers for low-speed system management communications.

The I^{2}C bus and SMBus are very similar; at frequencies at or below 100 kHz, they tend to be interchangeable. However, the following sections describe some important differences.

Timeout and Clock Rates

I^{2}C has no minimum clock rate, and as such there is no minimum clock frequency duration. However, SMBus does not allow the clock to be slower than 10 kHz; a device will reset if the clock remains low for more than 35 ms.

I^{2}C allows clock rates of 100 kHz, 400 kHz, and 3.4 MHz, whereas SMBus is limited to a maximum clock rate of 100 kHz.

Logic Levels

Logic high is defined on I^{2}C as 3.0 V or 0.7 * V_{DD}. On SMBus, logic high is defined as 2.1 V.

Logic low is defined on I^{2}C as 1.5 V or 0.3 * V_{DD}. On SMBus, logic low is defined as 0.8 V.

Current Levels

The sink current also varies between I^{2}C and SMBus. In I^{2}C the maximum is 3 mA, whereas SMBus has a maximum of 350 µA. This determines the lowest acceptable value of the pull-up resistor. At 3 V, an I^{2}C bus should have a pull-up of > 1 kΩ; SMBus should have a pull-up of > 8.5 kΩ. However, many SMBus systems violate this rule; a common range for both SMBus and I^{2}C tends to be in the 2.4–3.9 kΩ range.

Throughout this document, we will refer to the bus as an I^{2}C bus. For information about compatibility of your NI 845*x* device with SMBus, refer to the hardware documentation for your device.

SPI Bus

The SPI bus is a standard established by Motorola and is used to communicate with devices such as EEPROMs, real-time clocks, converters (ADC and DAC), and sensors.

SPI Terminology

This manual uses the following SPI bus terms:

CLK	CLocK. The clock is generated by the master device and controls when data is sent and read.
MOSI	Master Output, Slave Input. The MOSI line carries data from the master to the slave.
MISO	Master Input, Slave Output. The MISO carries data from the slave to the master.
CS or SS	Chip Select or Slave Select. Connection from the master to a slave that signals the slave to listen for SPI clock and data signals.
CPOL	Clock POLarity. The polarity indicating whether the clock makes positive or negative pulses.
CPHA	Clock PHAse. This controls the positioning of the data bits relative to the clock edges.
Shift Register	A shift register is connected to the MOSI and MISO lines. As data is read from the input, it is placed into the shift register. Data from the shift register is

	placed into the output, creating a full-duplex communication loop.
Master	The master device provides the clock signal and determines the chip select line state.
Slave	The slave device receives the clock and chip select from the master. The maximum number of slaves is dependent on the number of available chip select lines.

SPI Bus

The SPI bus is a four-wire, full-duplex serial interface. Three of the wires, SCK, MOSI, and MISO, are shared along with a fourth wire, known as the chip select, which is a direction connection between the master and a single slave.

Communication across SPI uses a system known as data exchange. Whenever a bit is written to an SPI device across the MOSI lines, the SPI device concurrently returns a bit on the MISO line. Because data is transferred in both directions, it is up to the receiving device to know whether the received by is meaningful or not. For example, to receive data from an EEPROM, the master must configure the EEPROM to send *n* bytes of data and then must send *n* bytes to be exchanged for valid data. These bytes can usually be any value, and writing them serves only to clock the data out of the receiving device.

Clock and Polarity

Parameters called clock polarity (CPOL) and clock phase (CPHA) determine the clock idle state and the edge of the clock signal when the data is driven and sampled. These parameters are sometimes expressed as four modes, as shown in Table 1-1.

Table 1-1. SPI Modes

SPI Mode	Polarity	Phase
0	0	0
1	0	1
2	1	0
3	1	1

When the polarity is 0, the clock idles low. When the polarity is 1, the clock idles high. When the phase is 0, data is latched at the clock transition from idle to asserted. When the phase is 1, the data is latched at the clock transition from asserted to idle. Figure 1-2 shows how the four SPI modes affect the clock and sample times.

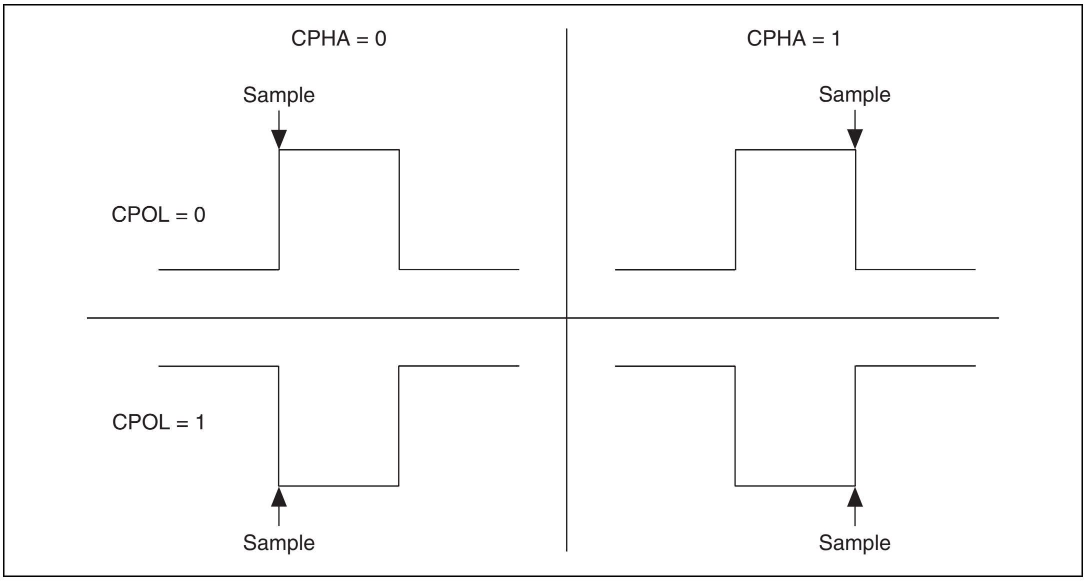


Figure 1-2. SPI Polarity Phase Differences

Error Handling

Unlike I^{2}C, SPI has no acknowledgement mechanism or flow control. This prevents the SPI master from knowing whether a slave received a data byte correctly or even whether it is connected to the bus.

2 Installation

This chapter explains how to install the NI-845*x* software and hardware.

Software Installation

This section discusses installing the NI-845*x* software on Microsoft Windows.

Note You need administrator privileges to install the NI-845*x* software on your computer.

1. Insert the *NI-845x Software* CD into your CD-ROM drive. The installer launches if your CD-ROM drive plays data CDs automatically. If the installer does not launch automatically, navigate to the CD using Windows Explorer and launch the `autorun.exe` file from your *NI-845x Software* CD.
2. The Installation Wizard guides you through the necessary steps to install the NI-845*x* software. You can go back and change values where appropriate by clicking the **Back** button. You can exit the setup where appropriate by clicking **Cancel**.
3. When installation is complete, select **Finish**.

Hardware Installation

Step 1: Unpack the Devices, Accessories, and Cables

Your device ships in an antistatic package to prevent electrostatic discharge (ESD) damage to the device. ESD can damage several components on the device.

To avoid such damage, take the following precautions:

- Ground yourself using a grounding strap or by touching a grounded object.
- Touch the antistatic package to a metal part of the computer chassis before removing the device from the package.

Remove the device from the package and inspect the device for loose components or any sign of damage. Notify National Instruments if the device appears damaged in any way. Do not install a damaged device into your computer or PXI chassis.

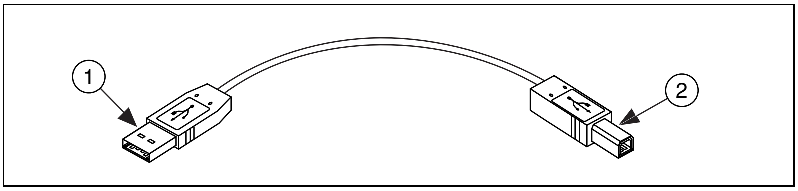
Store the device in the antistatic package when the device is not in use.

For safety and compliance information, refer to the device documentation packaged with your device.

Step 2: Install the Devices, Accessories, and Cables

Complete the following steps to install an NI USB device:

1. Connect the USB cable from the computer USB port or from any other hub that provides USB power to the USB port on the device. The following figure shows the USB cable and its connectors.



1 Host/Hub/PC USB Port	2 NI USB Device USB Port

2. Power on your computer or PXI chassis. On some Windows systems, the Found New Hardware wizard opens with a dialog box for every device installed. Click **Next** or **Yes** to install the software for each device.
3. Install accessories and/or terminal blocks according to the instructions in their user guides.

Step 3: Confirm that Your Device Is Recognized

To verify that the USB device is recognized, complete the following steps:

1. Double-click the **Measurement & Automation** icon on the desktop to open Measurement & Automation Explorer (MAX).
2. Expand **Devices and Interfaces**.
3. Verify that the device appears under **USB Devices**. If the device does not appear, press <F5> to refresh the view in MAX. If the device is still not recognized, refer to `ni.com.support/install` for troubleshooting information.

3 Using the NI-845*x* API

The NI-845*x* API consists of references, property nodes, and functions. A reference is a handle to an entity. For example, to access an NI 845*x* device, you first must create a device reference by providing the name of the NI 845*x* device configured in Measurement & Automation Explorer (MAX). After creating the device reference, the NI-845*x* software functions use the returned handle to determine which NI 845*x* device to communicate with.

The NI-845*x* API has other references also. An example is a configuration reference that describes the device characteristics used for communication. An I^2C configuration reference contains properties such as the bus clock rate and device address to use for communication. Refer to the specific API calls for more information on how references are used in the NI-845*x* API.

A reference can be passed into a property node to configure characteristics specific to the reference type. In addition, many of the API functions require a reference to know the object to perform an action on.

4

Using the NI-845*x* I²C API

This chapter helps you get started with the I²C API.

I²C Basic Programming Model

The I²C Basic API provides the most fundamental I²C transaction types: write, read, and write/read. You can access the majority of off-the-shelf I²C devices using these transactions. The I²C Basic API allows you to easily and quickly develop applications to communicate with these devices. For those situations in which the I²C Basic API does not provide the functionality you need, use the I²C Advanced API to create custom I²C transactions.

When you use the I²C Basic API, the first step is to create an I²C configuration to describe the communication requirements between the NI 845*x* and the I²C device. To make an I²C configuration, create an I²C configuration reference and set the appropriate properties as desired. You can then read or write data to the I²C device.

The diagram in Figure 4-1 describes the programming model for the NI-845*x* I²C Basic API. Within the application, you repeat this programming model for each I²C device. The diagram is followed by a description of each step in the model.

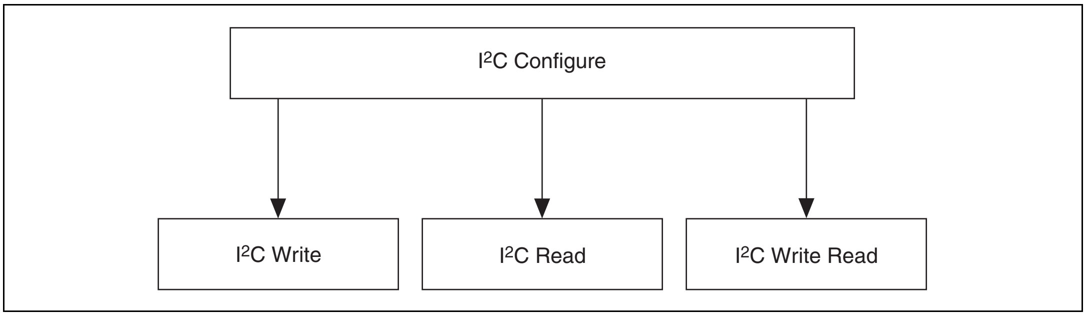


Figure 4-1. Basic Programming Model for I²C Communication

I^{2}C Configure

Use the **NI-845x I2C Configuration Property Node** to set the specific I^{2}C configuration that describes the characteristics of the device to communicate with.

I^{2}C Write

Use **NI-845x I2C Write.vi** to write an array of data to an I^{2}C slave device.

I^{2}C Read

Use **NI-845x I2C Read.vi** to read an array of data from an I^{2}C slave device.

I^{2}C Write Read

Use **NI-845x I2C Write Read.vi** to write an array of data followed by a read (combined format) on an I^{2}C slave device.

I^{2}C Advanced Programming Model

The Philips I^{2}C specification is extremely flexible and allows multiple possibilities for constructing transactions beyond those handled by the I^{2}C Basic API. The I^{2}C Advanced API provides a set of script commands that allow you great flexibility in creating custom I^{2}C transactions for your particular needs. For example, you can use scripting in the following scenarios:

- Validating a new device design, when you want to issue individual I^{2}C conditions to the bus, with or without variable delays in between, so that you can observe device response.
- Issuing a transaction to a device and measuring its responses (using NI 845*x* DIO pins configured for input) at multiple points within the transaction.
- Using the NI 845*x* DIO pins configured for output to provide additional control or addressing.
- Doing performance testing, in which you see how a device responds to variable delays, clock rate changes, etc. within a transaction.
- Issuing multiple reads and writes to a device, or multiple devices, within one transaction, to avoid relinquishing the bus.

When you use the I^{2}C Advanced API, the first step is to create a script that describes the communication between an I^{2}C master and an I^{2}C slave device. Then you execute the script and extract the resource data if needed.

The script size is limited only by the amount of memory available on your PC. The number of read commands, **NI-845x I2C Script Read.vi**, **NI-845x I2C Script DIO Read Port.vi**, and **NI-845x I2C Script DIO Read Line.vi** within each script is limited to 64.

The diagram in Figure 4-2 describes an example of programming with the scripting functions for the NI-845*x* I²C Advanced API. The diagram is followed by a description of each step in the model.

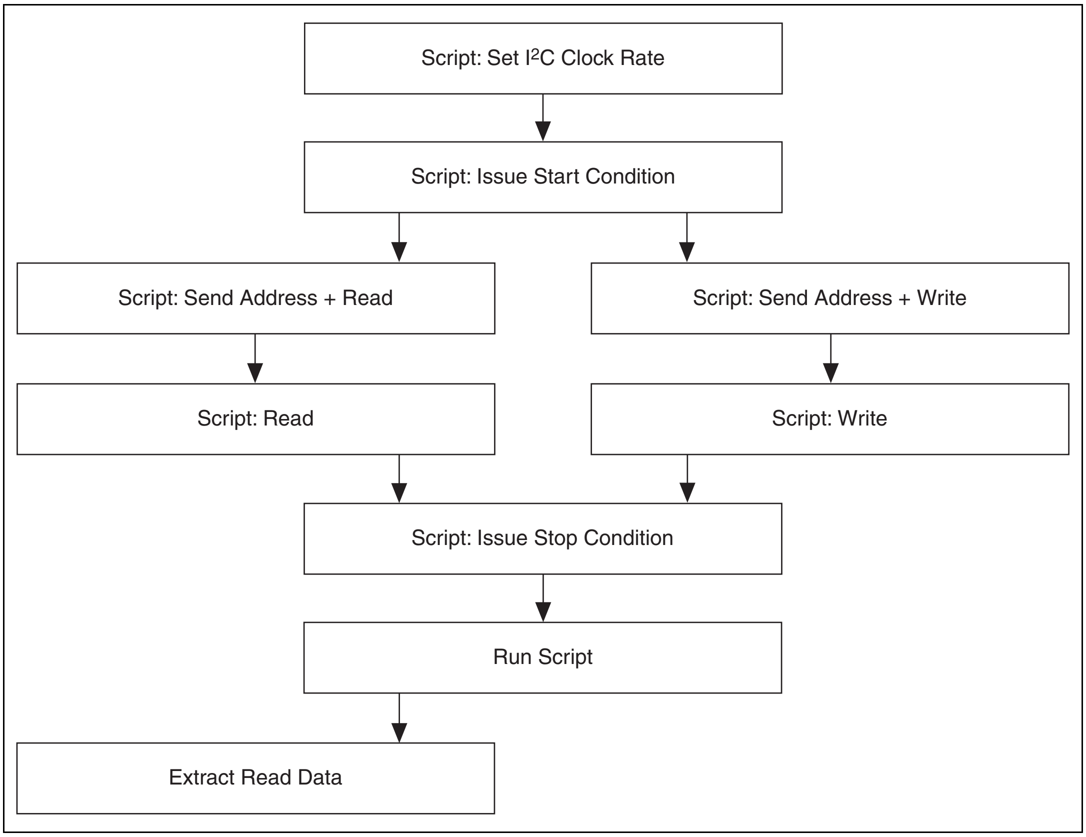

Script: Set I²C Clock Rate
Script: Issue Start Condition
Script: Send Address + Read
Script: Send Address + Write
Script: Read
Script: Write
Script: Issue Stop Condition
Run Script
Extract Read Data

Figure 4-2. Example of Advanced Programming Model with Scripting API for I²C Communication

Script: Set I²C Clock Rate

Use **NI-845x I2C Script Clock Rate.vi** to add an I²C Script Clock Rate command to an I²C script referenced by **i2c script reference in**. This command sets the I²C clock rate for the I²C port you specify when you use **NI-845x I2C Run Script.vi** to execute the script.

Script: Issue Start Condition

Use **NI-845x I2C Script Issue Start.vi** to add an I^{2}C Script Issue Start command to an I^{2}C script referenced by **i2c script reference in**. This command issues a start condition on the I^{2}C bus connected to the I^{2}C port you specify when you use **NI-845x I2C Run Script.vi** to execute the script.

Script: Send Address + Read

Use **NI-845x I2C Script Address+Read.vi** to add an I^{2}C Script Address+Read command to an I^{2}C script referenced by **i2c script reference in**. This command writes a 7-bit address, followed by the direction bit set to read, to the I^{2}C bus connected to the I^{2}C port you specify when you use **NI-845x I2C Run Script.vi** to execute the script.

Script: Read

Use **NI-845x I2C Script Read.vi** to add an I^{2}C Script Read command to an I^{2}C script referenced by **i2c script reference in**. This command reads an array of data from a device connected to the I^{2}C port you specify when you use **NI-845x I2C Run Script.vi** to execute the script.

Script: Send Address + Write

Use **NI-845x I2C Script Address+Write.vi** to add an I^{2}C Script Address+Write command to an I^{2}C script referenced by **i2c script reference in**. This command writes a 7-bit address, followed by the direction bit set to write, to the I^{2}C bus connected to the I^{2}C port you specify when you use **NI-845x I2C Run Script.vi** to execute the script.

Script: Write

Use **NI-845x I2C Script Write.vi** to write an array of data to an I^{2}C slave device.

Script: Issue Stop Condition

Use **NI-845x I2C Script Issue Stop.vi** to add an I^{2}C Script Issue Stop command to an I^{2}C script referenced by **i2c script reference in**. This command issues a stop condition on the I^{2}C bus connected to the I^{2}C port you specify when you use **NI-845x I2C Run Script.vi** to execute the script.

Run Script

Use **NI-845x I2C Run Script.vi** to execute an I^{2}C script referenced by **i2c script reference in** on the device referenced by **device reference in**.

Extract Read Data

Use **NI-845x I2C Extract Script Read Data.vi** to extract the desired read data from an I^{2}C script, referenced by **i2c script reference in**, which has been processed by **NI-845x I2C Run Script.vi**. Each I^{2}C script read command (**NI-845x I2C Script Read.vi**, **NI-845x I2C Script DIO Read Port.vi, NI-845x I2C Script DIO Read Line.vi**) returns a **script read index**. Data may be extracted for each **script read index** in a script, by wiring each to a separate **NI-845x I2C Extract Script Read Data.vi**.

5

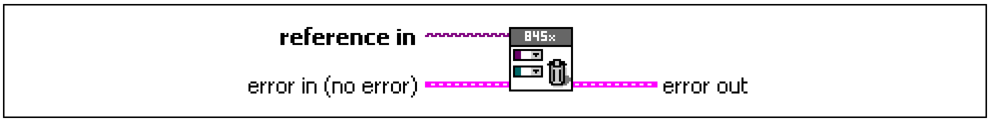
NI-845*x* I^{2}C API for LabVIEW

This chapter lists the LabVIEW VIs for the NI-845*x* I^{2}C API and describes the format, purpose, and parameters for each VI. The VIs in this chapter are listed alphabetically.

NI-845x Close Reference.vi

Purpose

Closes a previously opened reference.



Inputs

reference in is a reference to an NI 845*x* device, I²C configuration, SPI configuration, I²C script, or SPI script.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

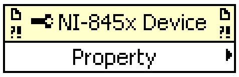
Description

Use **NI-845x Close Reference.vi** to close a previously opened reference.

NI-845x Device Property Node

Purpose

A property node with the NI-845*x* Device class preselected. This property node allows you to modify properties of your NI 845*x* device.



Inputs

device reference in is a reference to an NI 845*x* device.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

device reference out is a reference to an NI 845*x* device after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is

returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Description

The list below describes all valid properties for the **NI-845x Device Property Node**.

Active DIO Port

The **Active DIO Port** property sets the active DIO port for further DIO port configuration. The format for this property is a decimal string. For example, the string `0` represents DIO Port 0. For NI 845*x* devices with one DIO port, the port value must be set to `0`.

DIO Port Voltage

The **DIO Port Voltage** property configures the active DIO port with the desired voltage characteristics. **DIO Port Voltage** uses the following values:

`Open-Drain`

The port is configured for open-drain voltage.

`Push-Pull 3.3 V`

The port is configured for 3.3 V push-pull voltage.

The default value of this property is `Push-Pull 3.3 V`.

DIO Line Direction Map

The **DIO Line Direction Map** property sets the line direction map for the active DIO Port. The value is a bitmap that specifies the function of each individual line within the port. If bit *x* = 1, line *x* is an output. If bit *x* = 0, line *x* is an input.

The default value of this property is `0` (all lines configured for input).

NI-845x Device Reference

Purpose

Specifies the device resource to be used for communication.

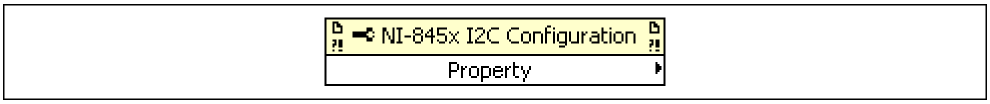
Description

Use the **NI-845x Device Reference** to describe the NI 845*x* device to communicate with. You can wire the reference into a property node to set specific device parameters or to an NI-845*x* API call to invoke the function on the associated NI 845*x* device.

NI-845x I2C Configuration Property Node

Purpose

A property node with the NI-845*x* I^{2}C Configuration class preselected. This property node allows you to query and modify I^{2}C configuration properties of your NI 845*x* device.


NI-845x I2C Configuration
Property

Inputs

i2c configuration in is a reference to a specific I^{2}C configuration that describes the characteristics of the device to communicate with.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

i2c configuration out is a reference to a specific I^{2}C configuration that describes the characteristics of the device to communicate with.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is

returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Description

The list below describes all valid properties for the **NI-845x I2C Configuration Property Node**.

Port

Specifies the I^{2}C port that this configuration communicates across.

Refer to your hardware user guide to determine the number of I^{2}C ports your NI 845*x* device supports.

The default value of this property is `0`.

Clock Rate in kHz

Specifies the I^{2}C clock rate. Refer to your hardware user guide to determine which clock rates your NI 845*x* device supports. If your hardware does not support the supplied clock rate, a warning is generated, and the next smallest supported clock rate is used. If the supplied clock rate is smaller than the smallest supported clock rate, an error is generated.

The default value of this property is `100` kHz.

Address Size

Specifies the addressing scheme to use when addressing the I^{2}C slave device this configuration describes. **Address Size** uses the following values:

`7 Bits`

The NI-845*x* hardware uses the standard 7-bit addressing when communicating with the I^{2}C slave device.

`10 Bits`

The NI-845*x* hardware uses the extended 10-bit addressing when communicating with the I^{2}C slave device.

The default value of this property is `7 Bits`.

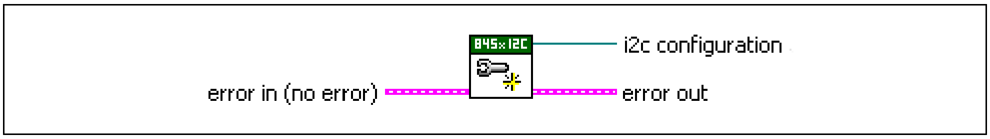
Address

Specifies the I^{2}C slave address. The default address is `0`. For 7-bit device addressing, the Philips I^{2}C Specification defines a 7-bit slave address and a direction bit. During the address phase of an I^{2}C transaction, these values are sent across the bus as one byte (slave address in bits 7–1, direction in bit 0). The NI-845*x* software follows the convention used in the Philips I^{2}C Specification and defines an address for a 7-bit device as a 7-bit value. The NI-845*x* software internally sets the direction bit to the correct value, depending on the function (write or read). Some manufacturers specify the address for their 7-bit device as a byte. In such cases, bits 7–1 contain the slave address, and bit 0 contains the direction. When using the NI-845*x* software, discard the direction bit and right-shift the byte value by one to create the 7-bit address.

NI-845x I2C Create Configuration Reference.vi

Purpose

Creates a new NI-845x I^{2}C configuration.



Inputs

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Output

i2c configuration is a reference to the newly created NI-845x I^{2}C configuration.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

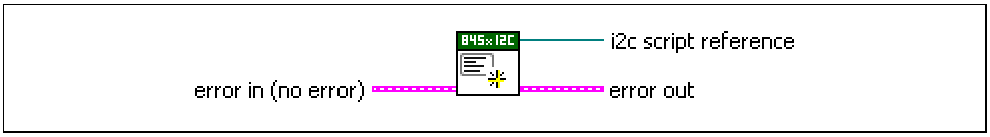
Description

Use **NI-845x I2C Create Configuration Reference.vi** to create a new configuration to use with the NI-845*x* I^{2}C Basic API. Pass the reference to a property node to make the configuration match the settings of your I^{2}C slave. Then, pass the configuration to the I^{2}C basic functions to execute them on the described I^{2}C slave. After you finish communicating with your I^{2}C slave, pass the reference into a new property node to reconfigure it or use **NI-845x Close Reference.vi** to delete the configuration.

NI-845x I2C Create Script Reference.vi

Purpose

Creates a new NI-845*x* I²C script.



Inputs

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Output

i2c script reference is a reference to the newly created NI-845*x* I²C script.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

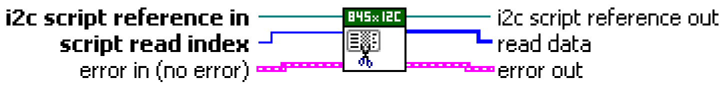
Description

Use **NI-845x I2C Create Script Reference.vi** to create a new script to use with the NI-845*x* I²C Advanced API. Pass the reference to I²C script functions to create the script. Then, call **NI-845x I2C Run Script.vi** to execute your script on your NI 845*x* device. After you finish executing your script, use **NI-845x Close Reference.vi** to delete the script.

NI-845x I2C Extract Script Read Data.vi

Purpose

Extracts the desired read data from an I²C script, referenced by **i2c script reference in**, which has been processed by **NI-845x I2C Run Script.vi**. Each script read command (**NI-845x I2C Script Read.vi**, **NI-845x I2C Script DIO Read Port.vi**, **NI-845x I2C Script DIO Read Line.vi**) returns a script read index. Data may be extracted for each script read index in a script, by wiring each to a separate **NI-845x I2C Extract Script Read Data.vi**.



Inputs

i2c script reference in is a reference to an I²C script that is run on an NI 845*x* device.

script read index identifies the read in the script whose data should be extracted.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

i2c script reference out is a reference to the I^{2}C script after this VI runs.

read data is the data returned for the script command specified by **script read index**.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Description

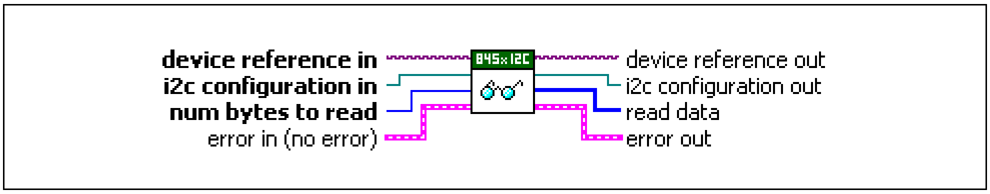
Use **NI-845x I2C Extract Script Read Data.vi** to extract the desired read data from an I^{2}C script, referenced by **i2c script reference in**, which has been processed by **NI-845x I2C Run Script.vi**. Each I^{2}C script read command (**NI-845x I2C Script Read.vi**, **NI-845x I2C Script DIO Read Port.vi, NI-845x I2C Script DIO Read Line.vi**) returns a script read index.

Data may be extracted for each script read in different ways. For example, you can wire the script read index output of each script read VI to its own **NI-845x I2C Extract Script Read Data.vi**. You can also place **NI-845x I2C Extract Script Read Data.vi** in a For Loop and wire the loop iteration terminal to the **script read index** input. Add one to the script read index output of the last read and wire this value to the loop count terminal. The output of the For Loop will be an array of read data arrays.

NI-845x I2C Read.vi

Purpose

Reads an array of data from an I²C slave device. Each byte except for the last byte is acknowledged.



Inputs

device reference in is a reference to an NI 845*x* device.

i2c configuration in is a reference to a specific I²C configuration that describes the characteristics of the device to communicate with. Connect this configuration reference into a property node to set the specific configuration parameters.

num bytes to read specifies the number of bytes to read from the I²C slave.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

device reference out is a reference to the NI 845*x* device after this VI runs.

i2c configuration out is a reference to the I²C configuration after this VI runs.

read data contains an array of read data from the I^{2}C slave.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Description

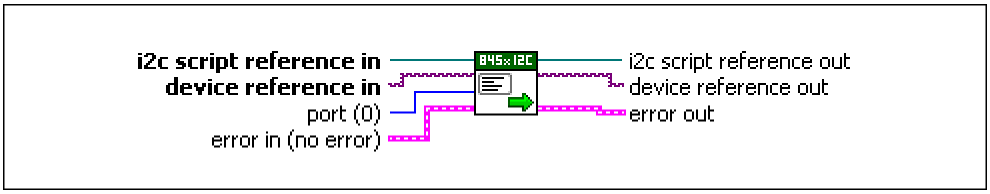
Use **NI-845x I2C Read.vi** to read an array of data from an I^{2}C slave device. Per the Philips I^{2}C specification, each byte read up to the last byte is acknowledged. The last byte is not acknowledged. This VI first waits for the I^{2}C bus to be free. If the I^{2}C bus is not free within the one second timeout of your NI 845*x* device, an error is returned. If the bus is free before the timeout, the NI 845*x* device executes a 7 or 10-bit I^{2}C read transaction, per the Philips I^{2}C specification. The address type (7 or 10-bit) and other configuration parameters are specified by the configuration wired into **i2c configuration in**. If the NI 845*x* device tries to access the bus at the same time as another I^{2}C master device and loses arbitration, the read transaction is terminated and an error is returned. If the address of the transaction is not acknowledged by the slave device, an error is returned. Otherwise, the transaction is completed, and a stop condition is generated per the Philips I^{2}C specification.

Before using **NI-845x I2C Read.vi**, you need to ensure that the configuration parameters specified in **i2c configuration in** are correct for the device you want to access.

NI-845x I2C Run Script.vi

Purpose

Executes an I²C script referenced by **i2c script reference in** on the device referenced by **device reference in**.



Inputs

i2c script reference in is a reference to an I²C script that is run on an NI 845*x* device.

device reference in is a reference to an NI 845*x* device.

port specifies the I²C port this script runs on.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

i2c script reference out is a reference to the I²C script after this VI runs.

device reference out is a reference to the NI 845*x* device after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

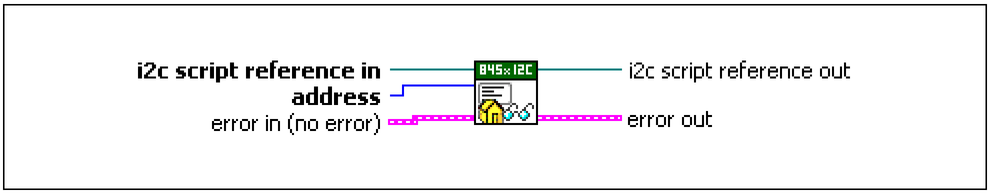
Description

Use **NI-845x I2C Run Script.vi** to execute an I^{2}C script referenced by **i2c script reference in** on the device referenced by **device reference in**. You must first create an I^{2}C script using the I^{2}C scripting VIs. Next, you wire its script reference into **i2c script reference in**. If you have multiple NI 845*x* devices installed in your system, you can select which device to write your I^{2}C script to by wiring its device reference to **device reference in**. If your NI 845*x* device supports multiple I^{2}C ports, you can also select which port to write your I^{2}C script to. For single I^{2}C port NI 845*x* devices, you must use the default port (`0`). In this way, you can create one script to run on various NI 845*x* devices, on various I^{2}C ports within those devices. **NI-845x I2C Run Script.vi** loads and executes your I^{2}C script on the NI 845*x* device and I^{2}C port you specify, then returns success or error. If your script contained any read commands, you may use **NI-845x I2C Extract Script Read Data.vi** to extract the read data after executing **NI-845x I2C Run Script.vi**.

NI-845x I2C Script Address+Read.vi

Purpose

Adds an I^{2}C Script Address+Read command to an I^{2}C script referenced by **i2c script reference in**. This command writes a 7-bit address to the I^{2}C bus. The direction bit is internally set to `1` for read.



Inputs

i2c script reference in is a reference to an I^{2}C script that is run on an NI 845*x* device.

address specifies the 7-bit address to read. For 7-bit device addressing, the Philips I^{2}C Specification defines a 7-bit slave address and a direction bit. During the address phase of an I^{2}C transaction, these values are sent across the bus as one byte (slave address in bits 7–1, direction in bit 0). The NI-845*x* software follows the convention used in the Philips I^{2}C Specification and defines an address for a 7-bit device as a 7-bit value. The NI-845*x* software internally sets the direction bit to the correct value, depending on the function (write or read). Some manufacturers specify the address for their 7-bit device as a byte. In such cases, bits 7–1 contain the slave address, and bit 0 contains the direction. When using the NI-845*x* software, discard the direction bit and right-shift the byte value by one to create the 7-bit address.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

i2c script reference out is a reference to the I^{2}C script after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

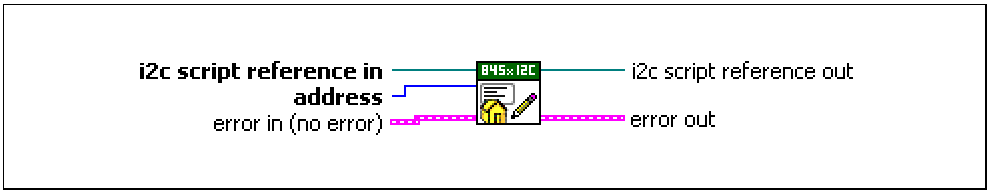
Description

Use **NI-845x I2C Script Address+Read.vi** to add an I^{2}C Script Address+Read command to an I^{2}C script referenced by **i2c script reference in**. This command writes a 7-bit address to the I^{2}C bus connected to the I^{2}C port you specify when you use **NI-845x I2C Run Script.vi** to execute the script. The direction bit is internally set to `1` for read. This command assumes that a start condition has been previously issued to the I^{2}C bus using an I^{2}C script start command. It clocks out the 7-bit address and direction bit and then waits for a slave device on the I^{2}C bus to acknowledge or not acknowledge the address. If a slave does not acknowledge the address, **NI-845x I2C Run Script.vi** exits with an error.

NI-845x I2C Script Address+Write.vi

Purpose

Adds an I²C Script Address+Write command to an I²C script referenced by **i2c script reference in**. This command writes a 7-bit address to the I²C bus. The direction bit is internally set to `0` for write.



Inputs

i2c script reference in is a reference to an I²C script that is run on an NI 845*x* device.

address specifies the 7-bit address to write. For 7-bit device addressing, the Philips I²C Specification defines a 7-bit slave address and a direction bit. During the address phase of an I²C transaction, these values are sent across the bus as one byte (slave address in bits 7–1, direction in bit 0). The NI-845*x* software follows the convention used in the Philips I²C Specification and defines an address for a 7-bit device as a 7-bit value. The NI-845*x* software internally sets the direction bit to the correct value, depending on the function (write or read). Some manufacturers specify the address for their 7-bit device as a byte. In such cases, bits 7–1 contain the slave address, and bit 0 contains the direction. When using the NI-845*x* software, discard the direction bit and right-shift the byte value by one to create the 7-bit address.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is

returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

i2c script reference out is a reference to the I²C script after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

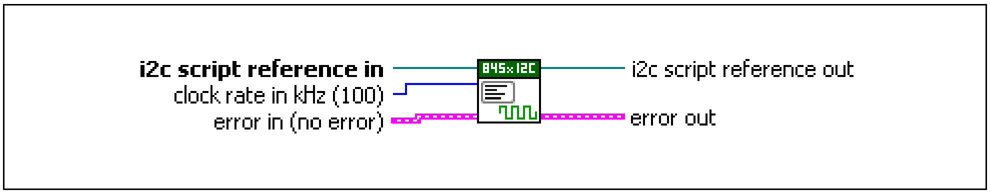
Description

Use **NI-845x I2C Script Address+Write.vi** to add an I²C Script Address+Write command to an I²C script referenced by **i2c script reference in**. This command writes a 7-bit address to the I²C bus connected to the I²C port you specify when you use **NI-845x I2C Run Script.vi** to execute the script. The direction bit is internally set to `0` for write. This command assumes that a start condition has been previously issued to the I²C bus using an I²C script start command. It clocks out the 7-bit address and direction bit and then waits for a slave device on the I²C bus to acknowledge or not acknowledge the address. If a slave does not acknowledge the address, **NI-845x I2C Run Script.vi** exits with an error.

NI-845x I2C Script Clock Rate.vi

Purpose

Adds an I²C Script Clock Rate command to an I²C script referenced by **i2c script reference in**. This command sets the I²C clock rate.



Inputs

i2c script reference in is a reference to an I²C script that is run on an NI 845*x* device.

clock rate in kHz specifies the I²C clock rate. Refer to your hardware user guide to determine which clock rates your NI 845*x* device supports.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

i2c script reference out is a reference to the I²C script after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

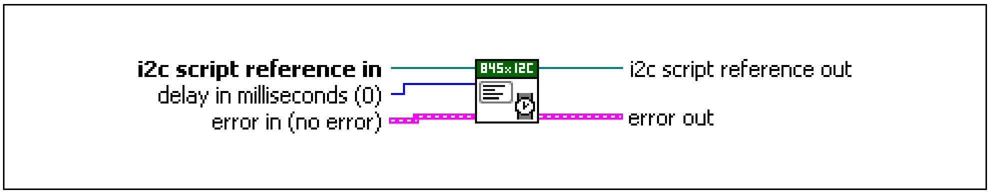
Description

Use **NI-845x I2C Script Clock Rate.vi** to add an I^{2}C Script Clock Rate command to an I^{2}C script referenced by **i2c script reference in**. This command sets the I^{2}C clock rate for the I^{2}C port you specify when you use **NI-845x I2C Run Script.vi** to execute the script. The NI 845*x* device can clock data only at specific rates. If the selected rate is not one of the rates your hardware supports, the NI-845*x* driver adjusts it down to a supported rate and generates a warning. If the selected rate is lower than all supported rates, an error is generated.

NI-845x I2C Script Delay.vi

Purpose

Adds an I^{2}C Script Delay command to an I^{2}C script referenced by **i2c script reference in**. This command adds a delay after the previous I^{2}C script command.



Inputs

i2c script reference in is a reference to an I^{2}C script that is run on an NI 845*x* device.

delay in milliseconds specifies the desired delay.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

i2c script reference out is a reference to the I^{2}C script after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

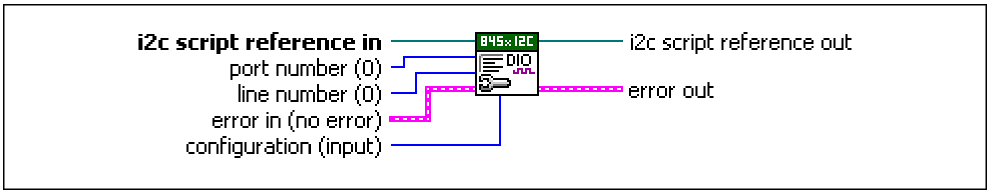
Description

Use **NI-845x I2C Script Delay.vi** to add an I²C Script Delay command to an I²C script referenced by **i2c script reference in**. This command adds a delay after the previous I²C script command.

NI-845x I2C Script DIO Configure Line.vi

Purpose

Adds an I²C Script DIO Configure Line command to an I²C script referenced by **i2c script reference in**. This command configures a DIO line on an NI 845*x* device.



Inputs

i2c script reference in is a reference to an I²C script that is run on an NI 845*x* device.

port number specifies the DIO port that contains the **line number**.

line number specifies the DIO line to configure.

configuration specifies the line configuration. **configuration** uses the following values:

input	The line is configured for input.
output	The line is configured for output.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

i2c script reference out is a reference to the I²C script after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

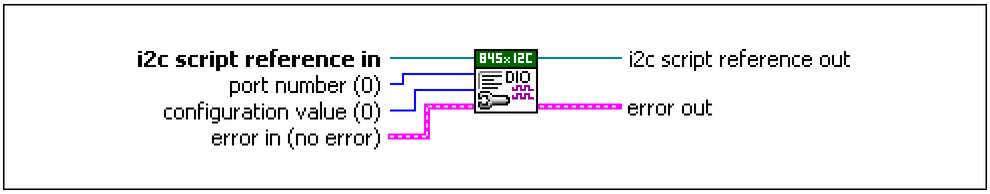
Description

Use **NI-845x I2C Script DIO Configure Line.vi** to add an I²C Script DIO Configure Line command to an I²C script referenced by **i2c script reference in**. This command allows you to configure one line, specified by **line number**, of a byte-wide DIO port, as an input or output. For NI 845*x* devices with multiple DIO ports, use the **port number** input to select the desired port. For NI 845*x* devices with one DIO port, **port number** must be left at the default (`0`).

NI-845x I2C Script DIO Configure Port.vi

Purpose

Adds an I²C Script DIO Configure Port command to an I²C script referenced by **i2c script reference in**. This command configures a DIO port on an NI 845*x* device.



Inputs

i2c script reference in is a reference to an I²C script that is run on an NI 845*x* device.

port number specifies the DIO port to configure.

configuration value is a bitmap that specifies the function of each individual line of a port. If bit *x* = 1, line *x* is an output. If bit *x* = 0, line *x* is an input.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

i2c script reference out is a reference to the I^{2}C script after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

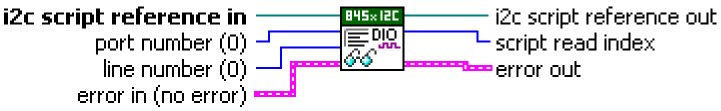
Description

Use **NI-845x I2C Script DIO Configure Port.vi** to add an I^{2}C Script DIO Configure Port command to an I^{2}C script referenced by **i2c script reference in**. This command allows you to configure all eight lines of a byte-wide DIO port. Setting a bit to `1` configures the corresponding DIO port line for output. Setting a bit to `0` configures the corresponding port line for input. For NI 845*x* devices with multiple DIO ports, use the **port number** input to select the port to configure. For NI 845*x* devices with one DIO port, **port number** must be left at the default (`0`).

NI-845x I2C Script DIO Read Line.vi

Purpose

Adds an I^{2}C Script DIO Read Line command to an I^{2}C script referenced by **i2c script reference in**. This command reads from a DIO line on an NI 845*x* device.



Inputs

i2c script reference in is a reference to an I^{2}C script that is run on an NI 845*x* device.

port number specifies the DIO port that contains the **line number**.

line number specifies the DIO line to read.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

i2c script reference out is a reference to the I^{2}C script after this VI runs.

script read index is the index of the read command within the script. It is used as an input into **NI-845x I2C Extract Script Read Data.vi**.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Description

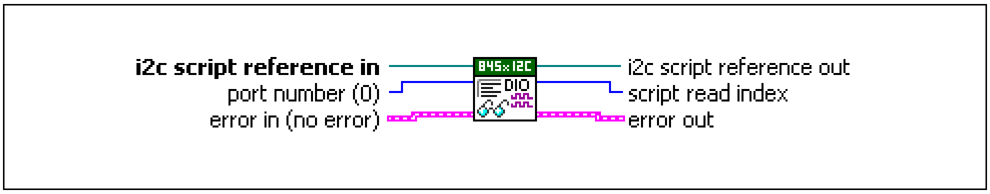
Use **NI-845x I2C Script DIO Read Line.vi** to add an I²C Script DIO Read command to an I²C script referenced by **i2c script reference in**. This command allows you to read one line, specified by **line number**, of a byte-wide DIO port. For NI 845*x* devices with multiple DIO ports, use the **port number** input to select the desired port. For NI 845*x* devices with one DIO port, **port number** must be left at the default (`0`).

To obtain the logic level read from the specified DIO port line, wire **script read index** to **NI-845x I2C Extract Script Read Data.vi** after script execution. If **NI-845x I2C Extract Script Read Data.vi** returns `0`, the logic level read on the specified line was low. If **NI-845x I2C Extract Script Read Data.vi** returns `1`, the logic level read on the specified line was high.

NI-845x I2C Script DIO Read Port.vi

Purpose

Adds an I²C Script DIO Read Port command to an I²C script referenced by **i2c script reference in**. This command reads from a DIO port on an NI 845*x* device.



Inputs

i2c script reference in is a reference to an I²C script that is run on an NI 845*x* device.

port number specifies the DIO port to read.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

- **status** is TRUE if an error occurred. This VI is not executed when status is TRUE.

- **code** is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

- **source** identifies the VI where the error occurred.

Outputs

i2c script reference out is a reference to the I²C script after this VI runs.

script read index is the index of the read command within the script. It is used as an input into **NI-845x I2C Extract Script Read Data.vi**.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Description

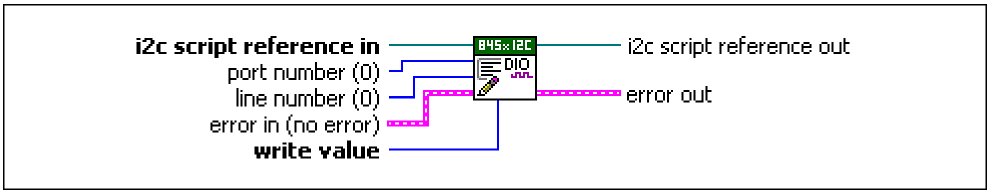
Use **NI-845x I2C Script DIO Read Port.vi** to add an I^{2}C Script DIO Read Port command to an I^{2}C script referenced by **i2c script reference in**. This command allows you to read all 8 bits on a byte-wide DIO port. For NI 845*x* devices with multiple DIO ports, use the **port number** input to select the desired port. For NI 845*x* devices with one DIO port, **port number** must be left at the default (`0`).

To obtain the data byte read from the specified DIO port, wire **script read index** to **NI-845x I2C Extract Script Read Data.vi** after script execution, which returns the data byte read by this script command.

NI-845x I2C Script DIO Write Line.vi

Purpose

Adds an I²C Script DIO Write Line command to an I²C script referenced by **i2c script reference in**. This command writes to a DIO line on an NI 845*x* device.



Inputs

i2c script reference in is a reference to an I²C script that is run on an NI 845*x* device.

port number specifies the DIO port that contains the **line number**.

line number specifies the DIO line to write.

write value specifies the value to write to the line. **write value** uses the following values:

`0` (Logic Low)	The line is set to the logic low state.
`1` (Logic High)	The line is set to the logic high state.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

i2c script reference out is a reference to the I²C script after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

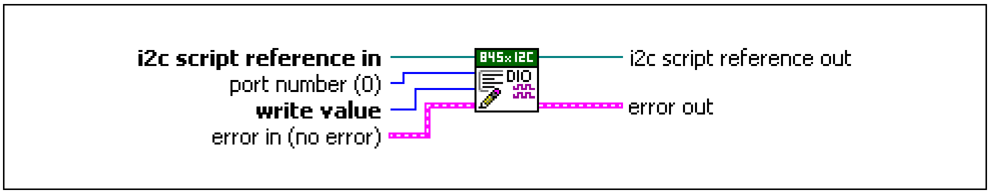
Description

Use **NI-845x I2C Script DIO Write Line.vi** to add an I²C Script DIO Write Line command to an I²C script referenced by **i2c script reference in**. This command allows you to write one line, specified by **line number**, of a byte-wide DIO port. If **write value** is `1`, the specified line's output is driven to a high logic level. If **write value** is `0`, the specified line's output is driven to a low logic level. For NI 845*x* devices with multiple DIO ports, use the **port number** input to select the desired port. For NI 845*x* devices with one DIO port, **port number** must be left at the default (`0`).

NI-845x I2C Script DIO Write Port.vi

Purpose

Adds an I²C Script DIO Write Port command to an I²C script referenced by **i2c script reference in**. This command writes to a DIO port on an NI 845*x* device.



Inputs

i2c script reference in is a reference to an I²C script that is run on an NI 845*x* device.

port number specifies the DIO port to write.

write value is the value to write to the DIO port. Only lines configured for output are updated.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

i2c script reference out is a reference to the I²C script after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

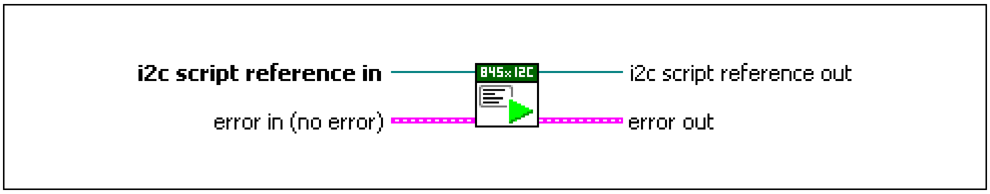
Description

Use **NI-845x I2C Script DIO Write Port.vi** to add an I^{2}C Script DIO Write Port command to an I^{2}C script referenced by **i2c script reference in**. This command allows you to write all 8 bits on a byte-wide DIO port. For NI 845*x* devices with multiple DIO ports, use the **port number** input to select the desired port. For NI 845*x* devices with one DIO port, **port number** must be left at the default (`0`).

NI-845x I2C Script Issue Start.vi

Purpose

Adds an I²C Script Issue Start command to an I²C script referenced by **i2c script reference in**. This command issues a start condition on the I²C bus.



Inputs

i2c script reference in is a reference to an I²C script that is run on an NI 845*x* device.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of 0 means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

i2c script reference out is a reference to the I²C script after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of 0 means success. A negative value means error: VI did not execute

the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

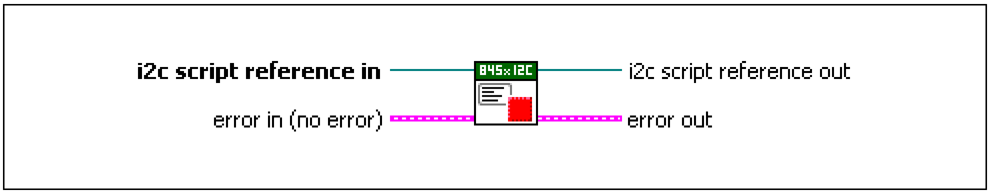
Description

Use **NI-845x I2C Script Issue Start.vi** to add an I^{2}C Script Issue Start command to an I^{2}C script referenced by **i2c script reference in**. This command issues a start condition on the I^{2}C bus connected to the I^{2}C port you specify when you use **NI-845x I2C Run Script.vi** to execute the script. This command first waits for the I^{2}C bus to be free. If the I^{2}C bus is not free within the one second timeout of your NI 845*x* device, an error is returned when **NI-845x I2C Run Script.vi** is executed. If the bus is free before the timeout, the NI 845*x* device issues the start condition on the I^{2}C bus connected to the specified I^{2}C port. This command should also be used to issue a restart condition within an I^{2}C transaction.

NI-845x I2C Script Issue Stop.vi

Purpose

Adds an I²C Script Issue Stop command to an I²C script referenced by **i2c script reference in**. This command issues a stop condition on the I²C bus.



Inputs

i2c script reference in is a reference to an I²C script that is run on an NI 845*x* device.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

i2c script reference out is a reference to the I²C script after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute

the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

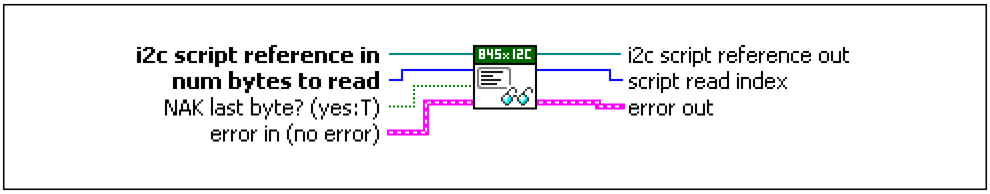
Description

Use **NI-845x I2C Script Issue Stop.vi** to add an I^{2}C Script Issue Stop command to an I^{2}C script referenced by **i2c script reference in**. This command issues a stop condition on the I^{2}C bus connected to the I^{2}C port you specify when you use **NI-845x I2C Run Script.vi** to execute the script. Per the Philips I^{2}C specification, all I^{2}C transactions must be terminated with a stop condition.

NI-845x I2C Script Read.vi

Purpose

Adds an I²C Script Read command to an I²C script referenced by **i2c script reference in**. This command reads an array of data from an I²C slave device.



Inputs

i2c script reference in is a reference to an I²C script that is run on an NI 845*x* device.

num bytes to read specifies the number of bytes to read from an I²C slave.

NAK Last Byte? sets whether the last byte read is acknowledged (FALSE) or not acknowledged (TRUE) by the I²C interface. If **NAK Last Byte?** is TRUE, all bytes up to the last byte read are acknowledged. The last byte read is not acknowledged. If **NAK Last Byte?** is FALSE, all bytes are acknowledged.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

i2c script reference out is a reference to the I²C script after this VI runs.

script read index is the index of the read command within the script. It is used as an input into **NI-845x I2C Extract Script Read Data.vi**.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Description

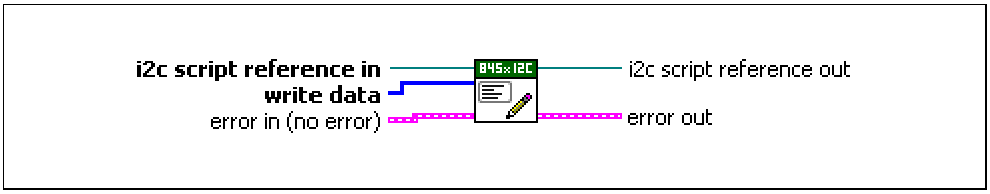
Use **NI-845x I2C Script Read.vi** to add an I²C Script Read command to an I²C script referenced by **i2c script reference in**. This command reads an array of data from a device connected to the I²C port you specify when you use **NI-845x I2C Run Script.vi** to execute the script. This command assumes that a start condition and address+read condition have been issued to the I²C bus using prior I²C script commands. It clocks in **num bytes to read** bytes from the I²C slave device, acknowledging each byte up to the last one. Depending on the type of I²C transaction you want to build, you may want to acknowledge (ACK) or not acknowledge (NAK) the last data byte read, which you can specify with the **NAK last byte?** input.

To obtain the data read from the specified I²C port, you can wire **script read index** to **NI-845x I2C Extract Script Read Data.vi** after execution of the script, which returns the data read by this script command.

NI-845x I2C Script Write.vi

Purpose

Adds an I^{2}C Script Write command to an I^{2}C script referenced by **i2c script reference in**. This command writes an array of data to an I^{2}C slave device.



Inputs

i2c script reference in is a reference to an I^{2}C script that is run on an NI 845*x* device.

write data contains an array of data to write to the I^{2}C slave.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

i2c script reference out is a reference to the I^{2}C script after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

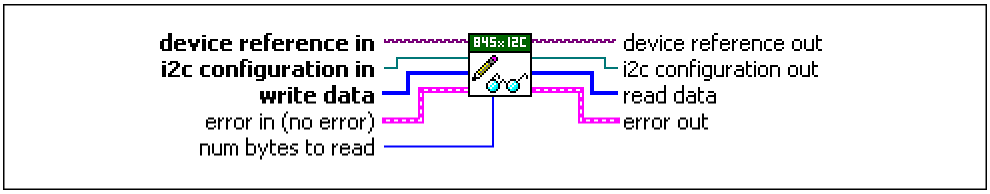
Description

Use **NI-845x I2C Script Write.vi** to add an I^{2}C Script Write command to an I^{2}C script referenced by **i2c script reference in**. This command writes an array of data to an I^{2}C slave device connected to the I^{2}C port you specify when you use **NI-845x I2C Run Script.vi** to execute the script. This command assumes that a start condition and address+write condition have been issued to the I^{2}C bus using prior I^{2}C script commands. It clocks the **write data** array into the I^{2}C slave device, testing for a slave device acknowledge after transmission of each byte. If a slave does not acknowledge a byte, **NI-845x I2C Run Script.vi** exits with an error.

NI-845x I2C Write Read.vi

Purpose

Performs a write followed by read (combined format) on an I^{2}C slave device.



Inputs

device reference in is a reference to an NI 845*x* device.

i2c configuration in is a reference to a specific I^{2}C configuration that describes the characteristics of the device to communicate with. Connect this configuration reference into a property node to set the specific configuration parameters.

write data contains an array of data to write to the I^{2}C slave.

num bytes to read specifies the number of bytes to read from the I^{2}C slave.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

device reference out is a reference to the NI 845*x* device after this VI runs.

i2c configuration out is a reference to the I^{2}C configuration after this VI runs.

read data contains an array of read data from the I^{2}C slave.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Description

Use **NI-845x I2C Write Read.vi** to perform a write followed by read (combined format) on an I^{2}C slave device. During the read portion of the transaction, per the Philips I^{2}C specification, each byte read up to the last byte is acknowledged. The last byte is not acknowledged. This VI first waits for the I^{2}C bus to be free. If the I^{2}C bus is not free within the one second timeout of your NI 845*x* device, an error is returned. If the bus is free before the timeout, the NI 845*x* device executes a 7 or 10-bit I^{2}C write/read transaction. Per the Philips I^{2}C specification, the write/read transaction consists of a start–write–restart–read–stop sequence.

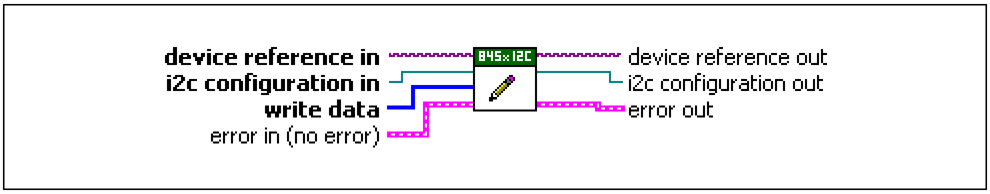
The address type (7 or 10-bit) and other configuration parameters are specified by the configuration wired into **i2c configuration in**. If the NI 845*x* device tries to access the bus at the same time as another I^{2}C master device and loses arbitration, the read transaction is terminated and an error is returned. If an address or byte write within the transaction is not acknowledged by the slave device, an error is returned. Otherwise, the transaction is completed and a stop condition is generated per the Philips I^{2}C specification. It should be noted that this type of combined transaction is provided because it is commonly used (for example, with EEPROMs). The Philips I^{2}C specification provides flexibility in the construction of I^{2}C transactions. The NI-845*x* I^{2}C scripting VIs allow creating and customizing complex I^{2}C transactions as needed.

Before using **NI-845x I2C Write Read.vi**, you need to ensure that the configuration parameters specified in **i2c configuration in** are correct for the device you want to access.

NI-845x I2C Write.vi

Purpose

Writes an array of data to an I^{2}C slave device.



Inputs

device reference in is a reference to an NI 845*x* device.

i2c configuration in is a reference to a specific I^{2}C configuration that describes the characteristics of the device to communicate with. Connect this configuration reference into a property node to set the specific configuration parameters.

write data contains an array of data to write to the I^{2}C slave.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

device reference out is a reference to the NI 845*x* device after this VI runs.

i2c configuration out is a reference to the I^{2}C configuration after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Description

Use **NI-845x I2C Write.vi** to write an array of data to an I^{2}C slave device. This VI first waits for the I^{2}C bus to be free. If the I^{2}C bus is not free within the one second timeout of your NI 845*x* device, an error is returned. If the bus is free before the timeout, the NI 845*x* device executes a 7 or 10-bit I^{2}C write transaction, per the Philips I^{2}C specification. The address type (7 or 10-bit) and other configuration parameters are specified by the configuration wired into **i2c configuration in**. If the NI 845*x* device tries to access the bus at the same time as another I^{2}C master device and loses arbitration, the write transaction is terminated and an error is returned. If any byte of the transaction is not acknowledged by the slave device, an error is returned. Otherwise, the transaction is completed, and a stop condition is generated per the Philips I^{2}C specification.

Before using **NI-845x I2C Write.vi**, you need to ensure that the configuration parameters specified in **i2c configuration in** are correct for the device you currently want to access.

6 Using the NI-845*x* SPI API

This chapter helps you get started with the SPI API.

NI-845x SPI Basic Programming Model

The SPI Basic API provides the most fundamental SPI transaction type: write/read. You can access most off-the-shelf SPI devices using this transaction. The SPI Basic API allows you to easily and quickly develop applications to communicate with these devices. For those situations in which the SPI Basic API does not provide the functionality you need, use the SPI Advanced API to create custom SPI transactions.

When you use the SPI Basic API, the first step is to create an SPI configuration to describe the communication requirements between the 845*x* and the SPI device. To make an SPI configuration, create an SPI configuration reference and set the appropriate properties as desired. You can then read or write data to the SPI device.

The diagram in Figure 6-1 describes the programming model for the NI-845*x* SPI Basic API. Within the application, you repeat this programming model for each SPI device. The diagram is followed by a description of each step in the model.

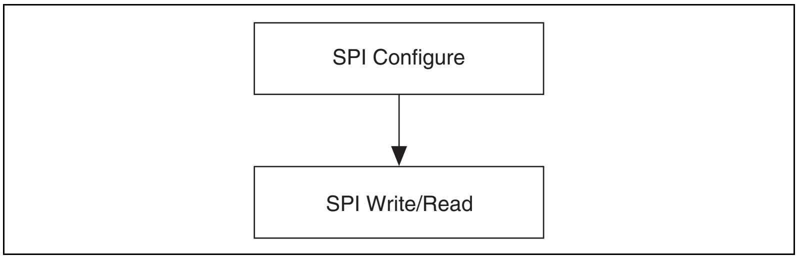


Figure 6-1. NI-845x SPI API Basic Programming Model

SPI Configure

Use the **NI-845x SPI Configuration Property Node** to set the specific SPI configuration that describes the characteristics of the device to communicate with.

SPI Write Read

Use **NI-845x SPI Write.vi** to exchange an array of data with an SPI slave device.

SPI Timing Characteristics

Figure 6-2 and Table 6-1 show the timing characteristics of the SPI bus when using the SPI Basic API. If the timing characteristics of your device do not fit within these parameters, you can use the SPI Advanced API to adjust the bus characteristics to match those of your device.

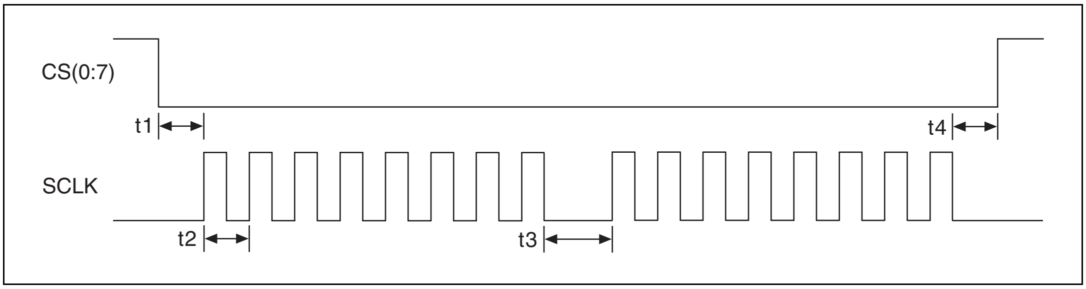


Figure 6-2. SPI Waveform

Table 6-1. NI USB-8451 SPI Timing Characteristics

Symbol	Parameter	Min	Max	Units
t1	CS(0:7) assertion to first SCLK edge	5	15.4	µs
t2	SCLK period	0.08333	20.83	µs
t3	SCLK setup time	8.5	19	µs
t4	Last SCLK edge to CS(0:7) deassertion	7.4	8.24	µs

NI-845x SPI Advanced Programming Model

The SPI Advanced API provides a set of script commands that allow you great flexibility to construct custom SPI transactions to address your particular needs. For example, you can use scripting in the following scenarios:

- Executing individual byte transfers on the bus, with or without variable delays in between, so that you can observe device response.
- Issuing a transaction to a device and measuring its responses (using NI 845*x* DIO pins configured for input) at multiple points within the transaction.
- Doing performance testing, in which you see how a device responds to a variable delay, clock rate change, etc. between each byte transfer within a transaction.
- Gang programming a set of EEPROMs, then verifying the data by reading from each one afterwards.
- Communicating with devices that have an active high chip select line.

When you use the SPI Advanced API, the first step is to create a script that describes the communication between an SPI master and an SPI slave device. Then you execute the script and read data if needed. The script size is limited only by the amount of memory available on your PC. The number of read commands, **NI-845x SPI Script Write Read.vi**, **NI-845x SPI Script DIO Read Port.vi**, and **NI-845x SPI Script DIO Read Line.vi** within each script is limited to 64.

The diagram in Figure 6-3 describes an example of programming with the scripting functions for the NI-845*x* SPI Advanced API. The diagram is followed by a description of each step in the model.

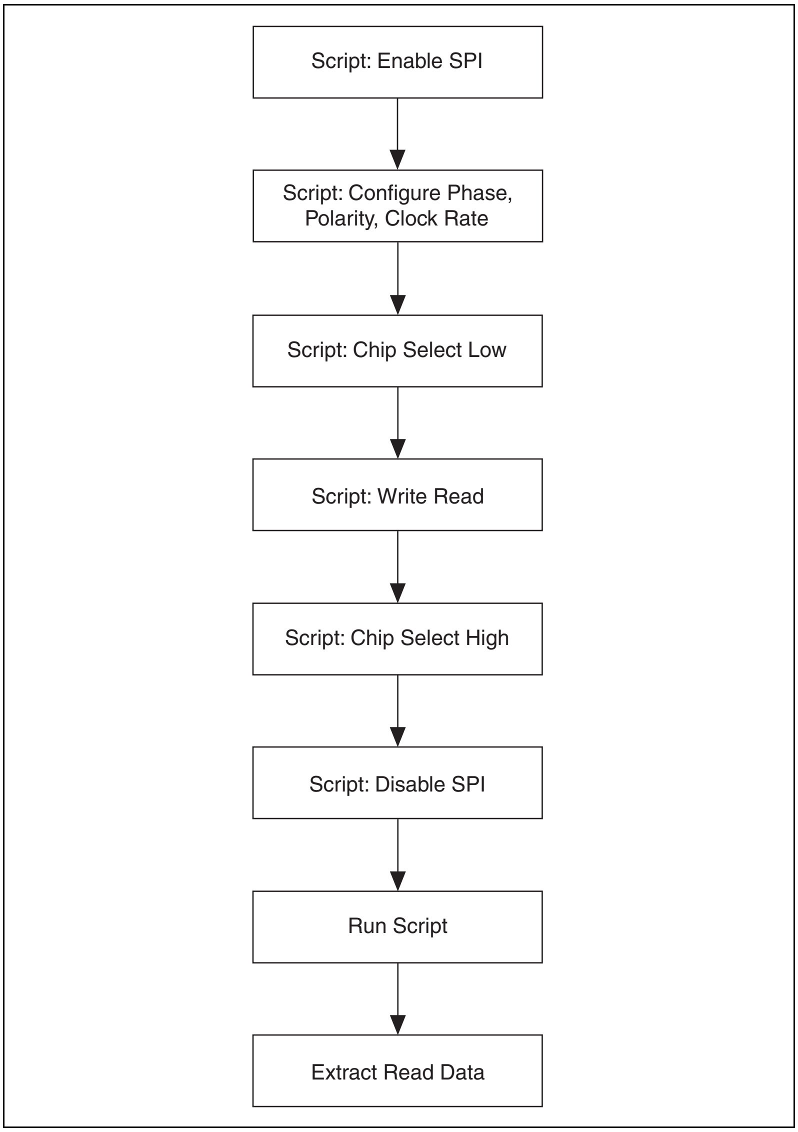


Figure 6-3. Scripting Functions Programming Example

Script: Enable SPI

Use **NI-845x SPI Script Enable SPI.vi** to add an SPI Script Enable SPI command to an SPI script referenced by **spi script reference in**. This command switches the pins on the SPI port you specify when you use **NI-845x SPI Run Script.vi**, from tristate to master mode function.

Script: Configure Phase, Polarity, Clock Rate

Use **NI-845x SPI Script Clock Polarity Phase.vi** to add an SPI Script Clock Polarity Phase command to an SPI script referenced by **spi script reference in**. This command sets the SPI clock idle state (CPOL) and clock edge position within each data bit (CPHA) for the SPI port you specify when you use **NI-845x SPI Run Script.vi** to execute the script.

Use **NI-845x SPI Script Clock Rate.vi** to add an SPI Script Clock Rate command to an SPI script referenced by **spi script reference in**. This command sets the SPI clock rate for the SPI port you specify when you use **NI-845x SPI Run Script.vi** to execute the script.

Script: Chip Select Low

Use **NI-845x SPI Script CS Low.vi** to add an SPI Script CS Low command to an SPI script referenced by **spi script reference in**. This command sets an SPI chip select to the logic low state.

Script: Write Read

Use **NI-845x SPI Script Write Read.vi** to add an SPI Script Write Read command to an SPI script referenced by **spi script reference in**. This command exchanges an array of data with an SPI slave device connected to the SPI port you specify when you use **NI-845x SPI Run Script.vi** to execute the script.

Script: Chip Select High

Use **NI-845x SPI Script CS High.vi** to add an SPI Script CS High command to an SPI script referenced by **spi script reference in**. This command sets an SPI chip select to the logic high state.

Script: Disable SPI

Use **NI-845x SPI Script Disable SPI.vi** to add an SPI Script Disable SPI command to an SPI script referenced by **spi script reference in**. This command tristates the pins on the SPI port you specify when you use **NI-845x SPI Run Script.vi**.

Run Script

Use **NI-845x SPI Run Script.vi** to execute an SPI script referenced by **spi script reference in** on the device referenced by **device reference in**.

Extract Read Data

Use **NI-845x SPI Extract Script Read Data.vi** to extract the desired read data from an SPI script, referenced by **spi script reference in**, which has been processed by **NI-845x SPI Run Script.vi**. Each SPI script read command (**NI-845x SPI Script Write Read.vi**, **NI-845x SPI Script DIO Read Port.vi, NI-845x SPI Script DIO Read Line.vi**) returns a **script read index**. Data may be extracted for each **script read index** in a script, by wiring each to a separate **NI-845x SPI Extract Script Read Data.vi**.

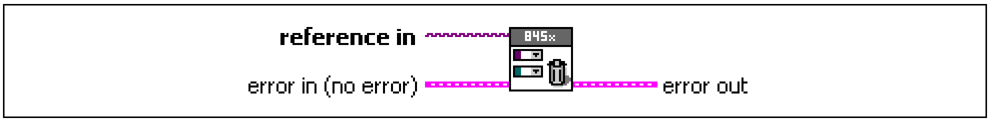
NI-845x SPI API for LabVIEW

This chapter lists the LabVIEW VIs for the NI-845*x* SPI API and describes the format, purpose, and parameters for each VI. The VIs in this chapter are listed alphabetically.

NI-845x Close Reference.vi

Purpose

Closes a previously opened reference.



Inputs

reference in is a reference to an NI 845*x* device, I^{2}C configuration, SPI configuration, I^{2}C script, or SPI script.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

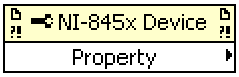
Description

Use **NI-845x Close Reference.vi** to close a previously opened reference.

NI-845x Device Property Node

Purpose

A property node with the NI-845*x* Device class preselected. This property node allows you to modify properties of your NI 845*x* device.



Inputs

device reference in is a reference to an NI 845*x* device.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

device reference out is a reference to an NI 845*x* device after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is

returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Description

The list below describes all valid properties for the **NI-845x Device Property Node**.

Active DIO Port

The **Active DIO Port** property sets the active DIO port for further DIO port configuration. The format for this property is a decimal string. For example, the string `0` represents DIO Port 0. For NI 845*x* devices with one DIO port, the port value must be set to `0`.

DIO Port Voltage

The **DIO Port Voltage** property configures the active DIO port with the desired voltage characteristics. **DIO Port Voltage** uses the following values:

`Open-Drain`	The port is configured for open-drain voltage.
`Push-Pull 3.3 V`	The port is configured for 3.3 V push-pull voltage.

The default value of this property is `Push-Pull 3.3 V`.

DIO Line Direction Map

The **DIO Line Direction Map** property sets the line direction map for the active DIO Port. The value is a bitmap that specifies the function of each individual line within the port. If bit *x* = 1, line *x* is an output. If bit *x* = 0, line *x* is an input.

The default value of this property is `0` (all lines configured for input).

NI-845x Device Reference

Purpose

Specifies the device resource to be used for communication.

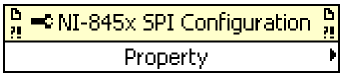
Description

Use the **NI-845x Device Reference** to describe the NI 845*x* device to communicate with. You can wire the reference into a property node to set specific device parameters or to an NI-845*x* API call to invoke the function on the associated NI 845*x* device.

NI-845x SPI Configuration Property Node

Purpose

A property node with the NI-845*x* SPI Configuration class preselected. This property node allows you to query and modify SPI configuration properties of your NI 845*x* device.



Inputs

spi configuration in is a reference to a specific SPI configuration that describes the characteristics of the device to communicate with.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

spi configuration out is a reference to a specific SPI configuration that describes the characteristics of the device to communicate with.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Description

The list below describes all valid properties for the **NI-845x SPI Configuration Property Node**.

Chip Select

Selects the **Chip Select** line for this device number.

The default value for this property is `0`.

Port

Specifies the SPI port that this configuration communicates across.

The default value for this property is `0`.

Refer to your hardware user guide to determine the number of SPI ports your NI 845*x* device supports.

Clock Rate in kHz

Specifies the SPI clock rate. Refer to your hardware user guide to determine which clock rates your NI 845*x* device supports. If your hardware does not support the supplied clock rate, a warning is generated, and the next smallest supported clock rate is used. If the supplied clock rate is smaller than the smallest supported clock rate, an error is generated.

The default value for this property is `1000` kHz (`1` MHz).

Clock Polarity

Sets the idle state of the clock line for the SPI Port. **Clock Polarity** uses the following values:

`0` (Idle Low)

Clock is low in the idle state.

`1` (Idle High)

Clock is high in the idle state.

The default value for this property is `0` (Idle Low).

I32

Clock Phase

Sets the positioning of the data bits relative to the clock edges for the SPI Port. **Clock Phase** uses the following values:

`0` (First Edge)

Data is centered on the first edge of the clock period.

`1` (Second Edge)

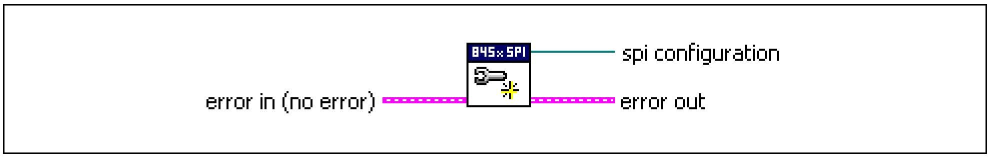
Data is centered on the second edge of the clock period.

The default value for this property is `0` (First Edge).

NI-845x SPI Create Configuration Reference.vi

Purpose

Creates a new NI-845*x* SPI configuration.



Inputs

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

spi configuration is a reference to the newly created NI-845*x* SPI configuration.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

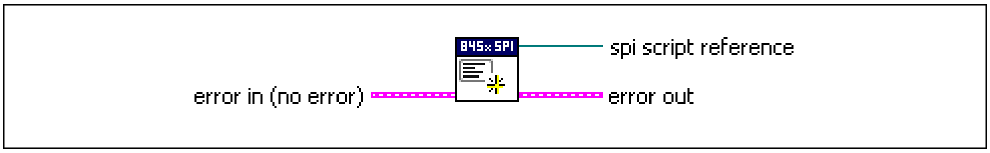
Description

Use **NI-845x SPI Create Configuration Reference.vi** to create a new configuration to use with the NI-845*x* SPI Basic API. Pass the reference to a property node to make the configuration match the settings of your SPI slave. Then, pass the configuration to the SPI basic functions to execute them on the described SPI slave. After you finish communicating with your SPI slave, pass the reference into a new property node to reconfigure it or use **NI-845x Close Reference.vi** to delete the configuration.

NI-845x SPI Create Script Reference.vi

Purpose

Creates a new NI-845*x* SPI script.



Inputs

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

spi script reference is a reference to the newly created NI-845*x* SPI script.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

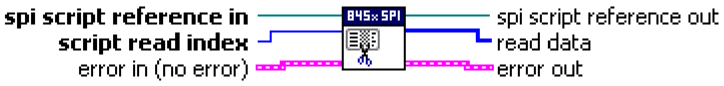
Description

Use **NI-845x SPI Create Script Reference.vi** to create a new script to use with the NI-845*x* SPI Advanced API. Pass the reference to SPI script functions to create the script. Then, call **NI-845x SPI Run Script.vi** to execute your script on your NI 845*x* device. After you have finished executing your script, use **NI-845x Close Reference.vi** to delete the script.

NI-845x SPI Extract Script Read Data.vi

Purpose

Extracts the desired read data from an SPI script, referenced by **spi script reference in**, which has been processed by **NI-845x SPI Run Script.vi**. Each script read command (**NI-845x SPI Script Write Read.vi**, **NI-845x SPI Script DIO Read Port.vi, NI-845x SPI Script DIO Read Line.vi**) returns a script read index. Data may be extracted for each script read index in a script, by wiring each to a separate **NI-845x SPI Extract Script Read Data.vi**.



Inputs

spi script reference in is a reference to an SPI script that is run on an NI 845*x* device.

script read index identifies the read in the script whose data should be extracted.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

- **status** is TRUE if an error occurred. This VI is not executed when status is TRUE.
- **code** is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.
- **source** identifies the VI where the error occurred.

Outputs

spi script reference out is a reference to an SPI script after this VI runs.

read data is the data returned for the script command specified by **script read index**.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Description

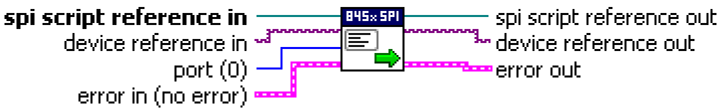
Use **NI-845x SPI Extract Script Read Data.vi** to extract the desired read data from an SPI script, referenced by **spi script reference in**, which has been processed by **NI-845x SPI Run Script.vi**. Each SPI script read command (**NI-845x SPI Script Write Read.vi**, **NI-845x SPI Script DIO Read Port.vi, NI-845x SPI Script DIO Read Line.vi**) returns a script read index.

Data may be extracted for each script read in different ways. For example, you can wire the script read index output of each script read VI to its own **NI-845x SPI Extract Script Read Data.vi**. You can also place **NI-845x SPI Extract Script Read Data.vi** in a For Loop and wire the loop iteration terminal to the **script read index** input. Add one to the script read index output of the last read and wire this value to the loop count terminal. The output of the For Loop will be an array of read data arrays.

NI-845x SPI Run Script.vi

Purpose

Executes an SPI script referenced by **spi script reference in** on the device referenced by **device reference in**.



Inputs

spi script reference in is a reference to an SPI script that is run on an NI 845*x* device.

device reference in is a reference to an NI 845*x* device.

port specifies the SPI port this script will run on.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

spi script reference out is a reference to the SPI script after this VI runs.

device reference out is a reference to the NI 845*x* device after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Description

Use **NI-845x SPI Run Script.vi** to execute an SPI script referenced by **spi script reference in** on the device referenced by **device reference in**. You must first create an SPI script using the SPI scripting VIs. Next, wire its script reference into **spi script reference in**.

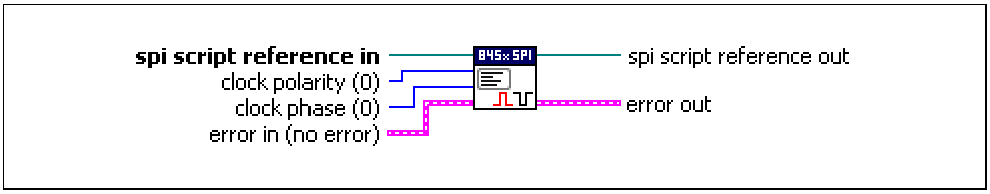
If you have multiple NI 845*x* devices installed in your system, you can select which device to write your SPI script to by wiring its device reference to **device reference in**. If your NI 845*x* device supports multiple SPI ports, you can also select which port to write your SPI script to. For single SPI port NI 845*x* devices, you must use the default port (`0`). In this way, you can create one script to run on various NI 845*x* devices, on various SPI ports within those devices.

NI-845x SPI Run Script.vi loads and executes your SPI script on the NI 845*x* device and SPI port you specify, then returns success or error. If your script contained any read commands, you may use **NI-845x SPI Extract Script Read Data.vi** to extract the read data after executing **NI-845x SPI Run Script.vi**.

NI-845x SPI Script Clock Polarity Phase.vi

Purpose

Adds an SPI Script Clock Polarity Phase command to an SPI script referenced by **spi script reference in**. This command sets the SPI clock idle state (CPOL) and clock edge position within each data bit (CPHA).



Inputs

spi script reference in is a reference to an SPI script that is run on an NI 845*x* device.

clock polarity sets the idle state of the clock line. The values for **clock polarity** are:

`0` (Idle Low)	low in idle state
`1` (Idle High)	high in idle state

clock phase sets the positioning of the data bits relative to the clock edges. The values for **clock phase** are:

`0` (First Edge)	data centered on first edge of clock period
`1` (Second Edge)	data centered on second edge of clock period

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

spi script reference out is a reference to the SPI script after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Description

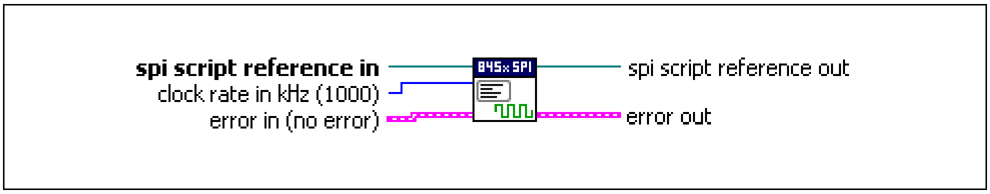
Use **NI-845x SPI Script Clock Polarity Phase.vi** to add an SPI Script Clock Polarity Phase command to an SPI script referenced by **spi script reference in**. This command sets the SPI clock idle state (CPOL) and clock edge position within each data bit (CPHA) for the SPI port you specify when you use **NI-845x SPI Run Script.vi** to execute the script.

Clock polarity sets the idle state of the SPI clock line. The default (`0`) sets the clock line to idle at a low logic level. Setting the clock polarity to `1` sets the clock line to idle at a high logic level. **Clock phase** sets the SPI clock edge on which the NI 845*x* SPI port centers each MOSI data bit. The default (`0`) centers each MOSI data bit on the first edge of each clock cycle. Setting the clock phase to `1` causes each MOSI data bit to be centered on the second edge of each clock cycle.

NI-845x SPI Script Clock Rate.vi

Purpose

Adds an SPI Script Clock Rate command to an SPI script referenced by **spi script reference in**. This command sets the SPI clock rate.



Inputs

spi script reference in is a reference to an SPI script that is run on an NI 845*x* device.

clock rate in kHz specifies the SPI clock rate. Refer to your hardware user guide to determine which clock rates your NI 845*x* device supports.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

spi script reference out is a reference to the SPI script after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

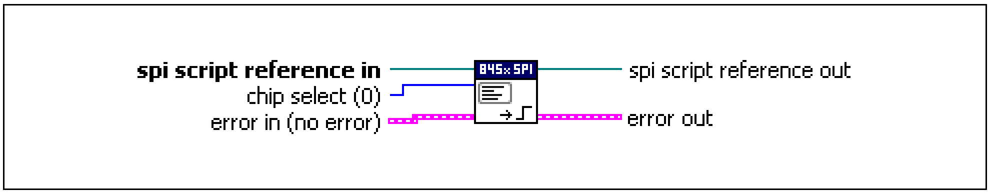
Description

Use **NI-845x SPI Script Clock Rate.vi** to add an SPI Script Clock Rate command to an SPI script referenced by **spi script reference in**. This command sets the SPI clock rate for the SPI port you specify when you use **NI-845x SPI Run Script.vi** to execute the script. The NI 845*x* device can clock data only at specific rates. If the selected rate is not one of the rates your hardware supports, the NI-845*x* software adjusts it down to a supported rate and generates a warning. If the selected rate is lower than all supported rates, an error is generated.

NI-845x SPI Script CS High.vi

Purpose

Adds an SPI Script CS High command to an SPI script referenced by **spi script reference in**. This command sets an SPI chip select to the logic high state.



Inputs

spi script reference in is a reference to an SPI script that is run on an NI 845*x* device.

chip select specifies the chip select to set high.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

spi script reference out is a reference to the SPI script after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

I32 **code** is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

abc **source** identifies the VI where the error occurred.

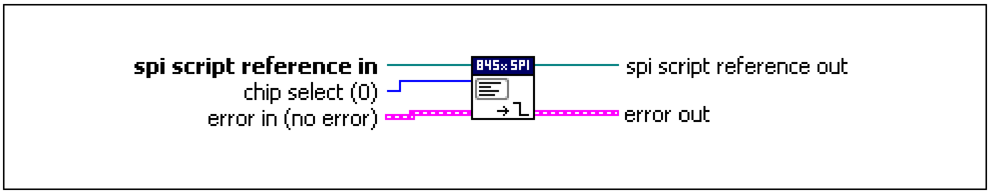
Description

Use **NI-845x SPI Script CS High.vi** to add an SPI Script CS High command to an SPI script referenced by **spi script reference in**. This command sets an SPI chip select to the logic high state.

NI-845x SPI Script CS Low.vi

Purpose

Adds an SPI Script CS Low command to an SPI script referenced by **spi script reference in**. This command sets an SPI chip select to the logic low state.



Inputs

spi script reference in is a reference to an SPI script that is run on an NI 845*x* device.

chip select specifies the chip select to set low.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

spi script reference out is a reference to the SPI script after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

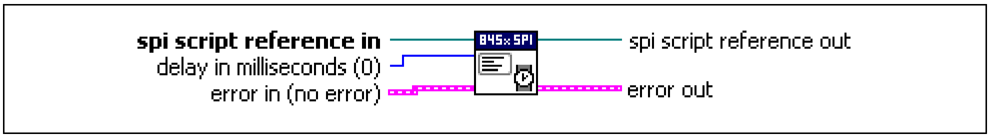
Description

Use **NI-845x SPI Script CS Low.vi** to add an SPI Script CS Low command to an SPI script referenced by **spi script reference in**. This command sets an SPI chip select to the logic low state.

NI-845x SPI Script Delay.vi

Purpose

Adds an SPI Script Delay command to an SPI script referenced by **spi script reference in**. This command adds a delay after the previous SPI script command.



Inputs

spi script reference in is a reference to an SPI script that is run on an NI 845*x* device.

delay in milliseconds specifies the desired delay.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

spi script reference out is a reference to the SPI script after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute

the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

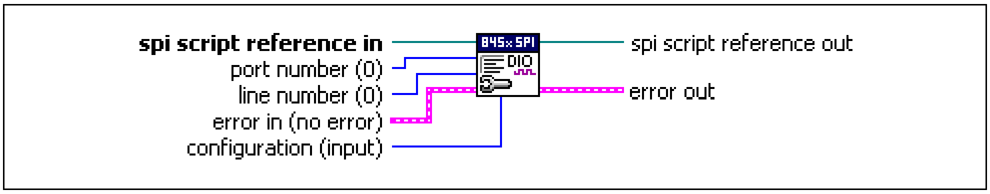
Description

Use **NI-845x SPI Script Delay.vi** to add an SPI Script Delay command to an SPI script referenced by **spi script reference in**. This command adds a delay after the previous SPI script command.

NI-845x SPI Script DIO Configure Line.vi

Purpose

Adds an SPI Script DIO Configure Line command to an SPI script referenced by **spi script reference in**. This command configures a DIO line on an NI 845*x* device.



Inputs

spi script reference in is a reference to an SPI script that is run on an NI 845*x* device.

port number specifies the DIO port that contains the **line number**.

line number specifies the DIO line to configure.

configuration specifies the line configuration. **configuration** uses the following values:

`input`	The line is configured for input.
`output`	The line is configured for output.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

spi script reference out is a reference to the SPI script after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

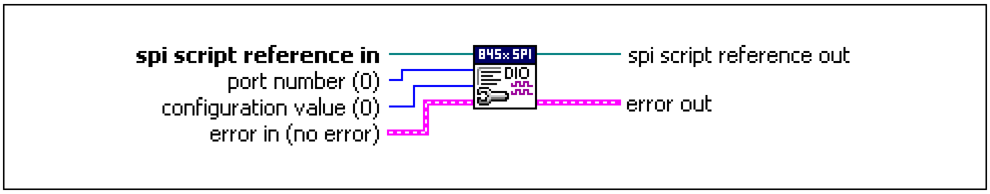
Description

Use **NI-845x SPI Script DIO Configure Line.vi** to add an SPI Script DIO Configure Line command to an SPI script referenced by **spi script reference in**. This command allows you to configure one line, specified by **line number**, of a byte-wide DIO port, as in input or output. For NI 845*x* devices with multiple DIO ports, use the **port number** input to select the desired port. For NI 845*x* devices with one DIO port, **port number** must be left at the default (`0`).

NI-845x SPI Script DIO Configure Port.vi

Purpose

Adds an SPI Script DIO Configure Port command to an SPI script referenced by **spi script reference in**. This command configures a DIO port on an NI 845*x* device.



Inputs

spi script reference in is a reference to an SPI script that is run on an NI 845*x* device.

port number specifies the DIO port to configure.

configuration value is a bitmap that specifies the function of each individual line of a port. If bit *x* = 1, line *x* is an output. If bit *x* = 0, line *x* is an input.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

spi script reference out is a reference to the SPI script after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

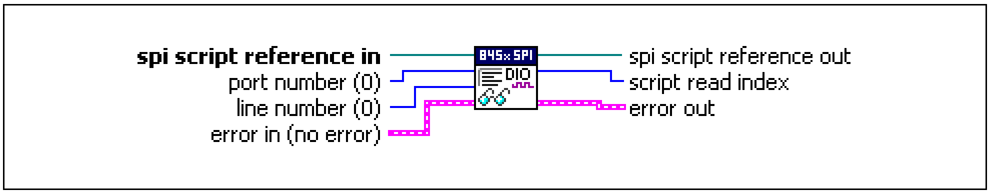
Description

Use **NI-845x SPI Script DIO Configure Port.vi** to add an SPI Script DIO Configure Port command to an SPI script referenced by **spi script reference in**. This command allows you to configure all eight lines of a byte-wide DIO port. Setting a bit to `1` configures the corresponding DIO port line for output. Setting a bit to `0` configures the corresponding port line for input. For NI 845*x* devices with multiple DIO ports, use the **port number** input to select the desired port. For NI 845*x* devices with one DIO port, **port number** must be left at the default (`0`).

NI-845x SPI Script DIO Read Line.vi

Purpose

Adds an SPI Script DIO Read Line command to an SPI script referenced by **spi script reference in**. This command reads from a DIO port on an NI 845*x* device.



Inputs

spi script reference in is a reference to an SPI script that is run on an NI 845*x* device.

port number specifies the DIO port that contains the **line number**.

line number specifies the DIO line to read.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

spi script reference out is a reference to the SPI script after this VI runs.

script read index is the index of the read command within the script. It is used as an input into **NI-845x SPI Extract Script Read Data.vi**.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Description

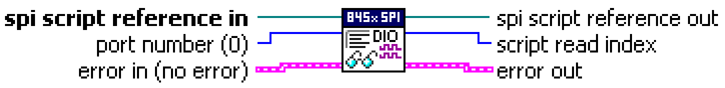
Use **NI-845x SPI Script DIO Read Line.vi** to add an SPI Script DIO Read command to an SPI script referenced by **spi script reference in**. This command allows you to read one line, specified by **line number**, of a byte-wide DIO port. For NI 845*x* devices with multiple DIO ports, use the **port number** input to select the desired port. For NI 845*x* devices with one DIO port, **port number** must be left at the default (`0`).

To obtain the logic level read from the specified DIO port line, wire **script read index** to **NI-845x SPI Extract Script Read Data.vi** after script execution. If **NI-845x SPI Extract Script Read Data.vi** returns `0`, the logic level read on the specified line was low. If **NI-845x SPI Extract Script Read Data.vi** returns `1`, the logic level read on the specified line was high.

NI-845x SPI Script DIO Read Port.vi

Purpose

Adds an SPI Script DIO Read Port command to an SPI script referenced by **spi script reference in**. This command reads from a DIO port on an NI 845*x* device.



Inputs

spi script reference in is a reference to an SPI script that is run on an NI 845*x* device.

port number specifies the DIO port to read.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

spi script reference out is a reference to the SPI script after this VI runs.

script read index is the index of the read command within the script. It is used as an input into **NI-845x SPI Extract Script Read Data.vi**.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Description

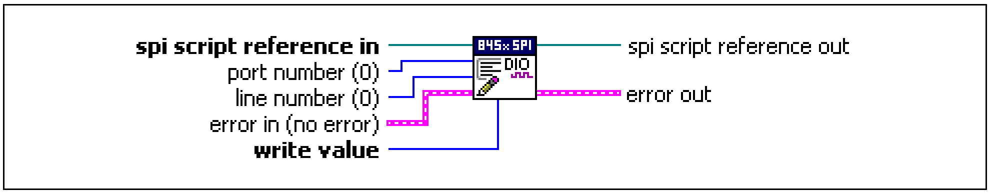
Use **NI-845x SPI Script DIO Read Port.vi** to add an SPI Script DIO Read Port command to an SPI script referenced by **spi script reference in**. This command allows you to read all 8 bits on a byte-wide DIO port. For NI 845*x* devices with multiple DIO ports, use the **port number** input to select the desired port. For NI 845*x* devices with one DIO port, **port number** must be left at the default (`0`).

To obtain the data byte read from the specified DIO port, wire **script read index** to **NI-845x SPI Extract Script Read Data.vi** after script execution, which returns the data byte read by this script command.

NI-845x SPI Script DIO Write Line.vi

Purpose

Adds an SPI Script DIO Write Line command to an SPI script referenced by **spi script reference in**. This command writes to a DIO line on an NI 845*x* device.



Inputs

spi script reference in is a reference to an SPI script that is run on an NI 845*x* device.

port number specifies the DIO port that contains the **line number**.

line number specifies the DIO line to write.

write value specifies the value to write to the line. **write value** uses the following values:

`0` (Logic Low)	The line is set to the logic low state.
`1` (Logic High)	The line is set to the logic high state.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

spi script reference out is a reference to the SPI script after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

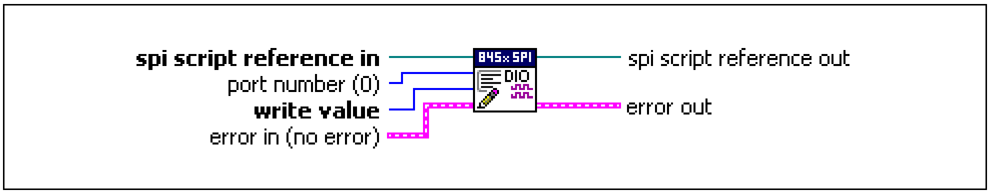
Description

Use **NI-845x SPI Script DIO Write Line.vi** to add an SPI Script DIO Write command to an SPI script referenced by **spi script reference in**. This command allows you to write one line, specified by **line number**, of a byte-wide DIO port. If **write value** is `1`, the specified line's output is driven to a high logic level. If **write value** is `0`, the specified line's output is driven to a low logic level. For NI 845*x* devices with multiple DIO ports, use the **port number** input to select the desired port. For NI 845*x* devices with one DIO port, **port number** must be left at the default (`0`).

NI-845x SPI Script DIO Write Port.vi

Purpose

Adds an SPI Script DIO Write Port command to an SPI script referenced by **spi script reference in**. This command writes to a DIO port on an NI 845*x* device.



Inputs

spi script reference in is a reference to an SPI script that is run on an NI 845*x* device.

port number specifies the DIO port to write.

write value is the value to write to the DIO port. Only lines configured for output are updated.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

spi script reference out is a reference to the SPI script after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

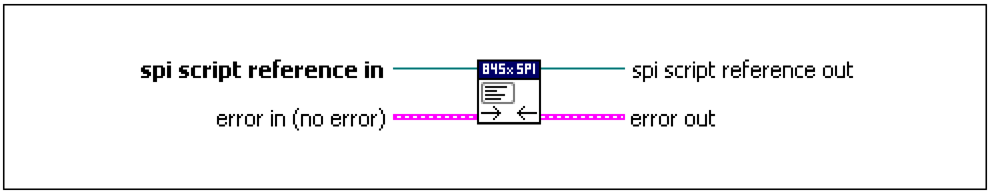
Description

Use **NI-845x SPI Script DIO Write Port.vi** to add an SPI Script DIO Write Port command to an SPI script referenced by **spi script reference in**. This command allows you to write all 8 bits on a byte-wide DIO port. For NI 845*x* devices with multiple DIO ports, use the **port number** input to select the desired port. For NI 845*x* devices with one DIO port, **port number** must be left at the default (`0`).

NI-845x SPI Script Disable SPI.vi

Purpose

Adds an SPI Script Disable SPI command to an SPI script referenced by **spi script reference in**. This command tristates the pins on an SPI port. It also tristates all chip select pins.



Inputs

spi script reference in is a reference to an SPI script that is run on an NI 845*x* device.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

spi script reference out is a reference to the SPI script after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute

the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

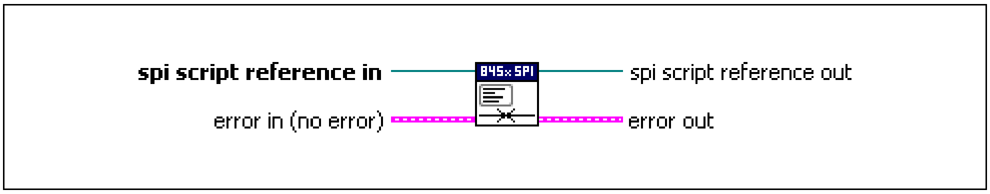
Description

Use **NI-845x SPI Script Disable SPI.vi** to add an SPI Script Disable SPI command to an SPI script referenced by **spi script reference in**. This command tristates the pins on the SPI port you specify when you use **NI-845x SPI Run Script.vi**. All chip select pins are also tristated.

NI-845x SPI Script Enable SPI.vi

Purpose

Adds an SPI Script Enable SPI command to an SPI script referenced by **spi script reference in**. This command switches the pins on an SPI port from tristate to master mode function. All chip select pins are switched from tristate to push-pull output driven high.



Inputs

spi script reference in is a reference to an SPI script that is run on an NI 845*x* device.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

spi script reference out is a reference to the SPI script after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Description

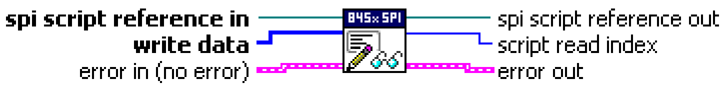
Use **NI-845*x* SPI Script Enable SPI.vi** to add an SPI Script Enable SPI command to an SPI script referenced by **spi script reference in**. This command switches the pins on the SPI port you specify when you use **NI-845x SPI Run Script.vi**, from tristate to master mode function.

Also, all chip select pins are switched from tristate to push-pull output driven high. It is important to keep this in mind if you are creating a script to access a device with an active high chip select input. You need to enable SPI and write the device chip select low until you want to access it, at which time you set the chip select high, perform the write/read, and then set the chip select low.

NI-845x SPI Script Write Read.vi

Purpose

Adds an SPI Script Write Read command to an SPI script referenced by **spi script reference in**. This command exchanges an array of data with an SPI slave device.



Inputs

spi script reference in is a reference to an SPI script that is run on an NI 845*x* device.

write data contains an array of data to write to the SPI slave.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of 0 means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

spi script reference out is a reference to the SPI script after this VI runs.

script read index is the index of the write/read command within the script. It is used as an input into **NI-845x SPI Extract Script Read Data.vi**.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Description

Use **NI-845x SPI Script Write Read.vi** to add an SPI Script Write Read command to an SPI script referenced by **spi script reference in**. This command exchanges an array of data with an SPI slave device connected to the SPI port you specify when you use **NI-845x SPI Run Script.vi** to execute the script.

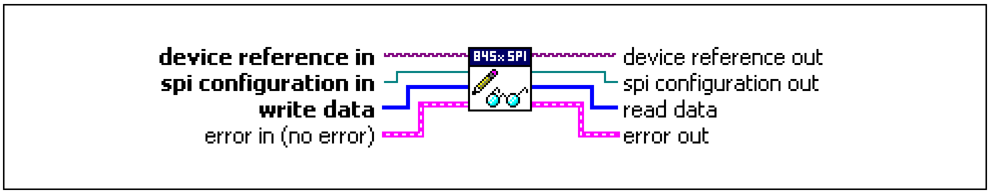
Due to the full-duplex nature of SPI, the size of the read data equals the size of the write data, unless there is an error. Some SPI devices act as receivers only and require one or more command and data bytes to be sent to them in one SPI transaction. As this is device specific, you need to review the device datasheet to package the required commands and data into the write data array. Other SPI devices act as transceivers. These devices can receive data much like receiver-only devices. But they can also transmit data, which usually requires writing one or more command bytes plus a number of bytes equal to the number of bytes desired to be read from the device. In most cases, the values of these bytes are not important, as they serve only to clock data out of the device. Here again, the SPI transaction formats are device specific, so you need to review the device datasheet to package the required commands and data into the write data array.

To obtain the data read from the specified SPI port, wire **script read index** to **NI-845x SPI Extract Script Read Data.vi** after script execution, which returns the data read by this script command.

NI-845x SPI Write Read.vi

Purpose

Exchanges an array of data with an SPI slave device.



Inputs

device reference in is a reference to an NI 845*x* device.

spi configuration in is a reference to a specific SPI configuration that describes the characteristics of the device to communicate with. Connect this configuration reference into a property node to set the specific configuration parameters.

write data contains an array of data to write to the SPI slave.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

device reference out is a reference to the NI 845*x* device after this VI runs.

spi configuration out is a reference to the SPI configuration after this VI runs.

read data contains an array of read data from an SPI interface.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Description

Use **NI-845x SPI Write Read.vi** to exchange an array of data with an SPI slave device. Due to the full-duplex nature of SPI, the size of the read data equals the size of the write data, unless there is an error. Some SPI devices act as receivers only and require one or more command and data bytes to be sent to them in one SPI transaction. As this is device specific, you need to review the device datasheet to package the required commands and data into the write data array. Other SPI devices act as transceivers. These devices can receive data much like receiver-only devices. But they can also transmit data, which usually requires writing one or more command bytes plus a number of bytes equal to the number of bytes desired to be read from the device. In most cases, the values of these bytes are not important, as they serve only to clock data out of the device. Here again, the SPI transaction formats are device specific, so you need to review the device datasheet to package the required commands and data into the write data array.

Before using **NI-845x SPI Write Read.vi**, you need to ensure that the configuration parameters specified in **spi configuration in** are correct for the device you currently want to access.

8

Using the NI-845*x* DIO API

This chapter helps you get started with the DIO API.

NI-845x DIO Basic Programming Model

When you use the DIO API, the first step is to configure the DIO port to be set for input or output as desired. Once the port is configured, you can write or read lines from the port. You can use either port or line I/O for all DIO calls. With the port calls, you can read or write all lines in a port at one time. Alternately, with the line calls, you can read or write the lines in a port one line at a time.

The diagram in Figure 8-1 describes the basic programming model for the NI-845x DIO API. Within the application, you repeat this basic programming model for each DIO call you need to make. The diagram is followed by a description of each step in the model.

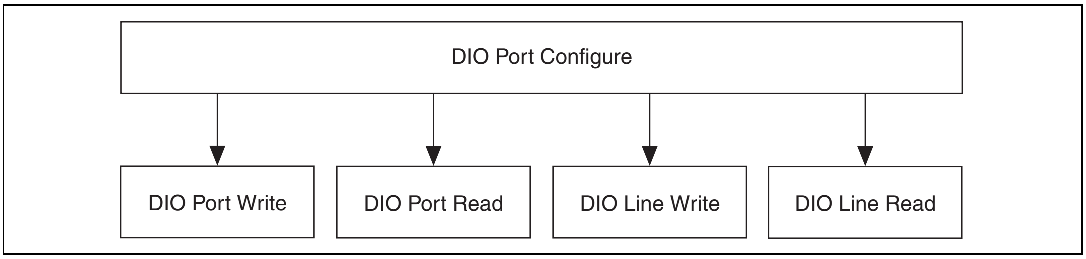


Figure 8-1. Basic Programming Model for DIO Communication

DIO Port Configure

The DIO Port configuration is set with the **NI-845x Device Property Node**. The **NI-845x Device Property Node** allows to you to set the following parameters for configuring the DIO Port:

- **Active DIO Port** is the active DIO port to configure. The subsequent property settings affect only the selected DIO port.
- **DIO Port Voltage** describes the voltage characteristics for the DIO port. Options include whether to use open-drain with external pull-ups or push-pull to drive 3.3 V on the DIO lines.

- **DIO Line Direction Map** indicates the direction (input or output) for each line in the 8-bit DIO port.

DIO Port Write

Use **NI-845x DIO Port Write.vi** to write an 8-bit pattern to the selected DIO port.

DIO Port Read

Use **NI-845x DIO Port Read.vi** to read an 8-bit pattern from the selected DIO port.

DIO Line Write

Use **NI-845x DIO Line Write.vi** to write a value to a particular line within the selected DIO port.

DIO Line Read

Use **NI-845x DIO Line Read.vi** to read a value from a particular line within the selected DIO port.

9

NI-845*x* DIO API for LabVIEW

This chapter lists the LabVIEW VIs for the NI-845*x* DIO API and describes the format, purpose, and parameters for each VI. The VIs in this chapter are listed alphabetically.

NI-845x Device Property Node

Purpose

A property node with the NI-845*x* Device class preselected. This property node allows you to modify properties of your NI 845*x* device.

Inputs

device reference in is a reference to an NI 845*x* device.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

device reference out is a reference to an NI 845*x* device after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is

returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Description

The list below describes all valid properties for the **NI-845x Device Property Node**.

Active DIO Port

The **Active DIO Port** property sets the active DIO port for further DIO port configuration. The format for this property is a decimal string. For example, the string `0` represents DIO Port 0. For NI 845*x* devices with one DIO port, the port value must be set to `0`.

DIO Port Voltage

The **DIO Port Voltage** property configures the active DIO port with the desired voltage characteristics. **DIO Port Voltage** uses the following values:

`Open-Drain`

The port is configured for open-drain voltage.

`Push-Pull 3.3 V`

The port is configured for 3.3 V push-pull voltage.

The default value of this property is `Push-Pull 3.3 V`.

DIO Line Direction Map

The **DIO Line Direction Map** property sets the line direction map for the active DIO Port. The value is a bitmap that specifies the function of each individual line within the port. If bit *x* = 1, line *x* is an output. If bit *x* = 0, line *x* is an input.

The default value of this property is `0` (all lines configured for input).

NI-845x Device Reference

Purpose

Specifies the device resource to be used for communication.

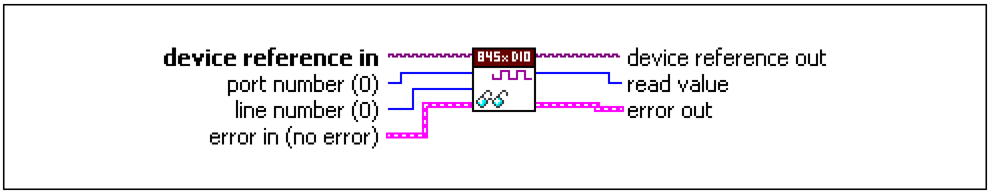
Description

Use the NI-845*x* Device Reference to describe the NI 845*x* device to communicate with. You can wire the reference into a property node to set specific device parameters or to an NI-845*x* API call to invoke the function on the associated NI 845*x* device.

NI-845x DIO Read Line.vi

Purpose

Reads from a DIO line on an NI 845*x* device.



Inputs

device reference in is a reference to an NI 845*x* device.

port number specifies the DIO port that contains the **line number**.

line number specifies the DIO line to read.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

device reference out is a reference to the NI 845*x* device after this VI runs.

read value is the value read from the line. **read value** uses the following values:

`0` (Logic Low)	The line read is in the logic low state.
`1` (Logic High)	The line read is in the logic high state.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

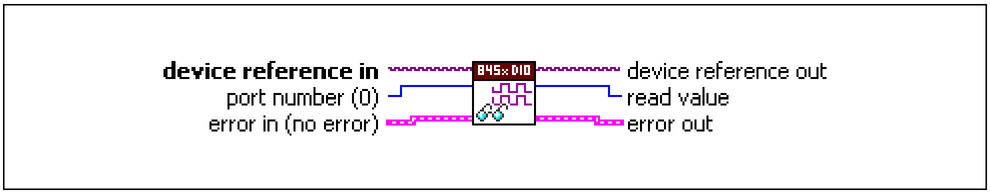
Description

Use **NI-845x DIO Read Line.vi** to read one line, specified by **line number**, of a byte-wide DIO port. For NI 845*x* devices with multiple DIO ports, use the **port number** input to select the desired port. For NI 845*x* devices with one DIO port, **port number** must be left at the default (`0`). If **read value** is `0`, the logic level read on the specified line was low. If **read value** is `1`, the logic level read on the specified line was high.

NI-845x DIO Read Port.vi

Purpose

Reads from a DIO port on an NI 845*x* device.



Inputs

device reference in is a reference to an NI 845*x* device.

port number specifies the DIO port to read.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

device reference out is a reference to the NI 845*x* device after this VI runs.

read value is the value read from the DIO port. If a DIO pin was previously configured for input, the logic level being driven onto it by external circuitry is returned. If a DIO pin was previously configured for output, the logic level driven onto the pin internally is returned. **read value** bit *n* = DIO *n*.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

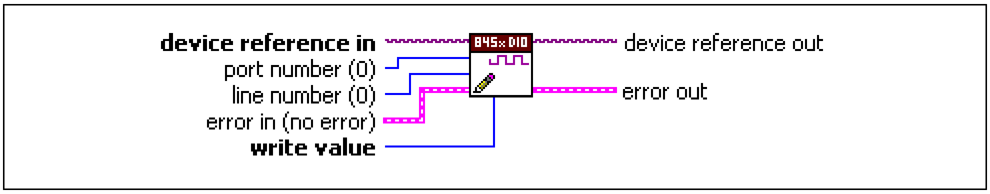
Description

Use **NI-845x DIO Read Port.vi** to read all 8 bits on a byte-wide DIO port. For NI 845*x* devices with multiple DIO ports, use the **port number** input to select the desired port. For NI 845*x* devices with one DIO port, **port number** must be left at the default (`0`).

NI-845x DIO Write Line.vi

Purpose

Writes to a DIO line on an NI 845*x* device.



Inputs

device reference in is a reference to an NI 845*x* device.

port number specifies the DIO port that contains the **line number**.

line number specifies the DIO line to write.

write value specifies the value to write to the line. **write value** uses the following values:

`0` (Logic Low)	The line is set to the logic low state.
`1` (Logic High)	The line is set to the logic high state.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

device reference out is a reference to the NI 845*x* device after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

status is TRUE if an error occurred.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

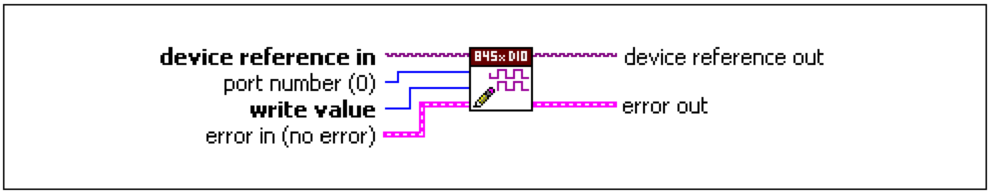
Description

Use **NI-845x DIO Write Line.vi** to write one line, specified by **line number**, of a byte-wide DIO port. If **write value** is `1`, the specified line's output is driven to a high logic level. If **write value** is `0`, the specified line's output is driven to a low logic level. For NI 845*x* devices with multiple DIO ports, use the **port number** input to select the desired port. For NI 845*x* devices with one DIO port, **port number** must be left at the default (`0`).

NI-845x DIO Write Port.vi

Purpose

Writes to a DIO port on an NI 845*x* device.



Inputs

device reference in is a reference to an NI 845*x* device.

port number specifies the DIO port to write.

write value is the value to write to the DIO port. DIO pins configured for input are not affected. If DIO *n* is configured for output, DIO *n* = **write value** bit *n*.

error in describes error conditions occurring before the VI executes. If an error has already occurred, the VI returns the value of the **error in** cluster in **error out**.

status is TRUE if an error occurred. This VI is not executed when status is TRUE.

code is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

source identifies the VI where the error occurred.

Outputs

device reference out is a reference to the NI 845*x* device after this VI runs.

error out describes error conditions. If the **error in** cluster indicated an error, the **error out** cluster contains the same information. Otherwise, **error out** describes the error status of this VI.

TF **status** is TRUE if an error occurred.

I32 **code** is the error code number identifying an error. A value of `0` means success. A negative value means error: VI did not execute the intended operation. A positive value means warning: VI executed intended operation, but an informational warning is returned. For a description of the **code**, wire the error cluster to a LabVIEW error-handling VI, such as the **Simple Error Handler**.

abc **source** identifies the VI where the error occurred.

Description

Use **NI-845x DIO Write Port.vi** to write all 8 bits on a byte-wide DIO port. For NI 845*x* devices with multiple DIO ports, use the **port number** input to select the desired port. For NI 845*x* devices with one DIO port, **port number** must be left at the default (`0`).

A Technical Support and Professional Services

Visit the following sections of the National Instruments Web site at `ni.com` for technical support and professional services:

- **Support**—Online technical support resources at `ni.com/support` include the following:
  - **Self-Help Resources**—For answers and solutions, visit the award-winning National Instruments Web site for software drivers and updates, a searchable KnowledgeBase, product manuals, step-by-step troubleshooting wizards, thousands of example programs, tutorials, application notes, instrument drivers, and so on.
  - **Free Technical Support**—All registered users receive free Basic Service, which includes access to hundreds of Application Engineers worldwide in the NI Developer Exchange at `ni.com/exchange`. National Instruments Application Engineers make sure every question receives an answer.

    For information about other technical support options in your area, visit `ni.com/services` or contact your local office at `ni.com/contact`.
- **Training and Certification**—Visit `ni.com/training` for self-paced training, eLearning virtual classrooms, interactive CDs, and Certification program information. You also can register for instructor-led, hands-on courses at locations around the world.
- **System Integration**—If you have time constraints, limited in-house technical resources, or other project challenges, National Instruments Alliance Partner members can help. To learn more, call your local NI office or visit `ni.com/alliance`.

If you searched `ni.com` and could not find the answers you need, contact your local office or NI corporate headquarters. Phone numbers for our worldwide offices are listed at the front of this manual. You also can visit the Worldwide Offices section of `ni.com/niglobal` to access the branch office Web sites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.

Glossary

Symbol	Prefix	Value
p	pico	10^{-12}
n	nano	10^{-9}
μ	micro	10^{-6}
m	milli	10^{-3}
k	kilo	10^{3}
M	mega	10^{6}
G	giga	10^{9}
T	tera	10^{12}

A

Arbitration	The procedure to allow multiple masters to determine which single master controls the bus for a particular transfer time.

C

CLK	CLocK. The clock is generated by the master device and controls when data is sent and read.
CPHA	Clock PHAse. This controls the positioning of the data bits relative to the clock edges.
CPOL	Clock POLarity. The polarity indicating whether the clock makes positive or negative pulses.
CS or SS	Chip Select or Slave Select. Connection from the master to a slave that signals the slave to listen for SPI clock and data signals.

I

I^2C	Inter-IC

M

Master	On the I^2C bus, a device that can initiate and terminate a transfer on the bus. The master is responsible for generating the clock (SCL) signal. On the SPI bus, the master device provides the clock signal and determines the chip select line state.
MISO	Master Input, Slave Output. The MISO carries data from the slave to the master.
MOSI	Master Output, Slave Input. The MOSI line carries data from the master to the slave.
Multimaster	The ability for more than one master to co-exist on the bus concurrently without data loss.

R

Receiver	Device receiving data from the bus.

S

SCL	Serial CLock (clock signal line).
SDA	Serial DAta (data signal line).
Shift Register	A shift register is connected to the MOSI and MISO lines. As data is read from the input, it is placed into the shift register. Data from the shift register is placed into the output, creating a full-duplex communication loop.
Slave	On the I^2C bus, a device addressed by the master. On the SPI bus, the slave device receives the clock and chip select from the master. The maximum number of slaves is dependent on the number of available chip select lines.
SMBus	System Management Bus

Synchronization	The defined procedure to allow the clock signals provided by two or more masters to be synchronized.

T

Transmitter	Device transmitting data on the bus.

Index