NI-Digital Pattern User Manual

2025-03-26

Contents

NI-Digital Pattern User Manual

The NI-Digital Pattern User Manual provides detailed descriptions of the product functionality and the step by step processes for use.

Looking for Something Else?

For information not found in the User Manual for your product, such as specifications and API reference, browse ***Related Information***.

Related information:

- NI-Digital Pattern Driver Release Notes
- NI-Digital Pattern Driver LabVIEW Reference
- NI-Digital Pattern Driver .NET Class Library
- NI-Digital Pattern Driver C Function Reference
- License Setup and Activation
- Software and Driver Downloads
- NI Learning Center

What is NI-Digital Pattern?

Use the Digital Pattern Editor with a Digital Pattern Instrument and the NI-Digital Pattern Driver software for digital testing of semiconductors, or devices under test (DUTs).

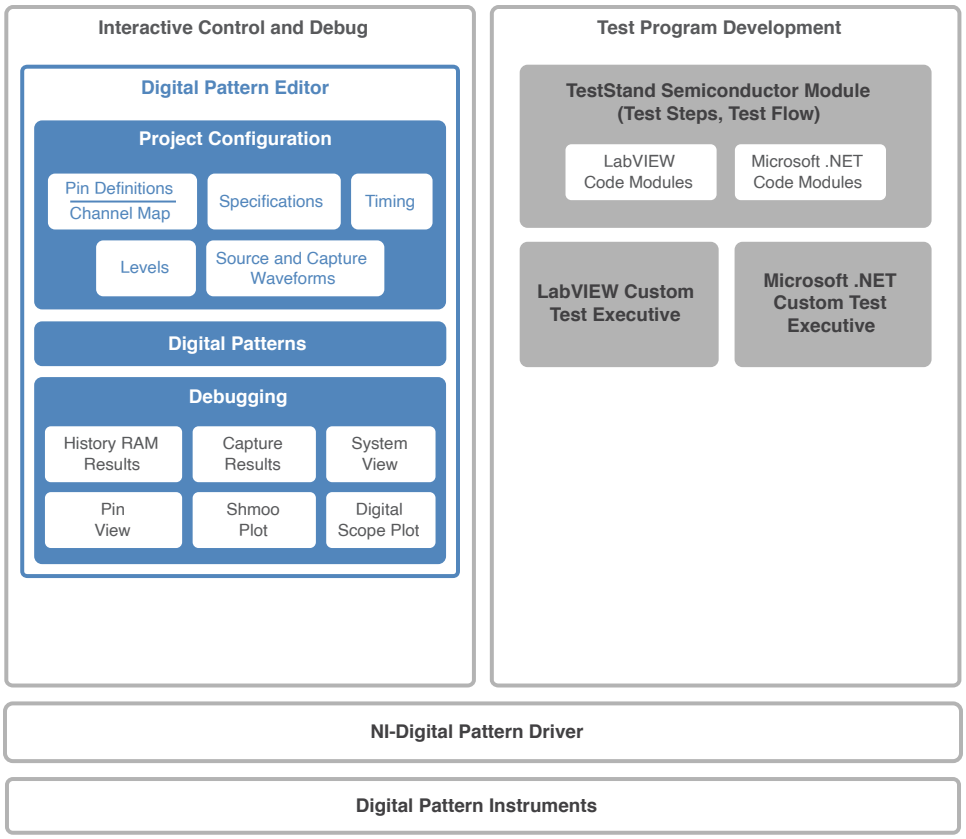
Key Features

The following features set NI-Digital Pattern apart:

- Intuitive interfaces for importing, editing, and creating test patterns.
- Development tools to create test code and interact with digital pattern instruments in LabVIEW, C, or .NET.
- Debugging tools for digital test patterns, such as overlaying pattern failures and digital scope.
- Integrated editors to develop imported digital test vectors and patterns.
- Support for native pin map and multisite, DUT-centric programming of Semiconductor Test Systems (STS).

Components of an NI-Digital Pattern System

The following figure displays the relationship between test programs, digital pattern project files, and test hardware.



This documentation assumes you are an experienced semiconductor test engineer familiar with the following:

- Common semiconductor test program hardware and software components.
- Digital pattern project components.
- Programming in LabVIEW or Microsoft Visual Studio.

Supported Hardware

NI-Digital Pattern supports the following hardware models:

- PXIe-6570
- PXIe-6571

Related reference:

- Getting Started with Digital Pattern Instruments
- NI-Digital Pattern Driver Examples
- Digital Pattern Editor
- Programming with the NI-Digital Pattern Driver

Related information:

- PXIe-6570 Specifications
- PXIe-6571 Specifications

New Features and Changes

Learn about updates—including new features and behavior changes—introduced in each version of NI-Digital Pattern.

Discover what's new in the latest releases of NI-Digital Pattern.

> **Note** If you cannot find new features and changes for your version, it may not include user-facing updates. However, your version may include non-visible changes such as bug fixes and compatibility updates. For information about non-visible changes, refer to your product ***Release Notes*** on ni.com.

NI-Digital Pattern 2024 Q3 Changes

General updates to the Digital Pattern Editor:

- View and clear NI-DCPower real-time alarms.
- View memory use by patterns and loaded waveforms.
- Added support for custom relays driven by NI-DAQmx digital lines.
- Added new advanced SMU properties.

NI-Digital Pattern 2024 Q2 Changes

Hardware support for the following instrument model:

- PXIe-6571 (8CH)

General updates to the Digital Pattern Editor System View:

- View the time sets currently loaded on the digital instruments.
- Support for MI & RF drivers to display NI-DMM, NI-SCOPE, NI-FGEN, NI-RF, and NI-DAQmx pins.

Updates to Digital Pattern Editor register maps:

- Edit DUT register maps directly in the Digital Pattern Editor.
- Use the **Import Semiconductor Device Control register map** command to import a

Semiconductor Device Control (SDC) register map.

Updates to Digital Pattern Editor advanced debugging tools:

- In System View, use the **Go to InstrumentStudio** command on source measure unit (SMU), data acquisition, digital multimeter, radio frequency (RF), oscilloscope, and waveform generator pins to launch InstrumentStudio. In InstrumentStudio, you can view, configure, monitor, and debug items such as pins, channels, or instruments.
- In System View, use the **Go to Pin View** command on Digital and DCPower pins to navigate to the Pin View pane and configure that pin.
- In Project Files View, use the **Go to InstrumentStudio** command on a pin map to launch InstrumentStudio configured for that pin map.

Related information:

- PXIe-6571 Specifications

NI-Digital Pattern 2024 Q1 Changes

Updates to the Digital Pattern Editor:

- In Pattern Grid View, delete multiple pin columns.
- In Pattern Grid View, represent selected pin state cell values as numeric values.
- In System View, apply relay configurations.
- In various views, use the **Open Pattern** button next to the **Pattern Name** text box to open the pattern.
- In a timing document, visually group time set rows.

NI-Digital Pattern 2023 Q4 Changes

Updates to the Digital Pattern Editor:

- In Pattern Grid View, navigate using the `Page Up` and `Page Down` keys.
- In Pattern Grid View, scroll pin and comment columns horizontally.
- Update to the Find in Pattern dialog box to become non-modal.

NI-Digital Pattern 2023 Q3 Changes

Hardware support for the following instrument models:

- PXIe-4051
- PXIe-4151

Related information:

- PXIe-4051 Specifications
- PXIe-4151 Specifications

NI-Digital Pattern 2023 Q2 Changes

NI-Digital Pattern 2023 Q2 is a maintenance release and does not include any new user-facing features or behavior changes.

NI-Digital Pattern 2023 Q1 Changes

- Added support for LabVIEW 2023 and later.

NI-Digital Pattern 2022 Q4 Changes

NI-Digital Pattern 2022 Q4 includes only bug fixes. Refer to the ***NI-Digital Pattern Driver Release Notes*** for a list of bug fixes.

Related information:

- NI-Digital Pattern Driver Release Notes

NI-Digital Pattern 21.8 Changes

Updates to the Digital Pattern Editor:

- Improved performance for System View measurements.
- Improved performance for creating DC-Power sessions.
- In digital scope, halt pattern execution at a specified vector.

- In Pattern Grid View, display the burst count.
- Save shmoo test results to a text file.

NI-Digital Pattern 21.0 Changes

Hardware support for the following instrument models:

- PXIe-4135 (35W)
- PXIe-4137 (35W)

Related information:

- PXIe-4135 Specifications
- PXIe-4137 Specifications

Getting Started with Digital Pattern Instruments

Use the getting started examples, located by default in the `<Public Documents>\National Instruments\NI-Digital\Examples\Getting Started` directory, help you learn key concepts. You can also access the getting started examples by selecting the **Start** menu and then navigating to NI Digital Pattern Examples in the National Instruments folder.

Additionally, the Digital Pattern Editor includes examples for getting started. In the Welcome window, click **Examples** or click the **Learning** tab to launch the Learning window, which includes examples of common usages of digital patterns.

Use the NI-Digital Pattern Driver examples to learn more about how to use the NI-Digital Pattern Driver software to configure and control digital pattern instruments. Use the NI Example Finder to locate LabVIEW and LabVIEW NXG examples. The Microsoft .NET examples are located in the `<Public Documents>\National Instruments\NI-Digital\Examples\DotNET 4.x` directory. The C examples are located in the `<Public Documents>\National Instruments\NI-Digital\Examples\C` directory.

Related reference:

- Getting Started - SPI Example
- Getting Started - Continuity Measurements Example
- Getting Started - Leakage Measurements Example
- NI-Digital Pattern Driver Examples

Getting Started - SPI Example

Purpose

This example demonstrates how to burst a Serial Peripheral Interface (SPI) command at a 1MHz vector rate from the Digital Pattern Editor on a Digital Pattern Instrument using the NI-Digital Pattern Driver software in LabVIEW, Microsoft .NET, and C. You can

use this example to help you learn key concepts or to serve as a starting point for applications you create.

> **Note** For this example to produce passing results, you must connect digital channels 2 and 3 together (pins MOSI and MISO) and you must manually make a loopback connection to a Digital Pattern Instrument. The example runs without the loopback connection but reports failures on each test step. NI does not provide a DUT for this example.

Example File Location

```
<Public Documents>\National Instruments\NI-Digital\
Examples\Getting Started
```

Highlighted Features

Digital Pattern Editor

NI-Digital Pattern Driver

Major API

NI-Digital Pattern Driver

Prerequisites

> **Note** For this example to produce passing results, you must connect digital channels 2 and 3 together (pins MOSI and MISO) and you must manually make a loopback connection to a Digital Pattern Instrument. The example runs without the loopback connection but reports failures on each test step. NI does not provide a DUT for this example.

How to Use This Example

Complete the following steps before you review the Digital Pattern Editor components and then use LabVIEW, Microsoft .NET, or C to burst a pattern and view the results.

1. Manually connect the digital pattern instrument channels 2 and 3 together (pins MOSI and MISO).
2. In the **Start** menu, navigate to **NI Digital Pattern Editor** in the **National Instruments** folder to launch the Digital Pattern Editor.

Review Components

1. Open the `SPI Example.digiproj` project, located in the appropriate development environment folder in the `<Public Documents>\National Instruments\NI-Digital\Examples\Getting Started` directory.
2. Review the list of files in the Project Explorer window, such as the pin and channel map, specifications, digital pin levels, digital time sets, digital pattern, and Shmoo plot. You can use folders to organize and manage files in a project. Double-click the file in the Project Explorer window to open it.
3. Review the options in the Pin View pane to the right of the document workspace. You can use the Pin View pane in conjunction with the pattern document during debugging to interactively view and modify the current state of the pin driver, active load, comparators, PPMU, and NI-DCPower instrument settings for a single pin at a time.
4. Double-click the `SPI Example.pinmap` file in the Project Explorer window to launch the pin and channel map editor, which displays an editable hierarchical view of the Instruments, Pins, Pin Groups, Sites, and Connections elements in the pin and channel map file.
5. On the Pin Map tab, select the default **PXI1Slot2 {Digital Pattern}** entry in the `<Instruments>` section and change the corresponding **Name** field to match the name of the digital pattern instrument resource in your system.
6. Double-click the `Specifications.specs` file in the Project Explorer window to launch the specifications document, which you can use to define a set of variables and associated numeric values that you can reference in levels, timing, and other specifications files. Review the `vcc` and `period` variables.
7. Double-click the `PinLevels.digilevels` file in the Project Explorer window to launch the levels document, which displays voltage levels and a description for digital pins and pin groups defined in the pin and channel map.
8. Click the **Apply Levels** button to load the levels values on the instrument.
9. Double-click the `Timing.digitiming` file in the Project Explorer window to launch the timing document, which displays components of the time sets, including the format and edge placement that shape the digital waveform on a

per-pin basis.

10. Click the **Apply Time Sets** button to load the time set values on the instrument.

Burst Pattern and View Results

1. Double-click the `SPI Pattern.digipat` file in the Project Explorer window to launch the pattern document grid view, which displays components of the binary pattern file, including time sets, labels, opcodes, vector identifiers, data to send to or received from pins and pin groups, and comments for each vector.
2. Click the **Load Pattern** button to load the pattern on the instrument.
3. Click the **Burst** button to burst the pattern on the instrument. If necessary, the editor prompts you to load the pattern, time set values, and levels values on the instrument. The **Passed**/**Failed**/**Disabled** indicator displays whether the last pattern burst includes any failures, regardless of what you log to History RAM, and whether the site was disabled for the last burst of the pattern. Disabled sites do not return results. The **Total Failures** indicator displays the number of failures in the pattern. The **Burst Complete** indicator displays the timestamp of when the last pattern burst completed. Pin state data cells with a red background and red text indicate a failure.
4. Double-click the `Shmoo.digishmoo` file in the Project Explorer window to launch the configured Shmoo plot, which displays a dynamically updated intensity plot of pass and fail values for two variables you specify using the specifications, timing, levels, pin and channel map, and pattern files in the project.
5. Click the **Run Shmoo** button to start the operation and view the results.
6. Select **Instruments** » **Disconnect** to disconnect the instrument session.

Run the LabVIEW Example

1. Open the SPI Example VI.
2. On the block diagram, review the comments, which explain each subVI. The VI calls the same pin and channel map, specifications, levels, timing, and pattern files referenced in the `SPI Example.digiproj` project you reviewed in the digital pattern editor.
3. On the front panel, select the digital pattern instrument resource in your system in the **Resource Name** control.
4. Run the VI.

Run the Microsoft .NET Example

1. Open the `NIDigital.SPIExample.sln` solution file in Microsoft Visual Studio.
2. In `MainForm.cs`, review the comments, which explain the NI-Digital Pattern Driver .NET API calls. The code uses the same pin and channel map, specifications, levels, timing, and pattern files referenced in the `SPI Example.digiproj` project you reviewed in the digital pattern editor.
3. Build and run the solution.
4. On the main form, enter the name of the digital pattern instrument resource in your system in the **Digital Instrument** text box.
5. Click the **Run Test** button.

Run the C Example

1. Open the `NIDigital.SPIExample.sln` solution file in Microsoft Visual Studio.
2. In `SPI Example.c`, review the comments, which explain the NI-Digital Pattern Driver C API calls. The code uses the same pin and channel map, specifications, levels, timing, and pattern files referenced in the `SPI Example.digiproj` project you reviewed in the digital pattern editor.
3. In the `main` function, enter the name of the digital pattern instrument resource in your system as the initialization value for the variable deviceID.
4. Build and run the solution.

Getting Started - Continuity Measurements Example

Purpose

This example demonstrates how to perform PPMU continuity measurements with a Digital Pattern Instrument using the NI-Digital Pattern Driver software in LabVIEW, Microsoft .NET, and C. You can use this example to help you learn key concepts or to serve as a starting point for applications you create.

Note For this example to produce passing results, you must connect an appropriate DUT to a Digital Pattern Instrument and configure a pin and channel map file for the DUT. The example runs without a DUT but reports

failures on each test step. NI does not provide a DUT for this example.

Most CMOS devices include bidirectional ESD protection diodes on digital pins. You can use these diodes to check whether the pin is connected properly to the die without an internal open or a short. This example forces a small positive current to the pin and expects that the voltage drop on the pin matches the forward voltage of the high side protection diode. Reverse the polarity of the current to exercise the other diode.

Example File Location

```
<Public Documents>\National Instruments\NI-Digital\
Examples\Getting Started
```

Highlighted Features

- Digital Pattern Editor
- NI-Digital Pattern Driver

Major API

NI-Digital Pattern Driver

Prerequisites

Note For this example to produce passing results, you must connect an appropriate DUT to a Digital Pattern Instrument and configure a pin and channel map file for the DUT. The example runs without a DUT but reports failures on each test step. NI does not provide a DUT for this example.

How to Use This Example

Complete the following steps before you review or modify the pin and channel map in the digital pattern editor and run the test in LabVIEW, Microsoft .NET, or C.

1. If you are using a DUT, manually connect the DUT to the digital pattern instrument.
2. In the **Start** menu, navigate to **NI Digital Pattern Editor** in the **National Instruments** folder to launch the Digital Pattern Editor.

Review the Pin Map

1. Open the `Continuity Example.digiproj` project, located in the appropriate development environment folder in the `<Public Documents>\National Instruments\NI-Digital\Examples\Getting Started` directory.
2. Double-click the `PinMap.pinmap` file in the Project Explorer window to launch the pin and channel map editor, which displays an editable hierarchical view of the Instruments, Pins, Pin Groups, Sites, and Connections elements in the pin and channel map file. This example pin and channel map includes a 2 input (A1, A2), 2 output (Y1, Y2) digital device with a power supply pin (VCC) also connected to the digital pattern instrument.
3. On the Pin Map tab, select the default **PXI1Slot2 {Digital Pattern}** entry in the `<Instruments>` section and change the corresponding **Name** field to match the name of the digital pattern instrument resource in your system.
4. Notice that the pins are organized into pin groups.
5. Modify the pins, sites, and connections to match the DUT you are using. Assign the pins to the predefined pin groups because the example code uses these pin group names to configure the measurements.

Run the LabVIEW Example

1. Open the Continuity VI.
2. On the block diagram, review the comments, which explains each subVI. The VI loads the same pin and channel map referenced in the `Continuity Example.digiproj` project you reviewed in the digital pattern editor.
3. On the front panel, select the digital pattern instrument resource in your system in the **Resource Name** control.
4. (Optional) Modify the test parameters to match the DUT you are using.
5. Run the VI.

Run the Microsoft .NET Example

1. Open the `NIDigital.ContinuityExample.sln` solution file in Microsoft Visual Studio.
2. In `MainForm.cs`, review the comments, which explain the NI-Digital Pattern Driver .NET API calls. The code uses the same pin and channel map referenced in

the `Continuity Example.digiproj` project you reviewed in the digital pattern editor.
3. Build and run the solution.
4. On the main form, enter the name of the digital pattern instrument resource in your system in the **Digital Instrument** text box.
5. Click the **Run Test** button.

Run the C Example

1. Open the `Continuity.sln` solution file in Microsoft Visual Studio.
2. In `Continuity.c`, review the comments, which explain the NI-Digital Pattern Driver C API calls. The code uses the same pin and channel map, specifications, levels, timing, and pattern files referenced in the `Continuity Example.digiproj` project you reviewed in the digital pattern editor.
3. In the `main` function, enter the name of the digital pattern instrument resource in your system as the initialization value for the variable deviceID.
4. Build and run the solution.

Getting Started - Leakage Measurements Example

Purpose

This example demonstrates how to perform input leakage measurements with the PPMU function of the Digital Pattern Instrument (PXIe-6570 or PXIe-6571) using the NI-Digital Pattern Driver software in LabVIEW, Microsoft .NET, and C. You can use this example to help you learn key concepts or to serve as a starting point for applications you create.

Note For this example to produce passing results, you must connect an appropriate DUT to a digital pattern instrument and configure a pin and channel map file for the DUT. The example runs without a DUT but reports false results on each test step. NI does not provide a DUT for this example.

Ideally, input pins have an infinite input impedance. In reality, however, the input impedance is non-infinite, which causes small amounts of current to leak into or out from the pin. This example forces a series of voltages to selected digital input pins of the DUT and measures the leakage current for each pin.

Example File Location

```
<Public Documents>\National Instruments\NI-Digital\
Examples\Getting Started
```

Highlighted Features

- Digital Pattern Editor
- NI-Digital Pattern Driver

Major API

NI-Digital Pattern Driver

Prerequisites

Note For this example to produce passing results, you must connect an appropriate DUT to a digital pattern instrument and configure a pin and channel map file for the DUT. The example runs without a DUT but reports false results on each test step. NI does not provide a DUT for this example.

How to Use This Example

Complete the following steps before you review or modify the pin and channel map in the digital pattern editor and run the test in LabVIEW, Microsoft .NET, or C.

1. If you are using a DUT, manually connect the DUT to a digital pattern instrument.
2. In the **Start** menu, navigate to **NI Digital Pattern Editor** in the **National Instruments** folder to launch the Digital Pattern Editor.

Review the Pin Map

1. Open the `Leakage Example.digiproj` project, located in the appropriate development environment folder in the `<Public Documents>\National Instruments\NI-Digital\Examples\Getting Started` directory.
2. Double-click the `PinMap.pinmap` file in the Project Explorer window to launch the pin and channel map editor, which displays an editable hierarchical view of the

Instruments, Pins, Pin Groups, Sites, and Connections elements in the pin and channel map file. This example pin and channel map includes a 2 input (A1, A2), 2 output (Y1, Y2) digital device with a power supply pin (VCC) also connected to the digital pattern instrument.

3. On the Pin Map tab, select the default **PXI1Slot2 {Digital Pattern}** entry in the `<Instruments>` section and change the corresponding **Name** field to match the name of the digital pattern instrument resource in your system.
4. Notice that the pins are organized into pin groups.
5. Modify the pins, sites, and connections to match the DUT you are using and update the pattern accordingly. Assign the pins to the predefined pin groups because the example code uses these pin group names to configure the measurements.

Run the LabVIEW Example

1. Open the Leakage VI.
2. On the block diagram, review the comments, which explains each subVI. The VI loads the same pin and channel map referenced in the `Leakage Example.digiproj` project you reviewed in the digital pattern editor.
3. On the front panel, select a digital pattern instrument resource in your system in the **Resource Name** control.
4. (Optional) Modify the test parameters to match the DUT you are using.
5. Run the VI.

Run the Microsoft .NET Example

1. Open the `NIDigital.LeakageExample.sln` solution file in Microsoft Visual Studio.
2. In `MainForm.cs`, review the comments, which explain the NI-Digital Pattern Driver .NET API calls. The code uses the same pin and channel map referenced in the `Leakage Example.digiproj` project you reviewed in the digital pattern editor.
3. Build and run the solution.
4. On the main form, enter the name of a digital pattern instrument resource in your system in the **Digital Instrument** text box.
5. Click the **Run Test** button.

Run the C Example

1. Open the `Leakage.sln` solution file in Microsoft Visual Studio.
2. In `Leakage.c`, review the comments, which explain the NI-Digital Pattern Driver C API calls. The code uses the same pin and channel map, specifications, levels, timing, and pattern files referenced in the `Leakage Example.digiproj` project you reviewed in the digital pattern editor.
3. In the `main` function, enter the name of the digital pattern instrument resource in your system as the initialization value for the variable deviceID.
4. Build and run the solution.

NI-Digital Pattern Driver Examples

Use the NI-Digital Pattern Driver examples to learn more about how to use the NI-Digital Pattern Driver software to configure and control digital pattern instruments.

LabVIEW and LabVIEW NXG Examples

Use the NI Example Finder to locate LabVIEW and LabVIEW NXG examples. In LabVIEW, select **Help** » **Find Examples** to launch the NI Example Finder. Find examples specific to NI-Digital by selecting **Hardware Input and Output** » **Modular Instruments** » **NI-Digital Pattern Driver**.

Microsoft .NET Examples

The Microsoft .NET examples are located in the `<Public Documents>\National Instruments\NI-Digital\Examples\DotNET 4.x` directory. In the **Start** menu, navigate to **NI Digital Pattern Examples** in the **National Instruments** folder.

C Examples

The C examples are located in the `<Public Documents>\National Instruments\NI-Digital\Examples\C` directory. In the **Start** menu, navigate to **NI Digital Pattern Examples** in the **National Instruments** folder.

Related reference:

- Getting Started with Digital Pattern Instruments

Digital Pattern Editor

Use Digital Pattern Editor to view, create, modify, and debug pin and channel maps, specifications, levels, timings, patterns, register maps, source waveforms, and capture waveforms. You can also use Digital Pattern Editor to configure the state of the digital pattern instrument.

Use the components of the editor to perform the following tasks:

- **Pin and channel map editor**—Assign DUT pin names to instrument channels and create pin groups with support for multiple test sites.
- **Specifications**—Define specification variables to use in levels, timing, and other files.
- **Levels**—Define levels for digital and power pins and pin groups.
- **Timing**—Define format and edge placement to shape digital patterns on a per-pin basis.
- **Patterns**—Burst digital patterns and display per-site, per-pin results.
- **Pin View pane**—Perform parametric voltage and current source and measure operations.
- **Shmoo plots**—Sweep parameters while repeatedly bursting a pattern and plot pass/fail results as Shmoo plots.
- **Digital scope**—Capture and display the analog waveform to inspect actual electrical behavior on pins at a particular point in a pattern.
- **Source and capture waveforms**—Configure waveform files to use in source and capture operations.
- **Register maps**—View and modify DUT register maps to display or change information stored in register memory.
- **TDR**—Measure and display time domain reflectometry values, and save measurements on a per-channel basis.

Related tasks:

- Levels

Related reference:

- Pin and Channel Map Editor
- Specifications
- Timing
- Patterns
- Pin View Pane
- Shmoo Plot
- Digital Scope
- Source Waveform Configurations
- Capture Waveform Configurations
- Getting Started with Digital Pattern Instruments
- NI-Digital Pattern Driver Examples
- Programming with the NI-Digital Pattern Driver

Getting Started with Digital Pattern Editor

Digital Pattern Editor is an application for testing and de-bugging pattern-based digital devices. In the **Start** menu, navigate to **NI Digital Pattern Editor** in the **National Instruments** folder to launch the Digital Pattern Editor.

Related reference:

- Creating and Opening Projects
- Accessing Examples

Creating and Opening Projects

Use the following methods to create or open a project:

- Select **File** » **New** to create a project or **File** » **Open Project** to open a project.
- In the Welcome window, click **Launch a Project** or click the Projects tab to display the Projects window, in which you can create a new project, select an existing project in the **Recent** list, or browse to an existing project.

Related reference:

- Interacting with Instruments

Accessing Examples

Use one of the following methods to access examples:

- In the Welcome window, click **Examples** or click the Learning tab to launch the Learning window, which includes examples of common usages of digital patterns.
- Use the getting started examples, located by default in the `<Public Documents>\National Instruments\NI-Digital\Examples\Getting Started` directory, to help you learn key concepts or to serve as a starting point for applications you create.
- Use the NI-Digital Pattern Driver examples to learn more about how to use the NI-Digital Pattern Driver software to configure and control digital pattern instruments.Use the NI Example Finder to locate LabVIEW and LabVIEW NXG examples.The Microsoft .NET examples are located in the `<Public Documents>\National Instruments\NI-Digital\Examples\DotNET 4.x` directory.The C examples are located in the `<Public Documents>\National Instruments\NI-Digital\Examples\C` directory.

Related reference:

- NI-Digital Pattern Driver Examples
- Getting Started with Digital Pattern Instruments

Digital Pattern Editor Menus

The Digital Pattern Editor environment includes the following menu items:

- **File** — Manage project files.
- **Edit** — Complete common editing operations.
- **Run** — Burst or abort pattern executions, Shmoo operations, or digital scope operations, depending on the active document.
- **Instruments** — Manage instrument sessions, synchronize multiple instruments, enable or disable sites, configure instrument settings, reset instrument sessions, or display pin or system information. Refer to the ***Related reference*** section at the end of this topic for a link to more information about interacting with instruments.
- **View** — Manage workspace pane display options.

- **Help** — Launch the ***Digital Pattern Editor Help***, context-sensitive help topics, or application and driver example help topics.

Related reference:

- Interacting with Instruments

Managing Projects

Create a project file (`.digiproj`) to organize and access digital configuration and pattern files, which you can double-click to open in the corresponding document window. Use the Project Explorer window toolbar buttons and right-click context menu items to manage the files and display options for the project.

Active Files

You can specify an active pin and channel map file, timing file, and levels file by right-clicking the file and selecting **Make Active** or **Make Inactive** to activate or deactivate a file. Files listed in bold in the Project Explorer window are the active files of the project. Only one pin and channel map file, one timing file, and one levels file can be active at a time. Changing the state of an inactive file to active automatically changes the state of the previously active file of the same file type to inactive. If you do not manually activate a file, the editor automatically activates the first instance of each file type you add to the project.

When you switch to a different active pin and channel map file, the editor disconnects any current instrument sessions and connects to existing open instrument sessions associated with the new active pin and channel map file.

Typically when you burst a pattern, the editor prompts you to apply the values in the active timing file or levels file when no timing or levels values have previously been applied to the digital pattern instrument or when previously applied timing or levels values change directly in the timing or levels document or indirectly through changes in the specifications document. Click the **Apply Time Sets** button in the active timing document or the **Apply Levels** button in the active levels document to manually apply timing and levels values from the active file and overwrite any previously applied values.

File Indicators

An asterisk (*) next to a filename in the Project Explorer window and in the document tab indicates that the file has been modified but not saved. The white arrow overlay on the file type icon  indicates a reference to a file that does not reside in or underneath the project directory. Use the Exclude and Include context menu items to change which files in the project directory the project uses. Excluding a file from the project leaves the file on disk. Deleting a file from the project deletes the file from disk.

File Types

A project can contain the following types of files. Refer to the ***Related reference*** section at the end of this topic for a link to more information about each of these components.

- Pin and channel maps (`.pinmap`)
- Specifications (`.specs`)
- Digital timing (`.digitiming`)
- Digital and power levels (`.digilevels`)
- Digital patterns (`.digipat`)
- Source waveform configurations (`.tdms`)

> **Note** Do not include other types of files with the `.tdms` file extension in the project.

- Capture waveform configurations (`.digicapture`)
- Shmoo plots (`.digishmoo`)

You can store digital project files (`.digiproj`) and the file types in the previous list in a third-party source code control (SCC) system. However, do not include the `.digiprojcache` file in source code control because it contains per-user settings and customizations that you typically do not share among users.

Because pin and channel maps, specifications, timing, and levels files are XML-based files, you can write custom scripts or use an in-house or third-party tool to convert existing files to the NI file formats.

Note When you use Windows Explorer to add or remove files from outside the Digital Pattern Editor, the Project Explorer window does not automatically refresh the list of files that are in the project directory but not yet included in the project. You must click the **Refresh** button on the Project Explorer window toolbar for the view to refresh. When you select **File » Add File** or right-click a file and select **Delete** from the context menu in the Digital Pattern Editor, the Project Explorer window refreshes automatically.

Related tasks:

- Levels

Related reference:

- Pin and Channel Map Editor
- Specifications
- Timing
- Patterns
- Source Waveform Configurations
- Capture Waveform Configurations
- Shmoo Plot
- Pin Values
- Levels File XML Structure

Related information:

- NI TDMS File Format

Workspace Configuration

The editor displays the Project Explorer window to the left of the main document window, which displays in the middle of the default workspace. The default configuration displays the Pin View pane and the Instrument Settings pane to the right of the main document window.

You can drag, dock, float, and resize documents in the document window, and you can resize or collapse the Project Explorer window, Pin View pane, and Instrument Settings

pane to configure a custom workspace layout to optimize development and debugging tasks. As you drag the document from the current location, the editor detaches the document and displays docking guides. The docking guides show you where you can place the document. As you move the mouse over a docking guide, the editor highlights where the document will relocate when dropped. The editor saves the workspace layout configuration in the `.digiprojcache` file when you save and close the project and restores the layout when you reopen the project. Select **View** » **Reset Workspace** to restore the workspace to a single document window with tabs for each open document.

You can drag a document by its tab away from the main workspace window and drop the document without highlighting a docking guide location to detach the document from the main workspace and create a new instance of the application window. All instances of the application window refer to the same project file and share the same state.Selecting **View** » **Reset Workspace** in a window in this case closes all the other windows.

Interacting with Instruments

The Digital Pattern Editor automatically connects to open instrument sessions for all available digital pattern instruments defined in the active pin and channel map file for a project when you take any of the following actions:

- Open a digital pattern project that includes an active pin and channel map file.
- Edit the active pin and channel map file.
- Switch to a different active pin and channel map file in the project.
- Select the **Instruments** » **Connect** menu item.

When you switch to a different active pin and channel map file, the editor disconnects any current instrument sessions and connects to existing open instrument sessions associated with the new active pin and channel map file.

When the editor cannot find open instrument sessions, it launches the Connect to Instruments dialog box. Refer to the ***Related reference*** section at the end of this topic for a link to more information about this dialog box.

Instruments Menu

Use the following **Instruments** menu items to manage instrument sessions, synchronize multiple instruments, run and apply TDR measurements, enable or disable sites, configure instrument settings, reset instrument sessions, or display pin or system information.

- **Connect** — Specifies to connect to open instrument sessions for all available digital pattern instruments, NI-DCPower instruments, and relay driver modules defined in the active pin and channel map file for the project. When the editor cannot find open instrument sessions, it launches the Connect to Instruments dialog box. This option is dimmed when no project is open.
- **Disconnect** — Specifies to disconnect any existing connections to instrument sessions for all digital pattern instruments, NI-DCPower instruments, and relay driver modules defined in the active pin and channel map for the project. By default, the editor disconnects all instrument sessions when you close the project or delete, exclude, or deactivate the active pin and channel map in the project. This option is dimmed when the editor is not actively connected to any instrument sessions. If you created the sessions using the **Create New Sessions** option in the Connect to Instruments dialog box, using this menu item to disconnect the instruments closes those sessions, and the instruments are no longer in use. If you instead connected to existing open sessions an external application created, using this menu item to disconnect the instruments only disconnects the Digital Pattern Editor from those instruments. The external application that created the sessions keeps the connection to the instruments, and the instruments maintain their state as long as the external application maintains its connection to the instruments.
- **Synchronized** — Specifies one of the following synchronization options. When connected to external sessions, this menu displays the synchronization state set by the owner of the external sessions and is dimmed because this can only be changed by the owner of the external sessions. For more information about synchronizing instruments with the NI TestStand Semiconductor Module™, refer to ***NI-Digital Pattern Instrument Synchronization with TestStand Semiconductor Module™***. Visit ni.com/info and enter the Info Code `exac4m` to access the tutorial.
  - **No** — Specifies no instrument synchronization of the digital pattern instruments. If multiple instruments from separate single-instrument sessions are currently synchronized, selecting this option prompts you to recreate the

sessions to disable the synchronization. You cannot disable the synchronization without recreating the instrument session.

> **Note** This option cannot disable the synchronization of instruments within the same group. To disable the synchronization that occurs when you assign digital pattern instruments to the same group, assign a unique **Group** name to each digital pattern instrument in the Pin Map.

- **Yes** — Specifies to synchronize all digital pattern instruments in single-instrument sessions to a single synchronization group per chassis. This is the default value. Select this menu item in the following situations:
  - You want to burst patterns in sync, starting at the same time, across all digital pattern instruments that the active pin and channel map uses
  - A site is shared across multiple instruments

> **Note** When the test system includes a PXIe-6674T, the menu item is **Yes, using 6674T**. When you select **Yes** to synchronize all digital pattern instruments and a PXIe-6674T in the test system uses a supported version of the NI-Sync driver, the editor opens a session to the PXIe-6674T and enables failure and match propagation with the following considerations:
>
> - If no digital pattern instruments in the pin map are within a group — The editor synchronizes failure and match propagation across all digital pattern instruments in the same chassis.
> - If some digital pattern instruments are within a group and others are not — The editor enables failure and match propagation according to whether the first instrument listed in the pin map is in a group. For example, if the first instrument listed in the pin map is assigned to a group, the editor applies failure and match propagation only to instruments in that specific group. If the first instrument is not in a group, the editor enables failure and match propagation on all synchronized single-instrument sessions in the chassis. If a single chassis includes multiple groups of digital pattern instruments, the editor only enables failure and match propagation for the first group listed in the pin map.

> **Note** When using a multi-chassis system with PXIe-6674T, NI recommends having only one PXIe-6674T per chassis. Failure and match propagation will only be synchronized for digital pattern instruments in the same chassis.

> **Note** When the editor is connected to existing open instrument sessions, you cannot select this menu item, and the **Synchronized** menu item displays **N/A**. If the external application that created the instrument sessions configured the sessions for synchronization, the instruments maintain synchronization while in use by the Digital Pattern Editor, but the editor cannot alter the synchronization settings for those sessions.

  - **Yes, with Timing Absolute Delay enabled** — Specifies whether NI-TClk performs fine granularity adjustment of the intermodule skew in the digital pattern instrument timing reference. When Timing Absolute Delay is not enabled, the digital pattern instrument limits the timing adjustment for intermodule skew to a granularity of one master clock period. When Timing Absolute Delay is enabled, the digital pattern instrument adjusts the full extent of the intermodule skew measured by NI-TClk for better timing alignment among synchronized instruments.

> **Note** Do not enable Timing Absolute Delay with digital pattern instruments in a Semiconductor Test System (STS). Timing absolute delay conflicts with the adjustment performed during STS timing calibration.

  - **N/A** — Specifies that instrument synchronization does not apply because the editor is connected to only one digital pattern instrument session.
  - **Unknown** — Specifies that the instrument synchronization configuration is unknown because the DPE is connected to externally owned sessions. The owners of the external sessions determine the synchronization status of the sessions.
- **Enabled Sites** — Launches the Sites dialog box, which you use to enable and disable sites defined in the active pin and channel map. You can disable sites for debugging purposes, such as ignoring sites that return results that are not relevant

to the issue you are debugging or disabling a site when a DUT fails a test.

Use the **Pattern Burst** column of the Sites dialog box to select sites for bursting patterns, including for Shmoo plots and the digital scope. When you remove the checkmark from the **Pattern Burst** checkbox of a site in the Sites dialog box, rather than disabling it, you can still view the site in the System View and modify pins in the site in the Pin View or System View pane, but bursting the pattern has no effect on the site.

Use the **Site Enabled** column of the Sites dialog box to enable or disable all operations on a site. When you disable a site by removing the checkmark from the **Site Enabled** checkbox of a site, the **Pattern Burst** column for that site automatically dims. The digital pattern instrument interacts only with enabled sites. The enabled or disabled setting persists until you close the instrument sessions you created.

The Sites dialog box displays the current status of all sites. When you connect to existing open instrument sessions, the sites maintain the status the existing sessions specify. All sites for which all instruments are available are enabled by default when you use the **Create New Sessions** option in the Connect to Instruments dialog box to create a connection to instrument sessions for all available digital pattern instruments, NI-DCPower instruments, and relay driver modules defined in the active pin and channel map file for the project.

- **History RAM and Signal Settings** — Displays the Instrument Settings pane, which you use to configure options for logging History RAM results and for routing events to PXI chassis backplane trigger lines.
- **Disconnect All Digital Pins** — Specifies to change the state of all Digital pins to Disconnect from another supported state.
- **Reset** — Specifies to close and recreate instrument sessions for all available digital pattern instruments and NI-DCPower instruments defined in the active pin and channel map for the project and resets all the settings for the instruments back to default values. This menu item is enabled only when you select the **Create New Sessions** option in the Connect to Instruments dialog box. You cannot reset instrument sessions an external application owns.
- **Pin View** — Displays the Pin View pane, which you can use to interactively view and modify the current state of the pin driver, active load, comparators, PPMU, and NI-DCPower instrument settings for a single pin at a time in the active instrument

session.

> **Note** You must install the NI-DCPower driver to access NI-DCPower-related functionality in the Digital Pattern Editor.

- **System View** — Use this view to see or edit all the settings and measurements for all the pins and relays in the active pin and channel map connected to a digital pattern instrument, NI-DCPower instrument, or relay driver module and to see a list of patterns currently loaded on the digital instruments. You can also use this view to edit pattern sequencer flags and registers and interactively debug and make changes to the state of your digital pattern instruments, NI-DCPower instruments, or relays.
- **Relay View** — Launches the System View and scrolls and expands the Relay section as necessary to show the relays. Refer to the ***Related reference*** section at the end of this topic for a link to more information about System View.

> **Note** You must install the NI-DCPower driver to access NI-DCPower-related functionality in the Digital Pattern Editor.

> **Note** You must install the DAQmx driver to access relay driver functionality in the Digital Pattern Editor.

Related reference:

- Connect to Instruments Dialog Box
- Instrument Settings Pane
- Pin View Pane
- System View

Related information:

- System View

Connect to Instruments Dialog Box

When the Digital Pattern Editor cannot find open instrument sessions, it launches the

Connect to Instruments dialog box, which includes the following connection options:

- **Create New Sessions** — Specifies to create new instrument sessions for all available digital pattern instruments, NI-DCPower instruments, and relay driver modules defined in the active pin and channel map for the project. Select this option when you want to use the instruments only with the digital pattern editor and do not need to share them with a separate test program. All sites for which all instruments are available are enabled by default when you use this option. If the DPE is connected to any simulated instruments, a hot pink banner will appear in the menu bar.
- **Demo Mode** — Specifies to run the editor in demo mode. The DPE simulates session connections in software to digital pattern instruments, NI-DCPower instruments, and relay driver modules in the active pin and channel map. Select this option only for demonstration purposes because generated results do not reflect a realistic execution of patterns. The DPE will indicate that it is in demo mode by displaying a banner in the menu bar.
- **Continue Disconnected** — Specifies to continue in disconnected mode. Select this option when you want to share instruments with a separate test program or when you only need to view and edit the digital project files in the Digital Pattern Editor. In disconnected mode, the editor tries in the background to connect to existing instrument sessions that match the active pin and channel map, and, if successful, automatically switches to connected mode. If the pin and channel map defines multiple sites across multiple instruments, the editor connects when all the instrument sessions for at least one site exist. To begin sharing instruments, use a separate test program with the same active pin and channel map in the digital project to create instrument sessions. The editor automatically connects to the instrument sessions when the sessions exist. Before you actively burst patterns or change the configuration of instruments from the editor, pause the test program, such as with a breakpoint, so that the test program is not actively trying to use the instruments at the same time as the editor.

Pin and Channel Map Editor

Use the pin and channel map editor to view, create, modify, and save pin and channel map files, which define the relationships among pins, sites, channels, and digital pattern instruments in the test system. If you change the pin and channel map file, you must manually update any other references to pins in levels, time sets, and patterns.

Select **File** » **New** or click the **Add** button + on the Project Explorer window toolbar to create a new pin and channel map file. Double-click a `.pinmap` file in the Project Explorer window or right-click the file and select **Open** from the context menu to launch the pin and channel map editor.

Outside of the pin and channel map editor, you can use the System View or the tooltip for the Pin selector in the Pin View pane to display the instrument and channel that correspond to each pin.

Note By default, the Digital Pattern Editor automatically connects and resets instrument sessions when you open a project with an active pin and channel map or when you edit or add an active pin and channel map to a project. The editor disconnects all instrument sessions when you close the project or delete, exclude, or deactivate the active pin and channel map in the project.

When you close the pin and channel map editor with the **Cancel** button and the pin map file has been edited since you opened it in the editor, a dialog box prompts you to discard changes or return to the editor. When you close the pin and channel map editor with the **OK** button, the file updates on disk with the changes you made in the editor, and the instrument sessions close and re-open as needed to reflect the changes to the pin and channel map file.

Errors and Warnings

The pin and channel map editor uses a red error icon in the Errors and Warnings window to indicate that the file does not conform to the pin and channel map file format, or schema. Click the error icon or double-click the error message to highlight the error on the XML tab. The Errors and Warnings window displays a warning in orange text when the file conforms to the pin and channel map schema but will generate an error at run time.

Note The Pin Map Editor might report errors or warnings if the pin map file was created in a newer version of NI TestStand Semiconductor Module™ or the Digital Pattern Editor and contains elements or attributes that the current version of the editor does not know about. However, in most cases the editor will still be able to use the pin map as the active pin map. All other functionality in the editor that depends on the pin map will still work

> because the editor ignores the elements and attributes it does not know about. NI recommends you use the Pin Map Editor of the newer version of NI TestStand Semiconductor Module™ or the Digital Pattern Editor to edit pin maps that contain such elements or attributes.

Configuring the Pin and Channel Map

The pin and channel map editor includes the following options and tabs:

- **Pin Map File** — Displays the pin and channel map file path.
- **Undo** — Removes the last edit made.
- **Redo** — Reinstates the last edit removed.
- **Pin Map tab** — Displays an editable, hierarchical view of the instruments, pins, pin groups, relays, relay groups, relay configurations, sites, and connections in the pin map file. For each item, you can edit the attributes of the item, insert new items, or insert comments in the file. Refer to the ***Pin Map Tab*** for more information.
- **XML tab** — Displays the pin and channel map file in a text format that you can edit.
- **Errors and Warnings window** — Displays issues to resolve. Click the **Goto Error in XML** error icon to highlight the error on the XML tab.

Related reference:

- Pin View Pane
- System View

Pin Map Tab

The Pin Map tab contains the following sections:

- **Instruments** — Use this section to specify the instruments required to execute the test program.
- **Pins** — Use this section to specify the DUT pins and other pin types connected to the tester.
- **Relays** — Use this section to specify the relays on the tester that the test program associates with the pin map file references.
- **Pin Groups** — Use this section to specify a grouping of pins that you can reference with a single name.

- **Relay Groups** — Use this section to specify a grouping of relays that you can reference with a single name.
- **Relay Configurations** — Use this section to specify a set of relays and their positions.
- **Sites** — Use this section to specify the sites on the tester.
- **Connections** — Use this section to specify mappings from pins to instrument channels for every site and for system resources.

Related tasks:

- Using a Register Map

Instruments

Use the **Instruments** section on the Pin Map Tab to specify the type of instruments required to execute the test program and the name and attributes of each instrument. You can right-click any item in the section you want to edit and use the context menu to complete common tasks.

Choose one of the following options to add an instrument to the pin map file:

- Click **<Add Instruments Here>** to display the Instruments pane, and click the button of the instrument type you want to add.
- Right-click **<Add Instruments Here>** and select the instrument type you want to add from the context menu.

The Pin Map Editor automatically adds the instrument to the **Instruments** section. Select an instrument in the **Instruments** section to display the Instruments pane, where you can edit the attributes of the instrument.

You can also cut, copy, and paste instruments, or add comments in the Instruments pane. Use the **Comment** button to specify a comment for the selected instrument. Comments display beneath the instrument they modify.

Note Consider using the following instrument naming convention for semiconductor test programs:
`InstrumentType_ModelNumber_PXIChassisLocation_SlotLocation`,

for example, `HSD_657x_C2_S03`, where `InstrumentType` is an ASCII description of the instrument, `ModelNumber` is the model number as defined at ni.com/, `PXIChassisLocation` uses a single digit to identify the PXI chassis (`Cx`), and `SlotLocation` uses double digits to identify the slot location (`Sxx`).

The Digital Pattern Editor supports the following instrument types and instrument attributes:

- **Digital Pattern** — Defines an NI-Digital Pattern instrument.
  - **Name** — Name of the instrument, as defined in Measurement & Automation Explorer (MAX).
  - **Number of Channels** — Number of channels available on the instrument.
  - **Group** — Name of the group that contains the instrument. By default, the Pin Map Editor sets this attribute to `Digital` when you add NI-Digital Pattern instruments to the pin map file. When you use the same group name for all NI-Digital Pattern instruments, the Digital Pattern Editor combines all digital pattern instruments from the same chassis into a single session. To create multiple NI-Digital Pattern sessions, use a unique group name for each set of digital pattern instruments for which you want to create a session.

    Note You only can group digital pattern instruments of the same model. For example, you can group two PXIe-6570s but not a PXIe-6570 and PXIe-6571. The instruments must be in the same chassis.

- **DCPower** — Defines an NI-DCPower instrument.
  - **Name** — Name of the instrument, as defined in NI Measurement & Automation Explorer (MAX).
  - **Number of Channels** — Number of channels available on the instrument.
  - **Channel Group Name/Channel(s) table**— Lists the channel groups and the channels assigned to each group. By default, the Pin Map Editor creates one channel group containing all instrument channels. Use the plus (+) or minus (-) buttons to add or remove channel groups.

    Note A channel group is a selection of channels controlled by the

same instrument session. DCPower channel groups can contain channels from different physical DCPower instruments.

- **Channel Group Name**— Name of the channel group(s). The Channel Group Name is case sensitive and must not duplicate an instrument name or a group name on another instrument type.

  Note DCPower channels cannot be added to groups of other instruments.

- **Channel(s)**— Channel(s) assigned to a channel group. When you add a new channel group, the Pin Map Editor prompts you to add channels to the new group. Define the channels as a comma-separated list (e.g., 0,1,3,...,n), a continuous range (e.g., 0:3), or as a combination of the two (e.g., 0:1,3).

  Note All channels for all instruments must be assigned to a channel group and no channel can be assigned to more than one group.

- **Convert DCPower Instruments**— When you create a new DCPower instrument in the pin map file, Digital Pattern Editor 20.6 and later will create a single channel group containing all instrument channels. Digital Pattern Editor creates a single session for each channel group of DCPower instruments in the pin map file. Digital Pattern Editor 19.0 and earlier do not allow channel grouping . Pin maps created with Digital Pattern Editor 19.0 and earlier do not contain channel group information.

  Complete the following steps to convert all DCPower instruments in the pin map to use channel groups:

  1. Open the Pin Map Editor.
  2. Select the DCPower instrument in the Instruments section on the Pin Map tab.
  3. Click **Convert DCPower Instruments**.
  4. Click **Yes** to complete the conversion.

  Note

- To use NI-DCPower channel groups you must use Digital Pattern Editor 20.6 or later and NI-DCPower Driver 2020 or later. Both are fully backwards compatible, but methods used by Digital Pattern Editor and the driver must change when using channel groups.

Pins

Use the **Pins** section on the Pin Map Tab to specify the DUT pins and other pin types connected to the tester. You can right-click any item in the section you want to edit and use the context menu to complete common tasks.

Choose one of the following options to add a pin connection to the pin map file:

- Click **<Add Pins Here>** to display the Pins pane, and click the button of the pin type you want to add.
- Right-click **<Add Pins Here>** and select the pin type you want to add from the context menu.

The Pin Map Editor automatically adds the pin connection to the **Pins** section. Select a pin connection in the **Pins** section to display the Pins pane, where you can edit the pin's name.

Note Pin and pin group names are case sensitive, must begin with a letter or underscore (_), and are limited to `A-Z`, `a-z`, `0-9`, or _ characters.

You can also cut, copy, and paste pins, or add comments in the Pins pane. Use the **Comment** button to specify a comment for the selected pin. Comments display beneath the pin they modify.

DPE supports the following pin connection types:

- **DUT pin** — Specifies a DUT pin, which is a pin on a DUT or a resource on the tester or DIB that is associated with one or more sites.
  A DUT pin can be one of the following:
  - A specific pin on the DUT. This type of pin is connected to one instrument

channel per site.
    - A resource on the tester or DIB that has instrument connections and is associated with one or more sites. This resource can have one connection per site or can have one connection per group of sites.
- **System pin** — Specifies a system-wide pin, which is a resource on the tester or DIB that is connected to an instrument.

  A system pin is a resource on the tester or DIB that is connected to an instrument. A system pin has a single connection and is associated with all sites . You can also use DUT pins for shared resources by specifying the sites that share the resource in the connection. Use a DUT pin instead of a system pin if you need to burst patterns to the pin using the NI-Digital Pattern instrument driver.

Pin Groups

Use the **Pin Groups** section on the Pin Map Tab to specify a grouping of pins that you can reference with a single name.

Choose one of the following options to add a pin group to the pin map file:

- Click **<Add Pin Groups Here>** to display the Pin Groups pane, and click the **Pin Group** button.
- Right-click **<Add Pin Groups Here>** and select Pin Group from the context menu.

The Pin Map Editor automatically adds the pin group to the **Pin Groups** section. Select a pin group in the **Pin Groups** section to display the Pin Groups pane, where you can change the pin group name and use the individual checkboxes or the **Select All** checkbox to add or remove DUT pins, system pins, or pin groups from the selected pin group.

Note Pin and pin group names are case sensitive and must begin with a letter or underscore (_) and are limited to `A-Z`, `a-z`, `0-9`, or _ characters.

Once you create a pin group, you can also add pins or other pin groups to the pin group using one of the following options:

- Click **<Add Pin References Here>**, and click the **Pin Reference** button in the Pin

Groups pane to display a drop-down menu of available pins and pin groups.
- Right-click **<Add Pin References Here>**, and select Pin Reference from the context menu to display a drop-down menu of available pins and pin groups.
- Drag pins or pins groups from the **Pins** or **Pin Groups** section into a pin group.

You can also cut, copy, and paste pins, or add comments in the Pin Groups pane. Use the **Comment** button to specify a comment for the selected pin group. Comments display beneath the pin group they modify.

Relays

Use the **Relays** section on the Pin Map tab to specify the relays on the site and the relays on the tester that the test program associates with the pin map file references. You can right-click any item in the section you want to edit and use the context menu to complete common tasks.

Choose one of the following options to add a relay to the pin map file:

- Click **<Add Relays Here>** to display the Relays pane, and click the button of the relay type you want to add.
- Right-click **<Add Relays Here>** and select the relay type you want to add from the context menu.

The Pin Map Editor automatically adds the relay to the **Relays** section. Select a relay in the **Relays** section to display the Relays pane, where you can edit the relay's name, open state display label, and closed state display label.

Note Relay group names are case sensitive, must begin with a letter or underscore (_), and are limited to `A-Z`, `a-z`, `0-9`, or _ characters.

You can also cut, copy, and paste relays, or add comments in the Relays pane. Use the **Comment** button to specify a comment for the selected relay. Comments display beneath the relay they modify.

DPE supports the following relay types:

- **Site relay** — Specifies a site relay, which is a relay on the tester or DIB that is

connected to a relay driver module and that is associated with one or more sites. A site relay can have one connection per site or can have one connection per group of sites.
- **System relay** — Specifies a system relay, which is a relay on the tester or DIB that is connected to a relay driver module and that is associated with all sites.

Relay Groups

Use the **Relay Groups** section on the Pin Map tab to specify a grouping of relays that you can reference with a single name. You can right-click any item in the section you want to edit and use the context menu to complete common tasks.

Choose one of the following options to add a relay to the pin map file:

- Click **<Add Relays Here>** to display the Relay Groups pane, and click the **Relay Group** button.
- Right-click **<Add Relays Here>** and select Relay Group from the context menu.

The Pin Map Editor automatically adds the relay to the **Relays** section. Select a relay in the **Relays** section to display the Relays pane, where you can edit the relay's name, open state display label, and closed state display label.

The Pin Map Editor automatically adds the relay group to the **Relay Groups** section. Select a relay group in the **Relay Groups** section to display the Relay Groups pane, where you can change the relay group name and use the individual checkboxes or the **Select All** checkbox to add or remove site relays, system relays, or relay groups from the selected relay group.

Note Relay group names are case sensitive and must begin with a letter or underscore (_) and are limited to `A-Z`, `a-z`, `0-9`, or `_` characters.

Once you create a relay group, you can also add relays or other relay groups to the relay group using one of the following options:

- Click **<Add Relay References Here>**, and click the **Relay Reference** button in the Relay Groups pane to display a drop-down menu of available relays and relay groups.

- Right-click **<Add Relay References Here>**, and select Relay Reference from the context menu to display a drop-down menu of available relays and relay groups.
- Drag pins or pins groups from the **Relays** or **Relay Groups** section into a relay group.

You can also cut, copy, and paste relay groups, or add comments in the Relay Groups pane. Use the **Comment** button to specify a comment for the selected relay group. Comments display beneath the relay group they modify.

Relay Configurations

Use the **Relay Configurations** section on the Pin Map tab to specify a set of relays and their relay position states. You can right-click any item in the section you want to edit and use the context menu to complete common tasks.

Choose one of the following options to add a relay configuration to the pin map file:

- Click **<Add Relay Configurations Here>** to display the Relay Configurations pane, and click the Relay Configuration button.
- Right-click **<Add Relay Configurations Here>** and select Relay Configuration from the context menu.

The Pin Map Editor automatically adds the relay configuration to the **Relay Configurations** section. Select a relay configuration in the **Relay Configurations** section to display the Relay Configurations pane, where you can change the relay configuration name and select the relay position states.

Alternatively, when you create a new relay configuration, you can click **<Add Relay Configurations Here>**, then click the **Relay Position** button. Select the relay or relay group you want to add from the top drop-down menu, then select the relay position state from the bottom drop-down menu. To edit relay position states, use the Relay Configurations pane or click the **Relay Configurations** section to display the Relay Configurations table. The Relay Configurations table includes a column for each relay or relay group and an editable column for each relay configuration.

You can also cut, copy, and paste relay configurations, or add comments in the Relay Configurations pane. Use the **Comment** button to specify a comment for the selected relay configuration. Comments display beneath the relay configuration they modify.

Sites

Use the **Sites** section on the Pin Map tab to specify the sites on the tester. Site numbers start at 0 and must be consecutive without gaps. You can right-click any item in the section you want to edit and use the context menu to complete common tasks.

Choose one of the following options to add a site to the pin map file:

- Click **<Add Sites Here>** to display the Sites pane, and click the Site button.
- Right-click **<Add Sites Here>** and select Site from the context menu.

The Pin Map Editor automatically adds the site to the **Sites** section. Select a site in the **Sites** section to display the Sites pane.

You can also cut, copy, and paste sites, or add comments in the Sites pane. Use the **Comment** button to specify a comment for the selected site. Comments display beneath the site they modify.

Connections

Use the **Connections** section on the Pin Map Tab to specify mappings among pins, sites, instruments, and instrument channels. You can right-click any item in the section you want to edit and use the context menu to complete common tasks.

Choose one of the following options to add a connection to the pin map file:

- Click **<Add Connections Here>** to display the Connections pane, and click the button of the connection type you want to add.
- Right-click **<Add Connections Here>** and select the connection type you want to add from the context menu.

The Pin Map Editor automatically adds the connection to the **Connections** section. Select a connection in the **Connections** section to display the Connections pane, where you can edit the attributes of the connection.

You can also cut, copy, and paste instruments, or add comments in the Connections pane. Use the **Comment** button to specify a comment for the selected connection. Comments display beneath the connection they modify.

The Digital Pattern Editor supports the following connection types:

- **Connection** — Specifies a direct connection between a DUT pin in a specific site and an instrument channel for one or more sites.
- **System Connection** — Specifies a direct connection between a system pin and an instrument channel.
- **Multiplexed Connection** — Specifies a multiplexed connection between the same DUT pin on multiple sites and a single instrument channel. Multiplexed connections are not natively supported in the Digital Pattern Editor.
- **Relay Connection** — Specifies a connection between a site relay and a relay driver module control line for one or more sites.
- **System Relay Connection** — Specifies a direct connection between a system relay and a control line of a relay drive module.

Connections Table

Select the top-level **Connections** section to display the Connections table, which includes a row for every possible connection that can be made with the available pins, relays, and sites. Use the **View Connections for** control to filter the view by pin categories, such as by all pins and relays, DUT pins and site relays by site, and system pins and relays.

The Connections table includes the following sortable columns:

- **Pin** — Specifies the name of a device pin to connect to an instrument channel or the name of a relay to connect to a control line on a relay driver module.

  Refer to the specifications document for your digital pattern instrument for more information about the number of channels you can connect for each digital pattern instrument. You do not have to connect every pin in the pin map, but you might receive a warning for unconnected pins. Pin names take precedence over channel names if names are duplicated.

  > **Note** Currently, the Digital Pattern Editor does not support using system pins in patterns.

- **Site** — Indicates the test site to which the connection for a particular row applies.

After selecting an instrument or relay driver module for the row, you can enter a comma-separated list of site numbers to allow multiple sites to share the connection.

- **Instrument** — Specifies the name of the instrument or relay driver module to connect. Select **<Disconnect>** to remove the association between the pin and the instrument or the relay and the relay driver module.
- **Channel** — Specifies an instrument channel number to assign to the pin or a control line on a relay driver module to assign to the relay.
- **Multiplexer** — If applicable, specifies the multiplexer required to create the route. Multiplexers are not natively supported in the Digital Pattern Editor.
- **Route** — If applicable, specifies the multiplexer route required to connect the pin and site to the instrument and channel. Multiplexer routes are not natively supported in the Digital Pattern Editor.

Specifications

Use the specifications document to view, create, modify, and save specifications files (`.specs`). An asterisk (`*`) next to a filename in the Project Explorer window and in the document tab indicates that the file has been modified but not saved.

Select **File** » **New** or click the **Add Item** button + on the Project Explorer window toolbar to create a new document. Double-click a `.specs` file in the Project Explorer window or right-click the file and select **Open** from the context menu to launch the specifications document.

Use a specifications file to define a set of variables and associated numeric values that you can reference in levels, time sets, other specifications files, Shmoo operations, and test program code modules. You can specify the values directly or calculate the values from other variables using formulas you write with simple arithmetic expressions. Levels and timing files automatically update with value changes you make in specifications files. If you change the variable name, you must manually update any other references to the variable name.

Configuring Specifications

Each variable must include a section name you designate by the text you type before the period (`.`) in the Section.Variable column. For example, the section name for the

variable `DC.vcc` is `DC`. Variables must be unique within each section of a specifications file and among all specifications files associated with the test program.

Double-click or start entering data in the empty row at the bottom of the specifications table to create a new specification and specify the appropriate values. Use the reorder arrows to drag and drop specification rows to change the order of the specification definitions.

You can delete rows by selecting cells in the rows and selecting **Delete Selected Specification(s)** from the context menu or by selecting an entire row by clicking on the row header or selecting every cell in the row and using the <Delete> key.

You can specify the following components of each specification:

- **Section.Variable** — Name of the variable to use when querying the value. Variable names are case sensitive and must begin with a letter or underscore (_), are limited to `A-Z`, `a-z`, `0-9`, or `_` characters, and must include a period (`.`) to separate the section name from the variable name. Do not use math function names as variable names.
- **Formula** — Numeric value or formula definition for the variable. When you reference other variables to calculate a value, ensure that you use consistent units. By default, the specifications document displays the formula definition. Click the **Show Formulas/Show Computed Values** button to toggle the formula or numeric value view. Hover over the cell to display a tooltip with the formula and resolved value. The document highlights unresolved values and related dependencies and displays a tooltip with error information. Use the following options to specify a formula definition:

Option	Valid Values	Example(s)

Numeric values	—	`5.0`
		`-115.003`
		`3.4E-15`
		`150 ns`
		`3.4 mV`

Simple math operators	+, -, /, *, ( )	`5.0 + (3.14 * 2)`
Math functions	`abs`, `ceil`, `floor`, `fmod`,	`abs(-15) * 2`

Option	Valid Values	Example(s)
	`sin`, `cos`, `tan`, `asin`, `acos`,`atan`, `atan2`, `sinh`, `cosh`, `tanh`, `deg`, `rad`, `pi`, `exp`,`pow`, `log`, `log10`, `sqrt`, `min`, `max`	
Variable references	—	`AC.Variable1 * 15` where `Variable1` is another specification within the `AC` section
Units	Units: (take precedence over SI prefixes) ◦ V, volts, Volts, volt, Volt ◦ A, amps, Amps, amp, Amp, amperes, Amperes, ampere, Ampere ◦ s, sec, Sec, secs, Secs, second, Second, seconds, Seconds ◦ dB, db, decibel, Decibel, decibels, Decibels ◦ dBm, dbm ◦ F, farad, Farad, farads, Farads ◦ H, henries, Henries, henry, Henry ◦ Hz, hertz, Hertz ◦ Ω, ohms, Ohms, ohm, Ohm ◦ W, watts, Watts, watt, Watt SI Prefixes: ◦ d, deci (scaling factor of 1E-1) ◦ c, centi (scaling factor of 1E-2) ◦ m, milli (scaling factor of 1E-3) ◦ µ, u, micro (scaling factor of 1E-6) ◦ n, nano (scaling factor of	`10 ns + 15 ns`

Option	Valid Values	Example(s)
	1E-9) ◦ p, pico (scaling factor of 1E-12) ◦ f, femto (scaling factor of 1E-15) ◦ da, deca (scaling factor of 1E+1) ◦ h, hecto (scaling factor of 1E+2) ◦ k, kilo (scaling factor of 1E+3) ◦ M, mega (scaling factor of 1E+6) ◦ G, giga (scaling factor of 1E+9) ◦ T, tera (scaling factor of 1E+12) ◦ P, peta (scaling factor of 1E+15)	

- **Comment** — Describes what the formula represents. This field does not accept the <Enter> or <Tab> keys.

Related reference:

- Specifications File XML Structure
- Editing Documents
- Getting Started with Digital Pattern Instruments
- NI-Digital Pattern Driver Examples

Specifications File XML Structure

The specifications XML schema, located at `<Program Files>\National Instruments\Digital Pattern Editor\Specifications.xsd`, defines the structure for a specifications XML file.

Schema Version Policy

The schema version uses a major.minor notation. The version of the schema reflects changes to the schema, not changes to the Digital Pattern Editor.

Changes to the schema version indicate breaking changes to the schema. The editor does not load files that use a later schema version than the version specified for that version of the editor.

Use the following table to map schema versions and editor versions.

Note Digital Pattern Editor 18.0 and later can read all pin map file versions regardless of the schema version. These editor versions will ignore any parts of the pin map file that the editor does not use.

Schema Version	Digital Pattern Editor Version
PinMap.xsd version 1.2 Specifications.xsd version 1.0	Digital Pattern Editor 16.0
PinMap.xsd version 1.3	Digital Pattern Editor 17.0
PinMap.xsd	Digital Pattern Editor 18.0 and later

Levels

Use the pin levels sheet (`.digilevels`) to view, create, modify, save, and apply levels to your project. The sheet displays voltage and current levels for pins and pin groups connected to digital pattern instruments and NI-DCPower instruments. Pins and pin groups must be defined in the pin and channel map file.

Before you can access NI-DCPower functionality in Digital Pattern Editor, you must

install the NI-DCPower driver.

1. Select **File** » **New** » **Pin Levels Sheet** to create a new levels sheet.
   Each row in the sheet represents a unique set of level parameters for a pin or pin group.

   > **Note** Level parameters defined in pin or pin groups farther down the list override parameters defined earlier in the list.

2. Drag and drop the pin or pin group rows with the **reorder arrows** button to change the order of the level parameters.

   > **Note** Multiple pin groups can contain the same pin, enabling multiple unique levels for the same pin. Digital Pattern Editor highlights duplicate pins and pin groups in yellow as a warning for you to review and confirm the validity of the duplications.

3. To configure levels, enter and edit numeric values or formulas for each digital pin, power pin, or pin group. Formulas can include references to variables defined in specifications files included in the project. By default, the levels sheet displays the formula definition.
   Click the **Show Formulas/Show Computed Values** button to toggle the formula or numeric value view. Hover over the cell to display the formula and resolved value. The sheet highlights unresolved values and related dependencies. Hovering over these displays error information. Levels files automatically update value changes made in specifications files.

   > **Note** If you do not specify a new value for a duplicate pin or pin row, Digital Pattern Editor uses values from earlier rows in the levels sheet, API calls, or previously loaded levels.

4. Double-click or start entering data in the empty row at the bottom of the levels table to create a new levels row and specify values.

   > **Note** Delete rows by clicking on the row header and pressing the <Delete> key.

5. Set pin values.

6. Click the **Apply Levels** button to load level values to instruments after you make changes in the levels sheet. Applying levels values affects all enabled sites for which pattern bursting is also enabled.

Related reference:

- Pin Values
- Editing Documents
- Getting Started with Digital Pattern Instruments
- NI-Digital Pattern Driver Examples
- Levels File XML Structure

Pin Values

Use the following guidelines when setting pin values within a pin levels sheet in your project.

Note When specifying levels, specify them from the perspective of the DUT, not the digital pattern instrument.

Digital Pins

- Pin Item—Specifies the name of a digital pin or pin group defined in the pin and channel map.

  Note Digital Pattern Editor does not support using system pins in patterns. To burst from a pin that spans all sites, configure a pin with a shared pin connection in the Pin Map Editor to share a channel across multiple sites and use that pin in your pattern instead.

- V_{IH}— Specifies the voltage that the instrument applies to the input of the DUT when the instrument drives a logic high (1).
- V_{IL}— Specifies the voltage that the instrument applies to the input of the DUT when the instrument drives a logic low (0).
- V_{OH}— Specifies the DUT output voltage above which the comparator on the instrument pin interprets a logic high (H).
- V_{OL}— Specifies the DUT output voltage below which the comparator on the

instrument pin interprets a logic low (L).

- Termination Mode— Specifies the behavior of the pin when the pin function is set to digital and the pin state for the current cycle is not a drive state. Use the drop-down menu to select one of the following modes:
  - Blank/Empty Space—Specifies to leave the termination mode unchanged.
  - High Z— Specifies that, for non-drive pin states (L, H, X, V, M, E), the pin driver is put in a high-impedance state and the active load is disabled. This is the default setting.
  - V_{TERM}—Specifies that, for non-drive pin states (L, H, X, V, M, E), the pin driver terminates the pin to the configured V_{TERM} voltage through a 50 Ω impedance. V_{TERM} is adjustable to allow for the pin to terminate at a set level.

    > **Note** This is useful for devices that operate incorrectly when an instrument pin is unterminated and floats to any voltage level within the instrument voltage range. Setting V_{TERM} to 0 V and selecting the V_{TERM} termination mode has the effect of connecting a the termination to ground, which provides an effective 50 Ω impedance for the pin. This improves signal integrity by reducing reflections while the DUT drives the pin.

  - Active Load— Specifies that, for non-drive pin states (L, H, X, V, M, E), the active load is connected and the instrument either sources or sinks a defined amount of current to load the DUT. The voltage at which the instrument changes between sourcing and sinking is specified by V_{COM}. The amount of current sourced by the instrument and sunk by the DUT is specified by I_{OL}. The amount of current sunk by the instrument and sourced by the DUT is specified by I_{OH}.
- V_{TERM} — Specifies the termination voltage the instrument applies during non-drive cycles when the termination mode is set to V_{TERM}. The instrument applies the termination voltage through a 50 Ω parallel termination resistance.
- V_{COM}— Specifies the commutating voltage level at which the active load circuit switches between sourcing current and sinking current.
- I_{OL}— Specifies the current that the DUT sinks from the active load while outputting a voltage below V_{COM}.
- I_{OH}— Specifies the current that the DUT sources to the active load while outputting a voltage above V_{COM}.
- Comment—Specifies a text description for the digital pin or pin group level set.

This field does not accept the <Enter> or <Tab> keys.

Power Pins

Note Loading level values on the instrument only changes the values of V_F and I_F. Use the Pin View pane or external code in a test program to specify whether the NI-DCPower instrument is the active source of current, voltage, or neither for the pin.

- Pin Item—Specifies the name of a the power pin or pin group defined in the pin and channel map.
- V_F—Specifies the voltage to force on the specified pin connected to an NI-DCPower instrument.
- I_F—Specifies the current to force on the specified pin connected to an NI-DCPower instrument.
- I_C — Specifies the maximum limit for the current at the pin when the NI-DCPower instrument forces voltage to the DUT. This value is enabled when the NI-DCPower instrument is in Force Voltage mode.
- V_C — Specifies the maximum limit for the voltage at the pin when the NI-DCPower instrument forces current to the DUT. Enabled only when the NI-DCPower instrument is in Force Current mode.
- I Range—Specifies the current range to use. Select a range from the pull-down menu.
- V Range—Specifies the voltage range to use. Select a range from the pull-down menu.
- Sense—Specifies the method used when collecting output and voltage measurements.
  - Local—Use a single set of connections for output and voltage measurement.
  - Remote—Use two sets of connections for output and voltage measurement.

    Note Remote sense measurements are appropriate for high-current applications. Remote sense enables more accurate voltage output and measurements when the output lead voltage drop is significant.

  - Comment—Specifies a text description for the power pin or pin group level set. This field does not accept the <Enter> or <Tab> keys.

PPMU Pins

- **Pin item**—Specifies the name of a digital pin or pin group defined in the pin and channel map.
- V_F—Specifies the voltage to force on the specified pin connected to an NI-Digital instrument.
- I_F—Specifies the current to force on the specified pin connected to an NI-Digital instrument.
- V_{CH}—Specifies the nominal voltage at the specified pin at which the high-side voltage clamp activates when the PPMU applies current to the DUT.
- V_{CL}—Specifies the nominal voltage at the specified pin at which the low-side voltage clamp activates when the PPMU applies current to the DUT.
- I Range—Specifies the current range to use. Use the drop-down menu to select a range.
- Comment—Specifies a text description for the power pin or pin group level set. This field does not accept the <Enter> or <Tab> keys.

Related tasks:

- Levels

Related reference:

- Editing Documents
- Getting Started with Digital Pattern Instruments
- NI-Digital Pattern Driver Examples
- Levels File XML Structure

Levels File XML Structure

The levels XML schema, located at `<Program Files>\National Instruments\Digital Pattern Editor\PinLevels.xsd`, defines the structure for a levels XML file.

Related reference:

- Editing Documents

- Getting Started with Digital Pattern Instruments
- NI-Digital Pattern Driver Examples
- Levels File XML Structure

Timing

Use the timing document to view, create, modify, save, and apply timing files (`.digitiming`). Timing files include period, edge multiplier (edge x), drive format, and edge information for pattern execution. Timing files currently can contain only one timing sheet, which is a collection of time sets. An asterisk (`*`) next to a filename in the Project Explorer window and in the document tab indicates that the file has been modified but not saved.

Select **File** » **New** or click the **Add Item** button on the Project Explorer window toolbar to create a new document. Double-click a `.digitiming` file in the Project Explorer window or right-click the file and select **Open** from the context menu to launch the timing document.

The timing document displays configuration components of the time sets, including the format and edge placement that shape the digital waveform on a per-pin basis. You can define multiple time sets with different edge multiplier, format, and edge placement components per pin to use on different vectors in the pattern. Each vector references only one time set. You can create as many time sets as you have unique combinations of pin behavior required in pattern burst up to the maximum number of uniquely named time sets the digital pattern instrument supports. Refer to the specifications document for the digital pattern instrument for more information about the number of supported time sets. You can create multiple timing files and apply those values to separate pattern bursts.

A time set can include timing definitions for multiple different pins or pin groups, each represented on a separate row in the timing document. You can define the same time set more than once for the same pin on different rows if needed. Time set components defined in pin or pin group rows farther down in the list override components defined in pins or pin groups earlier in the list. Use the reorder arrows to drag and drop time set rows to change the order of the time set definitions.

Applying Time Sets

Click the **Apply Time Sets** button or right-click the timing file (`.digitiming`) in the Project Explorer window and select **Apply Time Sets** from the context menu to delete any previously applied values and load all the time set values in the document on the instrument after you make changes. Applying time set values affects all sites, even disabled sites. The Digital Pattern Editor does not prompt you to reapply time set values when you enable a previously disabled site because the time set values have already been applied. The editor validates that all formulas resolve successfully and displays an error message and highlights the related cell for formulas that include errors. The editor prompts you to apply the values in the active timing file when no timing values have previously been applied to the instrument or when previously applied timing values change directly in the timing document or indirectly through changes in the specifications document.

Configuring Time Sets

You can enter and edit numeric values or formulas for the time set components. Formulas can include references to variables defined in specifications files included in the project. By default, the timing document displays the formula definition. Click the **Show Formulas/Show Computed Values** button to toggle the formula or numeric value view. Hover over the cell to display a tooltip with the formula and resolved value. The document highlights unresolved values and related dependencies and displays a tooltip with error information. Timing files automatically update with the value changes you make in specifications files.

Double-click or start entering data in the empty row at the bottom of the timing table to create a new time set and specify the appropriate values.

You can delete rows by selecting cells in the rows and selecting **Delete Selected Row(s)** from the context menu or by selecting an entire row by clicking on the row header or selecting every cell in the row and using the <Delete> key.

You can set values for the following components for time sets. You must specify a value for each component. The timing document highlights unspecified components in orange.

- **Name** — Specifies the name of the time set.

- **Period** — Specifies the period of vectors that use the time set. You can specify only one period per time set, meaning all pins in the time set use the same period. When you modify the period of a time set, all the rows for that time set automatically update to use the same period.

  Note You can specify edges that exceed the period of the time set up to the limits the digital pattern instrument supports, but all edges must still occur in order and must honor all minimum separation requirements. Do not schedule edges for future cycles before all edges of the current cycle complete. Validation for these situations does not occur.

- **Pin Item** — Specifies a pin or pin group defined in the pin and channel map.

  Note Currently, the Digital Pattern Editor does not support using system pins in patterns. To burst from a pin that spans all sites, configure a pin with a shared pin connection in the Pin Map Editor to share a channel across multiple sites and use that pin in your pattern instead.

- **Edge Multiplier** — Specifies the edge multiplier for a time set. You can choose 1x or 2x multipliers. If you choose 2x multiplier, you can specify values for the 2x edge components, including **Drive Data2**, **Drive Return2**, and **Compare Strobe2**. If the document does not contain any 2x time sets, the 2x edge columns are hidden. The document disables editing 2x edge cells for 1x time set rows.
- **Drive Format** — Specifies one of the following drive formats, as illustrated in the following figures.

Figure 1. Drive Formats, Including Surround by Complement

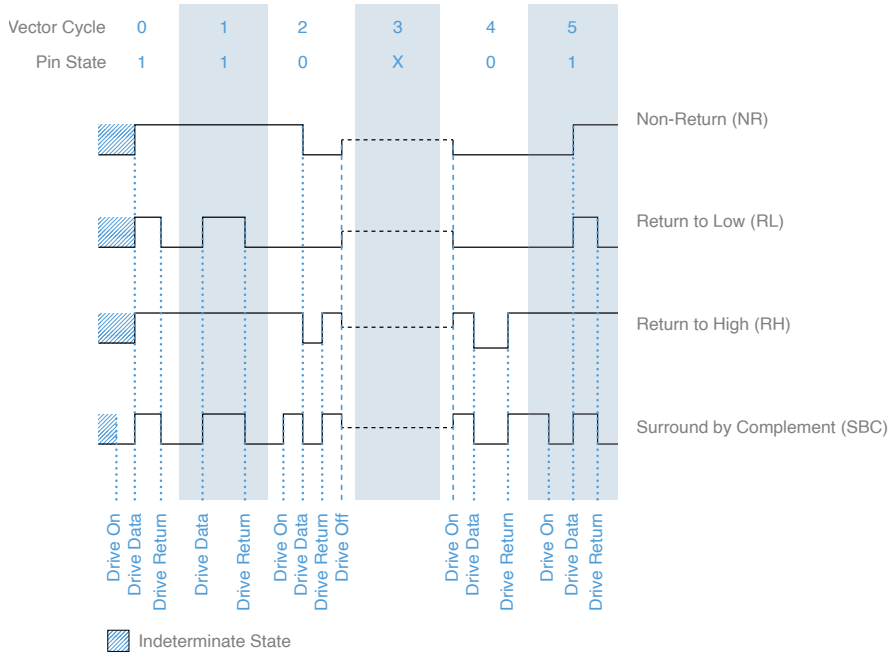
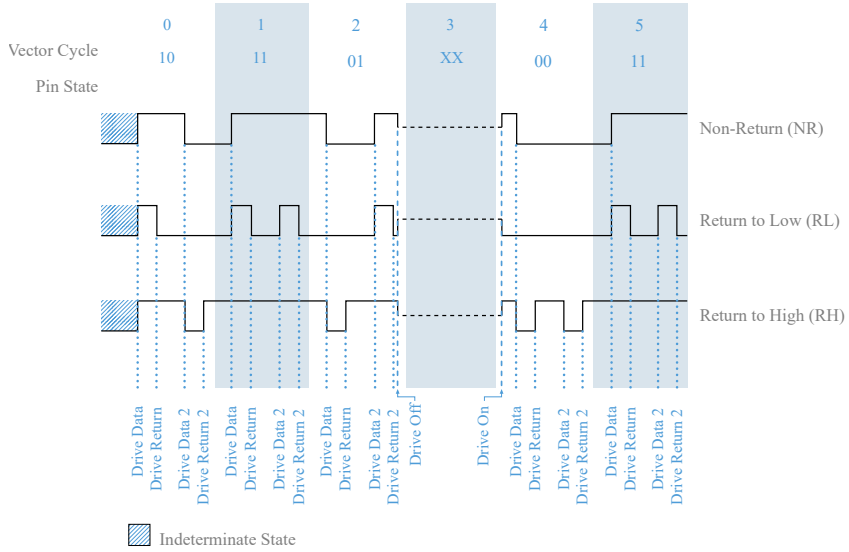


Figure 2. Drive Formats, 2x Multiplier



  - **NR** — Non-return. Setting **Drive Format** to NR disables the **Drive Return** cell in that row and clears that cell if a value was present.
  - **RL** — Return to low.
  - **RH** — Return to high.
  - **SBC** — Surround by complement.
- Use the **Drive On**, **Drive Data**, **Drive Return**, **Drive Data2**, **Drive Return2**, **Drive Off**, **Compare Strobe**, and **Compare Strobe2** columns to configure edge components for each pin or pin group per time set.

Note On power-up, pins are in a high impedance state. To avoid unexpected results for DUTs that are sensitive to the initial state of a pattern burst, programmatically specify the initial pin state by using the niDigital Write Static VI, the DigitalPinSet.WriteStatic .NET method, or the niDigital_WriteStatic C function. You can interactively set the initial state for a pin by selecting the corresponding driver symbol in the Drive section of the Pin View pane of the Digital Pattern Editor. After each pattern burst, the ending state of the last executed vector persists until the start of the next pattern burst or until you programmatically or interactively change the pin state or pin function.

- **Drive On** — Specifies the delay from the beginning of the vector period for turning on the pin driver. This option applies only when the previous vector left the pin in a non-drive pin state (L, H, X, V, M, E).
- **Drive Data** — Specifies the delay from the beginning of the vector period until the pattern data is driven to the pattern value.
- **Drive Return** — Specifies the delay from the beginning of the vector period until the pin changes from the pattern data to the return value, as specified by the format. Because not all formats use this column, the document disables editing this field when the drive format does not support it.
- **Drive Data2** — Specifies the delay from the beginning of the vector period until the 2x time set's second pattern data is driven to the pattern value. The document disables editing this column for 1x time sets. The document hides this column if there are no 2x time sets.
- **Drive Return2** — Specifies the delay from the beginning of the vector period until the pin changes from the 2x time set's second pattern data to the return value as specified by the format. Because not all formats use the **Drive Return2** column, the document disables editing this field when the drive format does not support it. The document also disables editing this column for 1x time sets. The document hides this column if there are no 2x time sets.
- **Drive Off** — Specifies the delay from the beginning of the vector period to turn off the pin driver when the next vector period uses a non-drive symbol (L, H, X, V, M, E).
- **Compare Strobe** — Specifies the time when the comparison happens within a vector period.
- **Compare Strobe2** — Specifies the time when a 2x time set's second comparison happens within a vector period. The document disables editing

this column for 1x time sets. The document hides this column if there are no 2x time sets.

- **Comment** — Specifies a text description for the time set row. This field does not accept the <Enter> or <Tab> keys.

Keyboard Shortcuts within the Timing Document

Action	Shortcut
Apply time sets	<Ctrl+L>

Related reference:

- Timing File XML Structure
- Editing Documents
- Getting Started with Digital Pattern Instruments
- NI-Digital Pattern Driver Examples

Timing File XML Structure

The timing XML schema, located at `<Program Files>\National Instruments\Digital Pattern Editor\Timing.xsd`, defines the structure for a timing XML file.

Patterns

Use the pattern document to view, create, modify, load, burst, and debug binary pattern files (`.digipat`). A digital pattern file can contain only one pattern.

Select **File** » **New** or click the **Add Item** button on the Project Explorer window toolbar to create a new document. Double-click a `.digipat` file in the Project Explorer window or right-click the file and select **Open** from the context menu to launch the pattern document.

Note If you are working with large pattern files, ensure that the system includes at least 8 GB of RAM. NI recommends 16 GB of RAM.

Text Pattern File Format

The Digital Pattern Editor requires a compiled, binary version of the pattern file to edit or burst. You can compile an ASCII text pattern file format (`.digipatsrc`) into a binary version. The ASCII format can be helpful in the following situations:

- Writing tools or scripts that generate pattern files.
- Writing tools or scripts that convert text pattern files from other testers to the NI text pattern file format.
- Storing revisions of the text pattern file in source code control systems.
- Using a third-party application to compare revisions of the pattern file.
- Manually editing pattern files in a standard text editor.

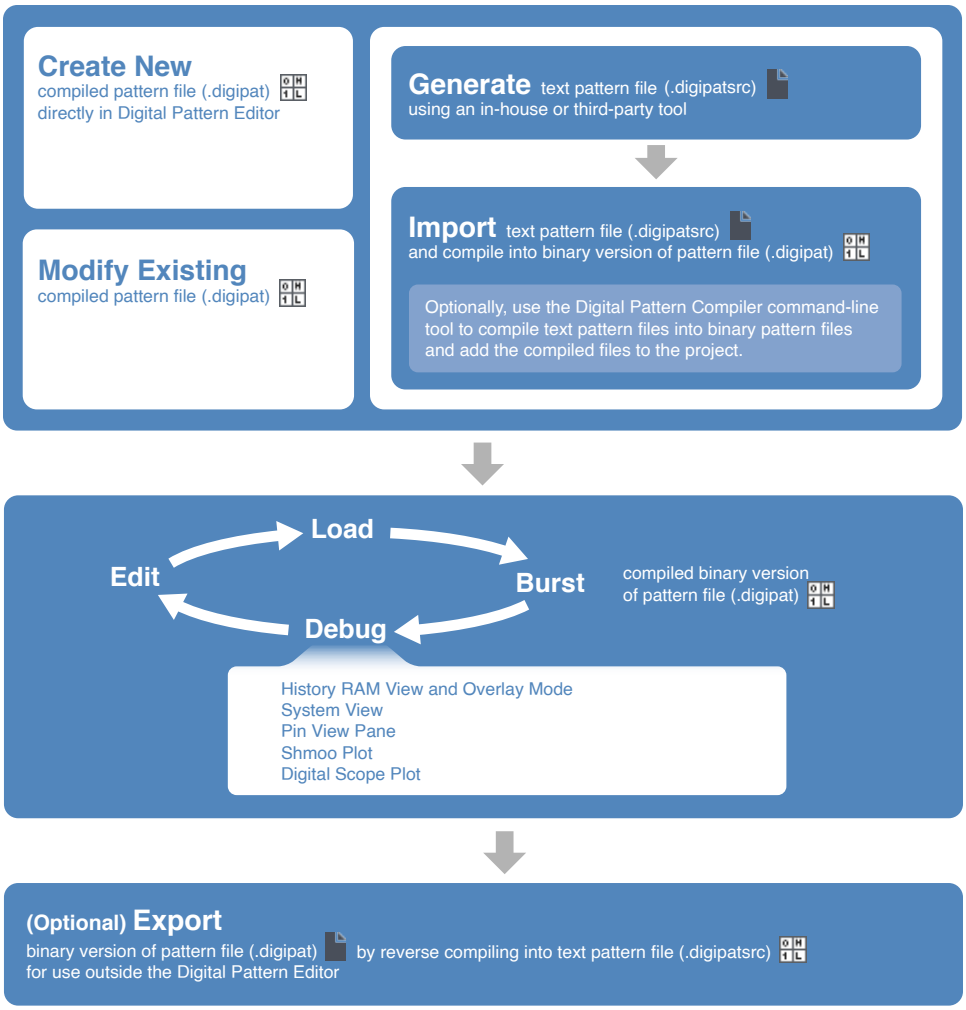
Related reference:

- Compiling Pattern Files
- Pattern Grid View
- Editing Documents
- Text Pattern File Syntax
- Opcodes
- Bursting Patterns and Viewing Results
- Debugging Techniques
- Keep Alive Patterns
- Scan Patterns
- Getting Started with Digital Pattern Instruments
- NI-Digital Pattern Driver Examples

Pattern File Workflow

You typically use the following workflow for pattern files.

Refer to the ***Related reference*** section at the end of this topic for a link to more information about each of these components.



Related reference:

- Generating Patterns
- Importing Patterns
- Bursting Patterns and Viewing Results
- Debugging Techniques
- Exporting Patterns

Generating Patterns

A digital pattern file can contain only one pattern. You typically use one of the following methods to generate pattern files:

- Create a simple pattern in the Digital Pattern Editor.
- Write custom scripts or use an in-house or third-party tool to convert existing pattern files or files that an automatic test pattern generation (ATPG) tool creates,

such as Waveform Generation Language (WGL) or Standard Test Interface Language (STIL) files, to the NI text pattern file format (`.digipatsrc`).
- Use custom tools to output `.digipatsrc` files, such as generating a pattern based on a list of register transactions.

Related reference:

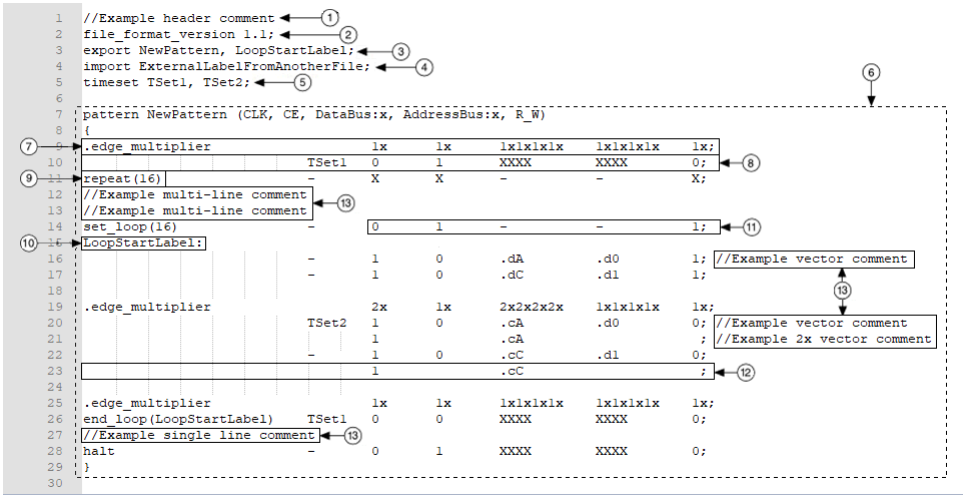
- Text Pattern File Syntax
- Importing Patterns
- Compiling Pattern Files

Related information:

- Third-Party Pattern Conversion Tools

Text Pattern File Syntax

The NI text pattern file (`.digipatsrc`) uses the following syntax. Refer to the ***Related reference*** section at the end of this topic for a link to more information about each of these components.



1. Header comment
2. File format version declaration (required)
3. Export declaration
4. Import declaration
5. Time set declaration

6. Pattern declaration (required)
7. Edge multiplier statement (required if using 2x vectors)
8. Vector statement
9. Opcode
10. Label
11. Pin state data
12. Additional pin state data
13. Vector comments

Related reference:

- Header Comments
- File Format Version Declaration
- Export Declaration
- Import Declaration
- Time Set Declaration
- Pattern Declaration
- Vector Statement
- Opcodes
- Identifiers and Reserved Keywords
- Pattern Grid View
- Scan Pattern File Syntax

Header Comments

The text version of the pattern file can include an arbitrary number of comments before the file format version declaration. Each line of a header comment uses double forward slashes (`'//'`) followed by Unicode characters.

By default, importing or compiling a text pattern file (`.digipatsrc`) into a binary version of the pattern file (`.digipat`) preserves comments. To remove comments when you create a compiled, binary version of the pattern file, use the `-no-comments` command-line option in the Digital Pattern Compiler.

Related reference:

- File Format Version Declaration
- Compiling Pattern Files

File Format Version Declaration

The required file format version declaration specifies the unique version of the NI text pattern file format to which the pattern complies using the following syntax:

```
file_format_version major.minor
```

where major and minor are decimal numbers.

Conditions

The following conditions apply to the file format version declaration:

- The file format version declaration must be the first declaration in the pattern file.
- The current version of the file format is `1.1`.
- You must add the pattern name to the export declaration to use the pattern name in the import declaration of another pattern.
- Vectors start loading in FVM and CVM only when you explicitly export the pattern name or when the pattern name is the target of a `jump`, `call`, or looping opcode. If you do not export the pattern name, vectors start loading in LVM.

> **Note** Pattern files with a file format version of `1.0` always export the pattern name, and vectors always start loading in FVM and CVM.

Export Declaration

The export declaration specifies the list of labels to export for other patterns to use as opcode targets using the following syntax:

```
export item1, item2, item3, …;
```

where item<n> is a valid label.

Conditions

The following conditions apply to the export declaration:

- The export declaration must be specified before the pattern declaration.
- You can include multiple export declarations on the same or separate lines.
- You cannot duplicate items in an export declaration.
  - Exported labels and pattern names must be unique across all loaded patterns.
  - Local labels that are not exported do not have to be unique across patterns because their scope is limited to a single pattern.
- You must export a pattern name to use it as a target of an opcode from another pattern.
- You must define the export as a pattern name or a label in the pattern file.
- You can use a pattern name as a **start label** parameter value for niDigital API calls without exporting the pattern name.

Note In pattern files with a file format version of `1.0`, you cannot use the pattern name as an exported label.

Related reference:

- Pattern Declaration
- Import Declaration

Import Declaration

The import declaration specifies the list of external labels the pattern vectors use as opcode targets using the following syntax:

```
import item1, item2, item3, …;
```

where item<n> is a valid label.

The Digital Pattern Editor assumes that any label you reference but do not specify in a pattern is an imported label. You can import only global labels, the pattern name, or an exported label from another pattern. To burst a pattern that imports a label, you must load the pattern with the exported label for the reference to be resolved.

Conditions

The following conditions apply to the import declaration:

- The import declaration must be specified before the pattern declaration.
- You can include multiple import declarations on the same or separate lines.
- You cannot duplicate items in an import declaration.
- You cannot specify an imported item in an export declaration because imports and exports are mutually exclusive.
- You cannot reuse an import identifier as an explicit vector label in the importing pattern.
- An import cannot be the same as the pattern name, which is the automatic global label for the first vector.

Related reference:

- Pattern Declaration
- Export Declaration
- Pattern Grid View

Time Set Declaration

The time set declaration specifies the list of time set references the pattern vectors use using the following syntax:

```
timeset item1, item2, item3, …;
```

where item<n> is a valid identifier.

Conditions

The following conditions apply to the time set declaration:

- The time set declaration must be specified before the pattern declaration.
- You can include multiple time set declarations on the same or separate lines.
- You cannot duplicate items in a time set declaration.

Related reference:

- Pattern Declaration
- Pattern Grid View
- Timing

Pattern Declaration

The required pattern declaration defines the sequence of pattern vectors and defines the pins and pin groups to which the pattern applies using the following syntax:

```
pattern pattern_name (pin_item1, pin_item2:b, …)

{

vector_statement1; [// comment]

vector_statement2; [// comment]

...

}
```

where pattern_name (required) and pin_item<n> are valid identifiers.

Conditions

The following conditions apply to the pattern declaration:

- The pattern declaration must be the last declaration in the pattern file.
- The pattern file can contain only one pattern declaration.
- The pattern_name must begin with a letter or underscore (_) and is limited to `A-Z`, `a-z`, `0-9`, or _ characters. The pattern name is independent from the filename of the pattern file.
- A pin_item is a reference to a pin or pin group defined in the pin and channel map file used to compile the pattern. The text pattern file does not reference the pin and channel map file directly. The pin_item identifier can include one of the following optional format specifiers to set the display format for pin groups:
  - `:b` — binary or symbolic (default)
  - `:u` — unsigned integer
  - `:x` — hexadecimal
- All pin_items must resolve to a mutually exclusive set of pins. Two or more pin_items must not resolve to the same pin or set of pins.
- You cannot duplicate a pin_item in a pattern declaration.

Keep Alive Pattern Declaration

Keep alive patterns use the following pattern declaration syntax:

```
keep_alive_pattern pattern_name (pin_item1, pin_item2:b, …)

{

vector_statement1; [// comment]

vector_statement2; [// comment]

...

}
```

where pattern_name (required) and pin_item<n> are valid identifiers.

Conditions

The conditions for standard pattern declarations apply to pattern declarations for keep alive patterns. The following additional conditions apply to keep alive patterns:

- A keep alive pattern can include up to only 16 vectors.
- You can use only dashes (–) for time sets. Keep alive patterns execute the same time set used on the vector on which you executed the `keep_alive` opcode to begin the keep alive pattern.
- You can use only the `keep_alive` and `repeat` opcodes in keep alive patterns.
- You cannot use labels in keep alive patterns.
- You can use only `0`, `1`, `X`, and – for the pin state data in a keep alive pattern.

Related reference:

- Pattern Grid View
- Vector Statement
- Flow Control Opcodes
- Keep Alive Patterns

Edge Multiplier Statement

The edge multiplier statement specifies the edge multiplier for subsequent vector statements.

The edge multiplier statement uses the following syntax:

```
.edge_multiplierpin1_edge_multiplierpin2_edge_multiplier
...;
```

where the edge multiplier keyword designates the start of an edge multiplier statement and is followed by an edge multiplier marker per pin item.

Edge Multiplier Statement Conditions

The following conditions apply to edge multiplier statements:

- The edge multiplier statement must come before the vector statement.
- The edge multiplier keyword (edge_multiplier) specifies the beginning of the edge multiplier statement.
- The number of edge multiplier markers that follow the edge multiplier keyword must equal the number of pin items specified in the pattern declaration.
- A terminating semicolon (;) specifies the end of an edge multiplier statement.

Edge Multiplier Marker Conditions

The following conditions apply to edge multiplier markers:

- An edge multiplier marker consists of multiple edge multiplier values concatenated together without whitespace.
- For a pin_item that is a single pin, the edge multiplier marker must contain only a single edge multiplier value.
- For a pin_item that is a pin group, the edge multiplier marker must contain an edge multiplier value for each pin in that pin group.

Vector Statement

The vector statement specifies the label for the vector, the opcodes to execute, the

data to drive to or compare from the DUT pins, and the time set that contains the timing information to apply to the data using the following syntax:

```
[label:] [opcode] timesetpin1_statepin2_state …; [// comment]
```

where the required timeset is a valid identifier the time set declaration defines or a single dash (–) to indicate to repeat the previous time set and pin<n>_state represents the pin states.

Vector Statement Conditions

The following conditions apply to vector statements:

- Every vector must include a time set reference, which can be a time set declared in the time set declaration or a single dash (–). A dash specifies to use the same time set as the previous cycle. Do not use a dash (–) for the time set on the first vector of a pattern file unless the file is used only as a target of a jump or call operation. At run time, the time set reference resolves to a time set. An error occurs when the time set declaration does not contain the time set you reference in the vector statement.
- A terminating semicolon (`;`) specifies the end of the vector data.
- The pattern sequencer does not automatically stop execution at the end of the pattern. You must specify the `halt` opcode on the last vector in the pattern to stop the execution of the pattern. Undefined behavior results in patterns that complete without a halt code.

Label Conditions

The following conditions apply to labels:

- The first vector in a pattern file is automatically assigned a label that matches the pattern name. You must export a pattern name to use it as a target of an opcode from another pattern. You can use a pattern name as a **start label** parameter value for niDigital API calls without exporting the pattern name. The label resolves to the address of the first vector in the pattern of the same name. If you rename or delete a label, you must manually update any other references to that label to avoid run-time errors.

Note For memory efficiency, use labels only on vectors that are targets for jumps, calls, or loops. Use comments for general documentation about the behavior of the pattern.

- Names must begin with an ASCII letter or underscore (_) and are limited to `A-Z`, `a-z`, `0-9`, or `_` characters.
- You cannot duplicate labels in the same pattern.
- A vector label cannot duplicate any item in the import declaration.
- Local labels are labels that have not been exported.
- The Digital Pattern Editor assumes that any label you reference but do not specify in a pattern is an imported label.

Opcode Conditions

The following conditions apply to opcodes:

- The vector can contain opcodes the digital pattern instrument processes when the vector executes. Some opcodes include parameters.

Pin State Conditions

The pin_item<n> identifier can include one of the following optional format specifiers for pin groups, for example, `PIN2:u` or `PIN2:x`.

Format Specifier	Description	Symbols	Notes
`:b`	Binary or symbolic; default format	• 0 — Drive zero. • 1 — Drive one. • L — Compare low. • H — Compare high. • X — Do not drive; mask compare. • M — Compare midband, not high or low. • V — Compare high or low, not midband; store results from capture functionality if	For binary data, the length of the binary data sequence must match the width of the pin item as defined by the pin definition. The exception to this rule is a single – character, which implies a data value of – for all

Format Specifier	Description	Symbols	Notes
		configured. • D — Drive data from source functionality if configured. • E — Compare data from source functionality if configured. • - — Repeat previous cycle. Do not use a dash (–) for the pin state on the first vector of a pattern file unless the file is used only as a target of a jump operation.  **Note** On power-up, pins are in a high impedance state. To avoid unexpected results for DUTs that are sensitive to the initial state of a pattern burst, programmatically specify the initial pin state by using the niDigital Write Static VI, the DigitalPinSet.WriteStatic .NET method, or the niDigital_WriteStatic C function. You can interactively set the initial state for a pin by selecting the corresponding driver symbol in the Drive section of the Pin View pane of the Digital Pattern Editor. After each pattern burst, the ending state of the last executed vector persists	pins in the pin item. When you are not driving data, the pins use the termination mode you select in the test program, such as High Z, Active Load, or V_{TERM}.

Format Specifier	Description	Symbols	Notes
		until the start of the next pattern burst or until you programmatically or interactively change the pin state or pin function.	
:u	Unsigned integer	• .d<decimalvalue> — Drive • .c<decimalvalue> — Compare	For decimal data, the data value when converted to binary must not exceed the width of the pin item. This restriction does not apply to leading zeros.
:x	Hexadecimal	• .d<hexvalue> — Drive • .c<hexvalue> — Compare	For hexadecimal data, the data value when converted to binary must not exceed the width of the pin item. This restriction does not apply to leading zeros.

The following conditions apply to pin state data:

- The number of pin_state items that follow the time set reference must equal the number of pin_items the pattern declaration specifies.
- For a pin_item that is a single pin, the pin_state must use one of the binary (symbolic) pin state symbols.

Note Currently, the Digital Pattern Editor does not support using system pins in patterns. To burst from a pin that spans all sites, configure a pin with a shared pin connection in the Pin Map Editor to share a channel

> across multiple sites and use that pin in your pattern instead.

- For a pin_item that is a pin group, the pin_state value width must match the number of pins defined for the pin group in the pin and channel map.

Comment Conditions

You can include comments on a single line or multiple lines separated with carriage returns before each label and vector statement. You can also include comments on the same line as a label and vector statement after the semicolon but before the end of the line. You cannot include comments on the line after the last vector statement. You cannot include comments on the same line as an edge multiplier statement. By default, the compiled, binary version of the pattern file preserves comments. To disable preserving comments in the compiled, binary version of the pattern file, use the `-no-comments` command-line option in the Digital Pattern Compiler. All comments start with double forward slashes `'//'` followed by Unicode characters.

Related reference:

- Time Set Declaration
- Import Declaration
- Pattern Declaration
- Pattern Grid View
- Opcodes
- Compiling Pattern Files

Additional Pin State Data

The vector statement specifies additional data to drive to or compare from the DUT pins using the following syntax:

```
pin1_statepin2_state …; [// comment]
```

where the pin states are the same pin states used in the vector statement. The additional pin state data row does not allow labels, time sets, or opcodes.

> **Note** Additional pin state data must be associated with a vector statement.

Additional Pin State Data Conditions

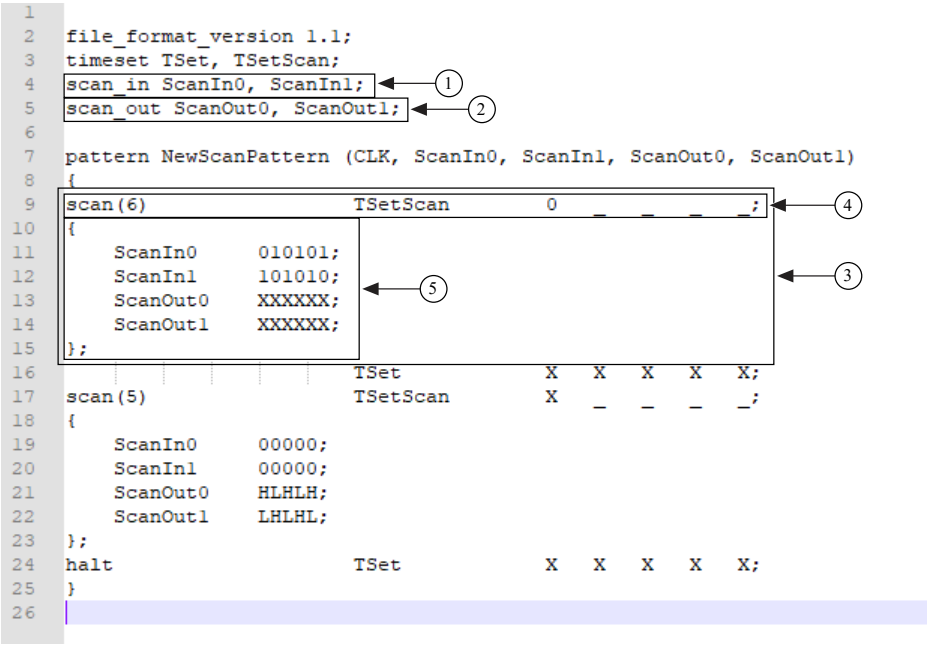
The following conditions apply to the additional pin state data:

- A terminating semicolon (;) indicates the end of the additional vector data.
- Only define additional pin state data for pin items on vectors that use the edge multiplier feature. Use whitespace as an optional placeholder for pin items that do not require additional pin state data.
- For a pin_item in a pin group that contains a mixture of edge multiplier values, only define additional pin state data for pins on the vector within the pin group that uses the edge multiplier feature. You must use an underscore (_) as a placeholder for pins within a pin group that do not require additional pin state data.
- The pin state combination must be valid for the entire vector. When using the edge multiplier feature, you only can use drive states with drive states (00, 01, 10, 11), non-driving pin states with non-driving pin states (LL, LH, HL, HH), or doubled states (XX, VV, DD, MM, EE, --).

Comment Conditions

Similar to vector statements, you can include comments on the same line as the additional pin state data after the semicolon but before the end of the line. You cannot include comments on the line between the additional pin state data and the vector statement.

Scan Pattern File Syntax



1. Scan input pins declaration
2. Scan output pins declaration
3. Scan vector statement
4. Scan vector template row
5. Scan vector data

Scan Input Declaration

The scan input declaration specifies the list of scan input pins in the pattern using the following syntax:

```
scan_in scanInPin0, scanInPin1, …;
```

where scanInPin<n> is a valid DUT pin in the pattern.

Conditions

The following conditions apply to the scan input declaration:

- A pin cannot be defined as both a scan input and a scan output pin.
- You cannot duplicate items in a scan input declaration.
- The pin must be a valid pin used in this pattern.

- Scan input pins must use 0, or 1 when defining pin state behavior in the scan data block.
- Scan input pins can use any normally usable pin state on parallel vectors.

Scan Out Declaration

The scan output declaration specifies the list of scan output pins in the pattern using the following syntax:

```
scan_out scanOutPin0, scanOutPin1, …;
```

where scanOutPin<n> is a valid DUT pin in the pattern.

Conditions

The following conditions apply to the scan output declaration:

- A pin cannot be defined as both a scan input and a scan output pin.
- You cannot duplicate items in a scan output declaration.
- The pin must be a valid pin used in this pattern.
- Scan output pins must use L, H, or X when defining pin state behavior in the scan data block.
- Scan output pins can use any normally usable pin state on parallel vectors.

Scan Vector Statement

The scan vector statement specifies the label for the vector, the scan opcode, the timeset that contains the timing information, and the data to drive to or compare from the DUT pins separated into two sections, the scan vector template row and the scan data block using the following syntax:

```
[label:] [scan opcode] timesetpin1_statepin2_state …; [// comment]
```

where the required timeset is a valid identifier the time set declaration defines or a single dash (–) to indicate to repeat the previous time set and pin<n>_state represents the pin states.

Scan Vector Statement Conditions

The conditions for standard vector statements apply to scan vector statements. The following additional conditions apply to scan vectors:

- You must use the scan opcode.
- The number of scan cycles, as defined by the scan opcode, must be greater than zero.

Scan Vector Template Row Conditions

The following additional conditions apply to the scan vector template row:

- Scan pins must use underscore (_) as a placeholder as their pin state data for the scan data template row. Scan data cannot be defined in the scan vector template row. To define behavior for scan pins, use the scan vector scan data block.
- Parallel pins define their pin state behavior in the scan vector template row. You must edit the pin state values of parallel pins in scan vector template rows.

Scan Vector Scan Data Block Conditions

The following additional conditions apply to the scan vector scan data block:

- Parallel pins cannot be defined in the scan vector scan data block. To define behavior for parallel pins, use the scan vector template row.
- Pins that define scan data must be scan pins used in the pattern.
- Scan data is defined and executed horizontally on a per pin basis.
- Pins cannot be duplicated in the scan vector scan data block.

Identifiers and Reserved Keywords

Identifiers are strings that describe labels, pattern names, opcodes, and time sets. Names must begin with an ASCII letter or underscore (_) and are limited to `A-Z`, `a-z`, `0-9`, or _ characters.

The following keywords are currently reserved identifiers. Future releases might include additional reserved identifiers.

- call
- capture
- capture_start
- capture_stop
- clear_seqflag
- clear_signal
- edge_multiplier
- end_loop
- event[0-3]
- export
- exit_loop
- exit_loop_if
- failed
- file_format_version
- halt
- import
- jump
- jump_if
- keep_alive
- keep_alive_pattern
- match
- matched
- pattern
- pulse_signal
- reg[0-15]
- repeat
- reset_trigger
- return
- seqflag[0-3]
- scan
- scan_in
- scan_out
- set_loop
- set_seqflag
- set_signal
- source

- source_d_replace
- source_start
- timeset
- write_reg

Opcodes

Opcodes are optional commands, or vector instructions, that execute on a per-vector basis after the vector executes and can change the execution of the pattern or the interaction with external instruments. Some opcodes include parameters.

Insert opcodes after the optional label and before the time set reference in a vector statement. The type-ahead feature in the pattern document displays opcodes and parameters the Digital Pattern Editor supports. You can combine certain opcodes on a single vector by separating the opcodes with a comma.

The following table lists the opcodes that the Digital Pattern Editor supports. Refer to the ***Related reference*** section at the end of this topic for a link to more information about opcodes.

Category	Supported Opcodes
Flow Control	• repeat • jump • jump_if • set_loop • end_loop • exit_loop • exit_loop_if • call • return • keep_alive • match • halt • scan

Category	Supported Opcodes
Sequencer Flags and Registers	• `set_seqflag` • `clear_seqflag` • `write_reg`
Signal and Trigger	• `set_signal` • `pulse_signal` • `clear_signal` • `reset_trigger`
Digital Source and Capture	• `capture_start` • `capture` • `capture_stop` • `source_start` • `source` • `source_d_replace`

Related reference:

- Flow Control Opcodes
- Pattern Responses to Comparisons
- Sequencer Flags and Registers Opcodes
- Signal and Trigger Opcodes
- Digital Source and Capture Opcodes

Flow Control Opcodes

Jump

Jump opcodes are analogous to if/else operations.

Syntax	Parameters	Descriptions
`jump(label)`	label is any local label within the pattern file or any label exported from another pattern file.	Executes the current vector and then executes the vector

Syntax	Parameters	Descriptions
		the label specifies. Execution continues from the vector the label defines.

`jump_if([!]failed, label)`
 	
`jump_if([!]seqflag, label)`
 	
`jump_if([!]matched, label)`
 	
`jump_if([!]trigger,`
	label is any local label within the pattern file or any label exported from another pattern file.
	
	Valid values for seqflag include seqflag0, seqflag1, seqflag2, and seqflag3.
	
	The optional `!` character inverts the `failed`, seqflag, or `matched` parameter.
	
	Valid values for trigger include trig0, trig1, trig2, and trig3.
	
	Note `matched` is not the same evaluation as `!failed`, and `!matched` is not the same evaluation as `failed`. Refer to the descriptions of the `matched` and `failed` parameters for more information.
	
	Note When you use the `match` opcode or the `matched` or `failed` parameter of the `jump_if` or `exit_loop_if` opcodes in a pattern that you burst on multiple synchronized digital pattern instruments, you must use the niDigital Enable Match Fail Combination VI, EnableMatchFailCombination .NET method, or niDigital_EnableMatchFailCombination C function to enable each synchronized instrument to process the comparison results. You must also use the NI-Sync driver and a PXIe-6674T timing and synchronization instrument to combine comparison
		Executes the current vector and then jumps to the specified label when the first parameter is true. You can also use an optional `!` operator to invert the parameter.
		
		Note Refer to the ***Related reference*** section at the end of this topic for a link to more information about pattern responses to comparisons.
		
		- `failed` - Jump if any comparison failed in the current pattern burst on any pin from any site. Continue to the next vector otherwise. This condition does not include the vectors that executed within the last 80 cycles or vectors that contain the `match` opcode. You can use the `repeat` opcode to implement an 80 cycle delay.
		
		Note Refer to

Syntax	Parameters	Descriptions
	results across digital pattern instruments synchronized using NI-TClk. Refer to the ***Related reference*** section at the end of this topic for a link to more information about using matched and failed functionality across multiple synchronized digital pattern instruments.	the ***Related information*** section at the end of this topic for a link to more information about synchronizing multiple instruments and using `failed` parameters on vectors that span more than one instrument or more than one site. • seqflag - Jump if the specified pattern sequencer flag is set. Continue to the next vector otherwise.  **Note** When you use the seqflag parameter for pins that span multiple digital pattern instruments, you must use NI-TClk to

Syntax	Parameters	Descriptions
		synchronize the instruments. Refer to the ***Related reference*** section at the end of this topic for a link to more information about using sequence flag triggers and synchronizing digital pattern instruments. • `matched` - Jump if the vector that executed exactly 80 cycles prior matched. Continue to the next vector otherwise. **Note** Refer to the ***Related reference*** section at the end of this topic for a link to more information about synchronizing multiple instruments and using

Syntax	Parameters	Descriptions
		`matched` parameters on vectors that span more than one instrument or more than one site. • `trigger` - Jump if the specified trigger is asserted. Continue to the next vector otherwise. **Note** When you use the `trigger` parameter for pins that span multiple digital pattern instruments, you must use NI-TClk to synchronize the instruments. Refer to the ***Related reference*** section at the end of this topic for a link to more information about using

Syntax	Parameters	Descriptions
		conditional jump triggers and synchronizing digital pattern instruments.

Example

```
sample_timeset L; // For the comments below, assume that no failed comparisons have
occurred before this vector.
// This vector will not fail even if the previous vector evaluates to false
// because it does not meet the required minimum number of cycles to wait.
jump_if(failed, error)     sample_timeset X;
continue: repeat(78)       sample_timeset X;
// Pipeline delay must be at least 80 cycles after the last failure.
// If it is less than 80, the !failed condition is always true.
// If it is 80 or greater, the !failed condition reflects the result of the
comparison.
// The jump to the end label executes if the !failed condition is true.  If the
failed condition is asserted
// because of a comparison 80 cycles ago, the pattern does not branch to the end
label and continues to the next
// vector with the error label.
jump_if(!failed, end)      sample_timeset X;
error: halt                sample_timeset X;
end: halt                  sample_timeset X;
```

Repeat, Loop, and Subroutine

Subroutine opcodes are analogous to function calls.

Syntax	Parameters	Description

`repeat(numeric)`
 	
`repeat(register)`
	numeric is the repeat count. Valid values are 1 - 65535.	Executes the vector the number of times the parameter specifies. If t parameter is a register,

Syntax	Parameters	Description
	register is the register from which to read the repeat count. Valid values for register include reg0 - reg15.	opcode reads the repea count from the specifie register. Use pattern sequencer registers to numeric values betwee pattern sequencer and time test program.

`set_loop(numeric)`
 	
`set_loop(register)`
	numeric is the number of times to repeat the loop. Valid values are 1 - 65535.
	
	register is the register from which to read the number of times to repeat the loop. Valid values for register include reg0 - reg15.
		Executes the current ve and then specifies the requested number of lo iterations for the next lo the pattern. If the paran is a register, the opcode the requested number loop iterations from the specified register to allo run-time test program control the number of t a loop executes.
		
		After you call the `set_` opcode, you must spec first and last vectors in loop. Create a label on vector at which to start iteration of the loop. Ca `end_loop` opcode on vector at which to end iteration of the loop. Us label you created for th vector of the loop as th parameter of the `end_` opcode. The digital pat instrument supports ne loops up to 8 levels dee number of nested loop validated only at run tir not at edit or compile t

`end_loop(label)`	label must be a local label.	Marks the last vector in loop. Executes the vect

Syntax	Parameters	Description
		then compares the completed loop count t number of iterations th `set_loop` opcode spe for the loop. If the com loop count is less than number of iterations specified, the pattern ju to the specified label, a loop iterates. If the loop count equals the numb iterations specified, the completes, the loop co stack is popped, and th vector after the `end_l` opcode executes. The c pattern instrument sup nested loops up to 8 lev deep. The number of n loops is validated only time, not at edit or com time. You can optionally use `exit_loop` and `exit_loop_if` opco exit the loop before the requested number of iterations complete.
`exit_loop(label)`	label is any local label within the pattern file or any label exported from another pattern file.	Executes the current ve pops the most nested l count from the loop co stack, and jumps to the you specify. You can optionally use `exit_loop` and `exit_loop_if` opco exit the loop before the

Syntax	Parameters	Description
		requested number of iterations complete.

`exit_loop_if([!]failed, label)`
 	
`exit_loop_if([!]seqflag, label)`
 	
`exit_loop_if([!]matched, label)`
 	
`exit_loop_if([!]trigger, label)`
	label is any local label within the pattern file or any label exported from another pattern file.
	
	Valid values for seqflag include seqflag0, seqflag1, seqflag2, and seqflag3.
	
	Valid values for trigger include trig0, trig1, trig2, and trig3.
	
	The optional `!` character inverts the `failed`, seqflag, or `matched` parameter.
	
	Note `matched` is not the same evaluation as `!failed`, and `!matched` is not the same evaluation as `failed`. Refer to the descriptions of the `matched` and `failed` parameters for more information.
	
	Note When you use the `match` opcode or the `matched` or `failed` parameter of the `jump_if` or `exit_loop_if` opcodes in a pattern that you burst on multiple synchronized digital pattern instruments, you must use the niDigital Enable Match Fail Combination VI, EnableMatchFailCombination .NET method, or niDigital_EnableMatchFailCombination C function to enable each synchronized instrument to process the comparison results. You must also use the NI-Sync driver and a PXIe-6674T timing and synchronization
		Executes the current ve and then conditionally the loop, based on the condition you specify. I condition is true, the lo counter stack is poppe the pattern execution j to the vector the label specifies. If the conditi false, the next vector in loop executes.
		
		You can optionally use `exit_loop` and `exit_loop_if` opco exit the loop before the requested number of iterations complete.
		
		Note Refer to ***Related reference*** section at the of this topic f link to more information a pattern respo to comparis
		
		• `failed` - Exits the if any comparison in the current patt burst on any pin fr any site. Continue next vector otherw

Syntax	Parameters	Description
	instrument to combine comparison results across digital pattern instruments synchronized using NI-TClk.	This condition doe include the vectors executed within th 80 cycles or vector contain the `match` opcode. You can us `repeat` opcode to implement an 80 c delay. **Note** Re the ***Rela informa*** section a end of th topic for link to m informat about synchron multiple instrume and using `failed` paramete on vecto that spar more tha one instrume more tha one site. • seqflag - Exits the l the specified patte sequencer flag is s Continue to the ne vector otherwise.

Syntax	Parameters	Description
		Note Wh you use t seqflag paramete pins that multiple digital pa instrume you must NI-TClk t synchron the instrume Refer to t ***Related*** ***referen*** section a end of th topic for link to m informat about us sequence triggers a synchron digital pa instrume • `matched` - Exits th loop if the vector t executed exactly 8 cycles prior match Continue to the ne vector otherwise. **Note** Re the ***Rela*** ***referen***

Syntax	Parameters	Description
		section a end of th topic for link to m informat about synchron multiple instrume and usin matche paramet on vecto that spar more tha one instrume more tha one site. • trigger - Terminate loop if the specifie trigger is asserted. Continue to the en loop iteration othe **Note** Wh you use t trigge paramet pins that multiple digital pa instrume you must NI-TClk t synchron the

Syntax	Parameters	Description
		instrume Refer to t ***Related referen*** section a end of th topic for link to m informat about us condition jump trig and synchron digital pa instrume
`call(label)`	label is any local label within the pattern file or any label exported from another pattern file.	Executes the current ve and then jumps to the subroutine the label specifies. The subrouti executes until it reache `return` opcode and th jumps back to the next after the vector with a `c` opcode. The digital pat instrument supports ne subroutine calls up to 8 deep. The Digital Patter Editor does not validate compile time or edit tir total number of nested subroutine calls in the patterns you load again number of nested subr calls the digital pattern instrument supports.

Syntax	Parameters	Description
		The vector immediately before a vector with a `c` opcode can use only `ca` `match` opcodes. You can use a `call` op on the first vector but n the last vector in a patt
`return`	--	Executes the current ve and then jumps to the v that follows the vector the corresponding call opcode.

Example 1

```
set_loop(100)                sample_timeset X; // Set the number of loop iterations
for the looping operation.
loop:                        sample_timeset X; // Set the label for the looping
operation.
exit_loop_if(seqflag0, exit) sample_timeset X; // Exit the loop prematurely if
seqflag0 is set to true.
end_loop(loop)               sample_timeset X;
exit: halt                   sample_timeset X;
```

Example 2

```
call(sub1) sample_timeset 0 L; // This vector executes first.
halt       sample_timeset 1 H; // This vector executes after the subroutine
returns.
...
sub1:      sample_timeset 0 L;
return     sample_timeset 0 L;
```

Keep Alive

Syntax	Parameters	Description
`keep_alive`	--	Stops the pattern burst and jumps to the keep alive pattern loaded on the digital pattern instrument. A keep alive pattern is a specialized, simple looping pattern designed to keep the DUT from unlocking clocks and PLLs while you perform other tasks, such as loading or unloading patterns, changing time sets, changing instrument configurations, or moving from one test to another in the test program. Use the `keep_alive` opcode instead of the `halt` opcode for these types of situations. You must use the `keep_alive` opcode on the last vector of a keep alive pattern to loop the keep alive pattern until you burst another pattern or until you abort the keep alive pattern.

Example

```
pattern RegularPattern(Clk,DIO1)
{
repeat(100) sample_timeset 1 1;
            sample_timeset 1 H;
repeat(100) sample_timeset 1 0;
keep_alive  sample_timeset 1 L; //Jumps to keep alive pattern
}

keep_alive_pattern KeepAlivePattern(Clk)
{
```

```
//Exit the keep alive pattern by bursting a new pattern
//or by calling the niDigital Abort Keep Alive VI or
//the DigitalPatternControl.AbortKeepAlive .NET method.
keep_alive  -              -;
}
```

Scan

Syntax	Parameters	Description
`scan(cycle_count)`	`cycle_count` is the number of scan cycles in this scan vector. The value must be a positive integer.	Designates a vector as a scan vector and declares the number of scan cycle subrows each scan vector has. **Note** Refer to the ***Related reference*** section at the end of this topic for a link to more information about scan text pattern file syntax.

Example

```
pattern NewScanPattern (CLK, ScanIn0, ScanIn1, ScanOut0, ScanOut1)
{
scan(6)
TSetScan          0 _ _ _ _;
{
                              ScanIn0    010101;
                        ScanIn1    101010;
                              ScanOut0   LHLHLH;
                              ScanOut1   HLHLHL;
};
```

Match

Syntax	Parameters	Description
`match`	--	Evaluates comparisons

Syntax	Parameters	Description
		(H, L, M, or V) on the vector without affecting the fail count and pass or fail results. **Note** Refer to the ***Related reference*** section at the end of this topic for a link to more information about pattern responses to comparisons. This opcode generates a `matched` (or `!matched`) condition after exactly 80 cycles. You can only use the `matched` condition on the 80th cycle with the `jump_if` or `exit_loop_if` opcodes. On vectors with the `match` opcode, all comparisons are evaluated to determine the state of the `matched` condition. You can use a `repeat` opcode to implement an 80 cycle delay. To wait for a DUT response, you can create a pipeline of
	Note When you use the `match` opcode or the `matched` or `failed` parameter of the `jump_if` or `exit_loop_if` opcodes in a pattern that you burst on multiple synchronized digital pattern instruments, you must use the niDigital Enable Match Fail Combination VI, EnableMatchFailCombination .NET method, or niDigital_EnableMatchFailCombination C function to enable each synchronized instrument to process the comparison results. You must also use the NI-Sync driver and a PXIe-6674T timing and synchronization instrument to combine comparison results across digital pattern instruments synchronized using NI-TClk.	

Syntax	Parameters	Description
		matches by using the `match` opcode multiple times within a loop to poll for a certain condition. **Note** Refer to the ***Related reference*** section at the end of this topic for a link to more information about synchronizing multiple instruments and using `matched` parameters on vectors that span more than one instrument or more than one site.

Example

```
match                   sample_timeset L;
repeat(79)              sample_timeset X;
jump_if(matched, end)   sample_timeset X; // Pipeline delay must be exactly 80
cycles after the match opcode.
Jump(error)             sample_timeset X;
end: halt               sample_timeset X;
```

Halt

Syntax	Parameters	Description
`halt`	--	Stops the execution of the pattern sequencer and the pattern burst. When the pattern sequencer halts, all the pins maintain their current programmed state. The pattern sequencer does not automatically stop execution at the end of the pattern. You must specify the `halt` opcode on the last vector in the pattern to stop the execution of the pattern. Undefined behavior results in patterns that complete without a halt code.

Related reference:

- Sequencer Flags and Registers Opcodes
- Pattern Responses to Comparisons
- Pattern Declaration
- Scan Pattern File Syntax

Sequencer Flags and Registers Opcodes

Syntax	Parameters	Description
`set_seqflag(seqflag_list)`	seqflag_list is a comma-separated list of sequencer flags. You cannot repeat flags. Valid values include seqflag0, seqflag1, seqflag2, and seqflag3.	Sets the specified pattern sequencer flag(s). Use sequencer flags to coordinate execution between the pattern sequencer and a test program at run time.
`clear_seqflag(seqflag_list)`	seqflag_list is a comma-	Clears the specified pattern

Syntax	Parameters	Description
	separated list of sequencer flags. You cannot repeat flags. Valid values include seqflag0, seqflag1, seqflag2, and seqflag3.	sequencer flag(s). Use sequencer flags to coordinate execution between the pattern sequencer and a test program at run time.
`write_reg(register, numeric)`	Valid values for register include reg0 - reg15. numeric is the data (16 bit) in decimal format. Valid values are 0 – 65535.	Writes to one of the pattern sequencer registers. Use pattern sequencer registers to pass numeric values between the pattern sequencer and a run-time test program.

Example 1

```
clear_seqflag(seqflag3, seqflag0)   sample_timeset  0 L;
```

Example 2

```
set_seqflag(seqflag2, seqflag1)     sample_timeset  1 H;
```

Example 3

```
write_reg(reg5, 45562)              sample_timeset  0 L;
```

Signal and Trigger Opcodes

Use signal and trigger opcodes to send event signals or conditional triggers from the pattern to other digital pattern instruments by using PXI trigger lines.

Syntax	Parameters	Description
`set_signal(event_list)`	event_list is a comma-separated list of events. You cannot duplicate events. Valid values include event0, event1, event2, and event3	Sets the specified events to the high (asserted) state and holds that state until the pattern executes the `clear_signal` opcode.

Syntax	Parameters	Description
		You can route events to PXI trigger lines to coordinate execution with other systems or instrumentation. To route events, use the niDigital Export Signal VI, the NIDigital.ExportSignal .NET method, the niDigital_ExportSignal C function, or the Export Pattern Opcode Events Signals section of the Instrument Settings pane of the Digital Pattern Editor.
`pulse_signal(event_list)`	event_list is a comma-separated list of events. You cannot duplicate events. Valid values include event0, event1, event2, and event3	Asserts and deasserts the specified events. Use this opcode when you want to pulse an event for a short period of time. You can route events to PXI trigger lines to coordinate execution with other systems or instrumentation. To route events, use the niDigital Export Signal VI, the NIDigital.ExportSignal .NET method, the niDigital_ExportSignal C function, or the Export Pattern Opcode Events Signals section of the Instrument Settings pane of the Digital Pattern Editor.
`clear_signal(event_list)`	event_list is a comma-separated list of events. You	Sets the specified events to the low (deasserted) state.

Syntax	Parameters	Description
	cannot duplicate events. Valid values include event0, event1, event2, and event3	You can route events to PXI trigger lines to coordinate execution with other systems or instrumentation. To route events, use the niDigital Export Signal VI, the NIDigital.ExportSignal .NET method, the niDigital_ExportSignal C function, or the Export Pattern Opcode Events Signals section of the Instrument Settings pane of the Digital Pattern Editor.
`reset_trigger(trigger_list)`	trigger_list is a comma-separated list of triggers. You cannot duplicate triggers. Valid values include trig0, trig1, trig2, and trig3	Sets the specified triggers to the deasserted state and rearms the trigger for latching the next assertion. Use `reset_trigger` to reset trigger states before conditional branches in a pattern using `jump_if` or `exit_loop_if` to ensure the pattern predicates execution on changes from a known state.

Example

```
set_signal(event3, event0)       sample_timeset  0 L;
clear_signal(event3, event0)     sample_timeset  1 H;
halt                             sample_timeset  0 L;
```

Related reference:

- Instrument Settings Pane

Digital Source and Capture Opcodes

Syntax	Parameters	Description
`capture_start(waveformName)`	waveformName is the name of the capture waveform. The name can begin with a letter or underscore (_) and is limited to `A-Z`, `a-z`, `0-9`, or _ characters. You can use up to 512 capture waveforms on the digital pattern instrument at a time. When you load the pattern, an error results when the combined loaded patterns include more than 512 waveforms.	Starts capturing a waveform as configured in the test program with the NI-Digital Source/Capture API or waveform file (`.digicapture`).
`capture`	--	Stores pin state data for each pin specified in the capture waveform. In parallel mode, the `capture` opcode stores one sample. In serial mode, each serial capture engine can be configured for one specific pin. On vectors with the `capture` opcode, the value on that pin is shifted in until the number of bits you specified for the sample width have been shifted, at which point the sample is stored. Always use a pattern value of `V` for the pin(s) being captured. Capture behavior is undefined for any value other than `V`. Refer to the instrument specifications for more

Syntax	Parameters	Description
		information about the number of samples you can capture for all patterns in a burst operation. You can combine the `capture` opcode with the following opcodes on a single vector by separating the opcodes with a comma: • `capture_stop` • `clear_seqflag` • `clear_signal` • `end_loop` • `match` • `repeat` • `set_loop` • `set_seqflag` • `set_signal` • `source` • `source_start` • `source_d_replace`
`capture_stop`	--	Ends capture of the waveform initiated by the `capture_start` opcode on a previous vector.
`source_start(waveformName)`	waveformName is the name of the capture waveform. The name can begin with a letter or underscore (_) and is limited to `A-Z`, `a-z`, `0-9`, or _ characters. You can use up to 512 capture waveforms on the digital pattern instrument at a time. When you load the pattern, an error results	Starts sourcing a waveform as configured in the test program with the NI-Digital Source and Capture API or waveform file (`.tdms`). **Note** A required delay of 3 µs must pass between the

Syntax	Parameters	Description
	when the combined loaded patterns include more than 512 waveforms.	start of the vector that uses this opcode and the subsequent start of a vector that uses a `source` opcode or `source_start` opcode. You can create and reserve a time set with a known period to meet the delay requirement.
`source`	--	Uses source data to replace pin state data for each `D` pin state in the vector. In parallel mode, the `source` opcode sends one sample. In serial mode, each serial source engine can be configured for one specific pin. On vectors with the `source` opcode and a `D` for that pin, the bit is shifted in until the number of bits you specified for the sample width have been shifted. When the number of bits you specified for the sample width have been shifted, execution moves on to the next sample. Waveforms behave as a circular buffer, meaning that when execution reaches the end of the waveform, it returns

Syntax	Parameters	Description
		to the beginning of the waveform automatically. You can combine the `source` opcode with the following opcodes on a single vector by separating the opcodes with a comma: • `capture` • `capture_start` • `capture_stop` • `clear_seqflag` • `clear_signal` • `end_loop` • `match` • `repeat` • `set_loop` • `set_seqflag` • `set_signal`
`source_d_replace (sourcePinState0, sourcePinState1)`	sourcePinState0 specifies the pin state to replace source memory `0`. Valid values include 0, 1, L, H, X. sourcePinState1 specifies the pin state to replace source memory `1`. Valid values include 0, 1, L, H, X.	Changes the behavior of source memory values `0` and `1` for all subsequent vectors with the source opcode and a D for that pin. By default, a `0` in the source data translates to pin state 0, or drive logical low, and a `1` in the source data translates to pin state 1, or drive logical high. You cannot combine the `source` opcode with the `source_d_replace` opcode. Include at least one vector of separation between `source_start` and any source opcode that uses `source_d_replace`

Syntax	Parameters	Description
		to replace source pin states. You must include a 3 µs delay from the vector with `source_start` to the vector before the `source` opcode.

Example 1

```
source_start(source_waveform)   sample_timeset  X X;
                                timeset_3us     X X; // Reserved time set
configured with a known period (3 us) to meet the delay requirement.
source                          sample_timeset  D H;
halt                            sample_timeset  X X;
```

Example 2

```
capture_start(waveform_name)     sample_timeset  X X; // Start capture waveform
with a capture waveform name of 'waveform_name'.
repeat(100), capture             sample_timeset  V 1; // Capture with a repeat
opcode.
capture                          sample_timeset  V 1; // Capture on a single
vector.
capture_stop                     sample_timeset  X X; // End capture from the
waveform.
halt                             sample_timeset  X X;
```

Example 3

```
capture_start(capture_waveform) sample_timeset  X X; // Start capture waveform with
a capture waveform name of 'capture_waveform'.
source_start(source_waveform)   sample_timeset  X X;
                                timeset_3us     X X; // Reserved time set
configured with a known period (3 us) to meet the delay requirement for
source_start.
repeat(65535), source, capture  sample_timeset  D V; // Source and capture with a
repeat opcode.
repeat(65535), source, capture  sample_timeset  D V; // Source and capture with a
repeat opcode.
```

```
repeat(65535), source, capture  sample_timeset  D V; // Source and capture with a
repeat opcode.
repeat(65535), source, capture  sample_timeset  D V; // Source and capture with a
repeat opcode.
repeat(65535), source, capture  sample_timeset  D V; // Source and capture with a
repeat opcode.
capture_stop                    sample_timeset  X X; // End capture from the
waveform.
halt                            sample_timeset  X X;
```

Related reference:

- Source Waveform Configurations
- Capture Waveform Configurations

Pattern Responses to Comparisons

Comparisons evaluate the output of the DUT at the strobe edge you specify for the time set of the vector. Failures for a vector result when any DUT response on any pin on any cycle does not match the expected value. All failures are logically combined into one matched or failed condition the sequencer can use. This functionality spans multiple digital pattern instruments when you use a PXIe-6674T configured to aggregate failure information. The sequencer executes vectors from memory, which includes information about the expected states of each pin. The delay of the vectors passing through stimulus conditioning circuitry, cabling/interconnect delays to and from the DUT, DUT response delays, and response conditioning circuitry affect the failure or match evaluation in a conditional jump or call in the pattern. The required delay for a failure or match evaluation in the pattern is 80 cycles for single and multiple digital pattern instruments in the test system.

Failures

To use a comparison result for a conditional jump in the pattern based on a failure, specify the L, H, V, M, or E pin states for the pins to compare and use the `failed` or `!failed` parameter of the `jump_if` or `exit_loop_if` opcode to execute the pattern in response to the evaluation. Because of the pipeline latency, at least 80 cycles must pass between the vector on which you specify the comparison to the vector on which you use the `jump_if` or `exit_loop_if` opcode. If any comparison on any pin results in a failure, the failed result remains true for all subsequent vectors

until the next pattern burst clears it.

Matches

Use the `match` opcode on a vector to evaluate parameters on specific pins. Specify the L, H, V, M, or E pin states for the pins to compare. For example, if a DUT has a `PLL_Locked` and a `Ready` output and you want to wait until both outputs assert, specify H as the pin state for these pins, set all other pins in the pattern to X, and loop the pattern until the `PLL_Locked` and the `Ready` pins both match H, at which point the vector is flagged as a match. For example:

```
pattern match_loop(PLL_Locked, Ready)
{
set_loop(1000)                                          sample_timeset X X; // Set
the timeout to 1,000 iterations.
repeat(80), match                                       sample_timeset H H; //
Repeat the required minimum number of cycles to wait.
loop_begin: exit_loop_if(matched, loop_exit), match     sample_timeset H H;
end_loop(loop_begin), match                             sample_timeset H H;
jump(timeoutPattern)                                    sample_timeset X X; //If
this vector executes, the loop timed out.
loop_exit: halt                                         sample_timeset X X; //Start
of the pattern to execute after the match.
...
}
```

To use a comparison result for a conditional jump in the pattern based on a match, use the `matched` or `!matched` parameter of the `jump_if` or `exit_loop_if` opcode to execute the pattern in response to the evaluation. Because of the pipeline latency, exactly 80 cycles must pass between the vector on which you specify the comparison to the vector on which you use the `jump_if` or `exit_loop_if` opcode. All comparisons on all pins must match to result in a match. The match result represents the evaluation from the vector that executed exactly 80 cycles ago.

Related reference:

- Flow Control Opcodes

Importing Patterns

Select **File » Import Pattern** to import text pattern files (`.digipatsrc` or `.digipat.gz`) into the Digital Pattern Editor. You must include an active pin and channel map file in the project before you can import a pattern or the editor prompts you to activate one.

Importing a text pattern file compiles the file into a binary version of the pattern file (`.digipat`) and adds it to the current project. All subsequent editing of the pattern file in the pattern document operates on the binary file (`.digipat`). You do not modify the text version of the pattern file (`.digipatsrc`), and the `.digipatsrc` file remains unchanged unless you overwrite it when you export the pattern.

When you select a single file in the Open file dialog box, you can use the corresponding Save As dialog box to specify a filename and destination location for the compiled, binary version of the pattern file. When you select multiple files in the Open file dialog box, you can use the Browse For Folder dialog box to specify a destination location for the compiled, binary versions of the pattern files. You cannot specify filenames in this situation. The editor uses the same name as the source filename but with a `.digipat` file extension.

Alternatively, you can use the Digital Pattern Compiler command-line tool (`DigitalPatternCompiler.exe`), located in `<Program Files>\National Instruments\Digital Pattern Compiler`, to compile text pattern files (`.digipatsrc` or `.digipat.gz`) to binary pattern files (`.digipat`).

Related reference:

- Text Pattern File Syntax
- Compiling Pattern Files
- Generating Patterns
- Exporting Patterns

Compiling Pattern Files

When you import a pattern file into the Digital Pattern Editor, the editor automatically compiles the file. You can also use the Digital Pattern Compiler command-line tool (`DigitalPatternCompiler.exe`), located in `<Program Files>\National Instruments\Digital Pattern Compiler`, to compile text pattern files

(`.digipatsrc` or `.digipat.gz`) to binary pattern files (`.digipat`). You must specify a pin and channel map file to compile text pattern files to binary pattern files. To compile compressed text pattern files (`.digipat.gz`), use the command-line argument `gzip-digipatsrc`.

Compiler Capabilities

Call the Digital Pattern Compiler with the `-help` option to list the available command-line options for the tool, which include commands to complete the following tasks:

- Perform syntax checking.
- Exclude comments in the target file.
- Decompile binary pattern files to text pattern files or compressed text pattern files to store in a source code control system or to use with a third-party application to compare revisions of the pattern file.
- Specify the output format and location for the target file.
- Compile a batch of files with one command.
- Deco

Related reference:

- Text Pattern File Syntax
- Generating Patterns
- Exporting Patterns

Editing Documents

Editing Values

As you enter or edit a value, the field displays blue text with a blue border. To commit the value, press <Enter>, <Tab>, or click another location in the view. To revert the change, press <Esc>. The field displays red text with a red border and a tooltip to indicate an error. To revert the change after an error, press <Esc> while the cursor is in the red highlighted field.

When you use the <Delete> key to clear the contents of a cell, the pattern document displays a dash (–) for the time set to repeat the previous time set and an `X` for the pin item values to ignore the value.

Adding and Deleting Vectors in Pattern Documents

Right-click a cell in a row in the pattern document and use the context menu to insert new vectors or scan items or delete selected vectors or scan items from the pattern. When you insert a new vector or scan item, the pattern document inserts a row and uses a dash (–) for the time set to repeat the previous time set and an X for the pin item values to ignore the value. Selecting all the cells in a row and pressing <Delete> also removes the vector or scan item.

Adding and Deleting Pin Item Columns in Pattern Documents

Right-click a cell in the pattern document and use the context menu to insert or delete pin item columns from a list of pins and pin groups that exist in the active pin map file. When you insert a new pin item column, the pattern document inserts the column to the left of the cell you selected and uses X for the pin item values. Pins can appear in a pattern only once. You cannot add pin groups that contain pins that already exist in the pattern until you remove the existing pin references from the pattern.

Change Pin Type

Right-click a pin and select **Change Pin Type** from the context menu to change the pin type to **Scan Input**, **Scan Output**, or **Input/Output (default)**.

Reorder Pin Item Columns in Pattern Documents

Right-click a cell in the pattern document and use the context menu to move selected pin item columns left or right. You cannot reorder pins in a pin group from the editor.

Filling and Inverting Pin State Data

Select and right-click pin state data cells and then select **Fill Pin States**, **Invert Pin State Logic**, or **Invert Pin State Drive/Compare** from the context menu to fill or invert the cells you selected. Select from the list of valid pin state values to fill the cells. Inverting cells inverts only drives and high and low compares and leaves all other pin state data cells unchanged. Selecting pin group cells applies the fill or inversion operation to all pins in the selected pin group.

Updating Pin Group Pins from Pin Map

Right-click in the pattern document and select **Update Pin Group Pins from Pin Map** from the context menu to update the contents of all pin groups in the pattern to match the contents of the pin groups as defined in the active pin map file. Use this technique to update the ordering of a pin group or to add or remove pins from a pin group to match changes made in the active pin map file. The editor moves pins that have been removed from the pin group in the active pin map file out of the pin group column, but the pins remain in the pattern to preserve the pin states. Delete the pin item column if you want to remove the pin from the pattern.

Locating Vectors in Pattern Documents and History RAM View

Click the **Find in Pattern** button on the document toolbar or press <Ctrl+F> to launch a dialog box in which you can specify the text you want to search for. The document highlights cells that include the search text. Use the **Find Previous** and **Find Next** buttons in the dialog box or on the document toolbar to navigate among the results. If you close the **Find in Pattern** dialog box, you can continue to browse through results by selecting **Shift** » **F3** to see previous results or **F3** to see the next results. Enable the **Match Pin Columns** option to limit the search to a serial pattern of pin state values, such as HLHL, only in the pin or pin group data columns of the pattern. For pin groups, you must expand the columns you want to search.

Click the **Go To Vector** button on the pattern document toolbar or press <Ctrl+G> to launch a dialog box in which you can enter a vector number to highlight and locate in the pattern document. You must select a vector number in the valid range. You can specify a scan cycle in a scan vector by specifying the scan vector followed by a colon and the scan cycle, for example, vectorNumber:scanCycle.

Copying and Pasting Data in Pattern Documents

You can copy and paste rows or groups of cells in the pattern document. You can paste to and from spreadsheet applications and text editors that support tab-delimited clipboard text. The behavior of the copy and paste operation depends on what you select to copy and the target location you select for the paste operation, as described in the following table.

Select for Copy Operation	Select for Paste Operation	Resulting Behavior
Full row (vector)	Single cell	Inserts the copied vectors into the pattern as new vectors immediately above the vector in which you selected the single cell as the target location.
Contiguous rectangular set of cells	More than one cell	Pastes the copied cell values into the cells you selected as the target location, repeating the values horizontally and vertically as necessary to fill all cells in the target location selection.
Set of cells that are not all full rows	Single cell	Pastes the copied cell values into the existing vectors starting at the single cell you selected as the target location.

Saving Pattern Documents

Changes you make in the pattern document are not committed to disk until you explicitly save the file. You must save the pattern before you can load or burst it on the digital pattern instrument. The Digital Pattern Editor prompts you to save modified files before loading and bursting a pattern.

Keyboard Shortcuts within the Pattern Document

Action	Shortcut
Load a pattern	<Ctrl+L>
Burst a pattern	F5
Launch a dialog box in which you can indicate a vector number to highlight and locate in the pattern document	<Ctrl+G>
Select active editor	<Ctrl+E>
Expand all scan vectors	<Ctrl+Shift+Add Sign> **Note** Use the Add Sign (+) on the

Action	Shortcut
	numeric keyboard.
Collapse all scan vectors	<Ctrl+Shift+Minus Sign> **Note** Use the Minus Sign (-) on the numeric keyboard.

Related tasks:

- Levels

Related reference:

- Specifications
- Timing
- Pattern Grid View
- Bursting Patterns and Viewing Results
- History RAM View
- Pin Values
- Levels File XML Structure

Pattern Grid View

Click the **Grid** tab on the pattern document toolbar to launch the pattern grid view. Use this view to see components of the binary pattern file, including time sets, labels, opcodes, vector numbers, pin state data that indicates drives and compares, comments for each vector, exported labels, and comments from the top of the file. Use the pattern waveform view for a graph-based representation of the pattern. You can select and right-click vectors in the pattern grid view and select **Show Vectors in Waveform View** from the context menu to launch the pattern waveform view. You can also click the **Waveform** tab on the pattern document toolbar to launch the pattern waveform view.An asterisk (*) next to a filename in the Project Explorer window and in the document tab indicates that the file has been modified but not saved.

Configuring the Pattern

The grid view includes the following components:

- **Pattern Name** — Specifies the pattern name as defined in the pattern declaration of the pattern file. If you change the pattern name, you must manually update any other references to the pattern name. Names must begin with an ASCII letter or underscore (_) and are limited to `A-Z`, `a-z`, `0-9`, or _ characters. The pattern name is independent from the filename of the pattern file.You must explicitly export the pattern name as a global label to use it as the target of an opcode from another pattern.
- **Loaded/Not Loaded/Out of Sync** — Indicates whether the pattern is loaded on the digital pattern instrument. **Out of Sync** indicates that the pattern loaded on the instrument does not match the pattern file opened in the document.
- **Keep Alive Pattern** — Indicates that the pattern is a keep alive pattern. The toolbar includes only buttons relevant to keep alive patterns. To burst a keep alive pattern, first load the keep alive pattern and then burst a standard pattern that jumps to a keep alive pattern.
- **Passed/Failed/Disabled** — Indicates the status of compares in the last burst of the pattern or indicates that the site was disabled for the last burst of the pattern. The editor briefly highlights the Passed/Failed status icon to indicate that the pattern finished bursting.
- **Total Failures** — Displays the number of failures in the pattern.
- **Burst Complete** — Displays the timestamp of the last pattern burst.
- **Burst Count** — Displays the number of pattern bursts completed during the last ***Burst Until Pass***, ***Burst Until Fail***, or ***Burst Until Abort*** operation.
- **Comments, Exports, and Time Sets** — Displays comments, exported labels, and time sets used. Use the expander buttons to expand or collapse this section.
  - **Comments** — Specifies the comments at the top of the text pattern file. By default, importing or compiling a text pattern file (`.digipatsrc`) into a binary version of the pattern file (`.digipat`) preserves comments. To remove comments when you create a compiled, binary version of the pattern file, use the `-no-comments` command-line option in the Digital Pattern Compiler.
  - **Export Labels** — Specifies the labels the pattern exports for other patterns to use as opcode targets. You must manually update all references to the export labels when you make changes in the pattern document. You must explicitly export the pattern name as a global label to use it as the target of an opcode

from another pattern.

> **Note** The first vector in a pattern file is automatically assigned a label that matches the pattern name.

  - **Time Sets Used** — Displays the time sets used in the pattern.
- **Time Set** — Specifies an existing, defined time set to use for the vector. You can add and modify time sets directly in the pattern, but if the time set does not exist on the instrument, the Digital Pattern Editor returns an error at pattern load or run time. Do not use a dash (–) for the time set on the first vector of a pattern file unless the file is used only as a target of a jump or call operation. Refer to the ***Related reference*** section at the end of this topic for a link to more information about timing. Refer to the specifications document for the digital pattern instrument for more information about the number of supported time sets.
- **Label** — Specifies a vector label. If you rename or delete a label, you must manually update any other references to that label to avoid run-time errors.Names must begin with an ASCII letter or underscore (_) and are limited to `A-Z`, `a-z`, `0-9`, or _ characters. The Digital Pattern Editor assumes that any label you reference but do not specify in a pattern is an imported label.
- **Opcode** — Specifies the opcodes the pattern sequencer executes on the vector. You can combine certain opcodes on a single vector by separating the opcodes with a comma. Refer to the ***Related reference*** section at the end of this topic for a link to more information about opcodes.
- **Pin or Pin Group Data** — Specifies the pin state data that indicates drives and compares for pins and pin groups for the vector. Pins or pin groups you reference in the pattern must be defined in the pin and channel map.

> **Note** Currently, the Digital Pattern Editor does not support using system pins in patterns. To burst from a pin that spans all sites, configure a pin with a shared pin connection in the Pin Map Editor to share a channel across multiple sites and use that pin in your pattern instead.

  For pin groups, click the format specifier x in the column heading to change the numeric representation of the pin state data. Click the Expand › or Collapse ‹ button to view or hide the individual pins in a pin group. This column uses the following symbols:
  - 0 — Drive zero.

- 1 — Drive one.
- L — Compare low.
- H — Compare high.
- X — Do not drive; mask compare.
- M — Compare midband, not high or low.
- V — Compare high or low, not midband; store results from capture functionality if configured.
- D — Drive data from source functionality if configured.
- E — Expect data from source functionality if configured.
- - — Repeat previous cycle. Do not use a dash (–) for the pin state on the first vector of a pattern file unless the file is used only as a target of a jump or call operation.

Note On power-up, pins are in a high impedance state. To avoid unexpected results for DUTs that are sensitive to the initial state of a pattern burst, programmatically specify the initial pin state by using the niDigital Write Static VI, the DigitalPinSet.WriteStatic .NET method, or the niDigital_WriteStatic C function. You can interactively set the initial state for a pin by selecting the corresponding driver symbol in the Drive section of the Pin View pane of the Digital Pattern Editor. After each pattern burst, the ending state of the last executed vector persists until the start of the next pattern burst or until you programmatically or interactively change the pin state or pin function.

Note When using the edge multiplier feature, not all pins will require all the available cells if a vector contains a mixture of edge multiplier values. The unused cells will display as blank cells. Adding pin state data to these blank cells will change the edge multiplier value for that pin on that vector.

- **Comment** — Specifies multiple lines of text as comments. This field does not accept the <Enter> or <Tab> keys. You can use Ctrl+<Enter> to start a new line in this field. The editor displays multi-line comments in a tooltip when you hover over a comment cell that contains multiple lines.

Keyboard Shortcuts within the Pattern Grid View

Action	Shortcut
Start a new line inside a cell in the comments column	<Ctrl+Enter>
Start in-place editing	F2
Find next	F3
Find next occurrence of current selection	<Ctrl+F3>
Find previous	<Shift+F3>
Find previous occurrence of current selection	<Ctrl+Shift+F3>
Expand selected scan vector	<Ctrl+Add Sign>  **Note** Use the Add Sign (+) on the numeric keyboard.
Collapse selected scan vector	<Ctrl+Minus Sign>  **Note** Use the Minus Sign (-) on the numeric keyboard.
Select all	<Ctrl+A>
Copy	<Ctrl+C>
Cut	<Ctrl+X>
Paste	<Ctrl+V>
Undo	<Ctrl+Z>
Redo	<Ctrl+Y>

Related reference:

- Editing Documents
- Text Pattern File Syntax
- Pattern Declaration
- Timing
- Opcodes

- Keep Alive Patterns
- Pattern Waveform View

Pattern Waveform View

Click the **Waveform** tab on the pattern document toolbar to launch the pattern waveform view. Use this view to see a read-only, vector-by-vector graphical representation of the digital waveforms that the pattern file and time sets describe. The view includes the associated time sets, per vector time set names, vector numbers, and pin state data that indicates drives and compares for each vector. This view is not a cycle-based simulation and does not show looped or repeated vectors multiple times, nor does this view represent any pattern branching.An asterisk (*) next to a filename in the Project Explorer window and in the document tab indicates that the file has been modified but not saved.

Reading the Pattern Waveform View

Use this view to verify the effects of timing and pin state data that determine the waveform the pattern specifies. Use the pattern grid view for a text-based representation of the pattern. You can select and right-click vectors in the pattern grid view and select **Show Vectors in Waveform View** from the context menu to launch the pattern waveform view. You can also click the **Waveform** tab on the pattern document toolbar to launch the pattern waveform view.

The graph can contain the following components:

- **Driving** — Indicates a driving pin state (0, 1, D).
- **Not Driving** — Indicates a non-drive pin state (L, H, X, V, M, E). An up or down arrow placed at the strobe time the time set specifies indicates the strobe edge of compare states.
- **Unknown State** — Indicates that sufficient drive state information does not exist for a particular section of the digital signal, such as before the Drive On edge at the beginning of a pattern, after a discontinuity, in situations where a dash (–) pin state cannot be resolved, or when information outside the pattern determines the drive state, such as a source waveform.
- **Missing/Invalid Timing or Pin States** — Indicates that sufficient timing information does not exist for a particular vector, such as when using invalid time sets, in situations where a dash (–) time set cannot be resolved; when an active pin and

channel map file or timing file do not exist in the project; or when a pin state configuration is invalid because a mix of drive and non-drive pin states exist across an edge multiplier.
- **Discontinuity** — Indicates a potential discontinuity in the pattern. The `call`, `jump`, `exit_loop`, `halt`, `keep_alive`, and `return` opcodes can potentially cause a discontinuity in the pattern execution such that the editor executes a vector that is different than the immediate next vector in the pattern. The editor does not show pending edges from the vectors before the discontinuity if the edges occur after the end of the vector before the discontinuity. This view is not a cycle-based simulation and does not show looped or repeated vectors multiple times, nor does this view represent any pattern branching.

The graph displays edges that extend past the end of the vector unless a discontinuity exists.

Configuring the Pattern Waveform View

The pattern waveform view automatically updates to reflect changes in the pattern and timing documents. The pattern waveform view includes the following components:

- **Timing** — Specifies the timing to use to describe the digital waveform.
  - **Active Timing Sheet** — Specifies to use the timing values in the current active timing document. This is the default value.
  - **Applied Timing** — Specifies to use the timing values currently applied to the digital pattern instrument.
- **Visible Vector Items** — Specifies the number of vector items to display at once. The pattern waveform view displays a maximum of 5,000 vector items at a time.

Keyboard Shortcuts within the Pattern Waveform View

Use the scroll bars, mouse wheel, or drag to control panning. Press <Ctrl> with the mouse wheel to control zooming.

Action	Shortcut
Expand all scan vectors	<Ctrl+Shift+Add Sign>

Action	Shortcut
	Note Use the Add Sign (+) on the numeric keyboard.
Expand selected scan vector	<Ctrl+Add Sign> **Note** Use the Add Sign (+) on the numeric keyboard.
Collapse all scan vectors	<Ctrl+Shift+Minus Sign> **Note** Use the Minus Sign (-) on the numeric keyboard.
Collapse selected scan vector	<Ctrl+Minus Sign> **Note** Use the Minus Sign (-) on the numeric keyboard.

Related reference:

- Pattern Grid View

Loading, Unloading, and Modifying a Pattern

Use the pattern document toolbar or context menu to interact with patterns.

Loading Patterns

Digital Pattern Editor prompts you to load a pattern on the digital pattern instrument before bursting. The editor also prompts you to load all referenced waveforms and required patterns if the waveforms and patterns have not already been loaded, such as patterns that contain the target locations for `call`, `jump`, `jump_if`, `exit_loop`, or `exit_loop_if` opcodes.

To manually load a pattern on the instrument, use the **Load Pattern** button in the pattern document toolbar.

> **Note** Loading patterns using this button does not load referenced patterns or waveforms. Use System View to view all patterns currently loaded on the instrument.

Unloading Patterns

To remove patterns and waveforms from an instrument, use the following buttons in the pattern document toolbar:

- **Unload All Patterns and Waveform Configurations**

  > **Note** This button does not unload keep alive patterns.

- **Unload All Patterns, Keep Alive Patterns, and Waveform Configurations**

> **Note** Unloading patterns also unloads source (`.tdms`) and capture (`.digicapture`) waveform configuration files.

Modifying Patterns

If you modify and save a pattern, you must unload and reload the pattern for the changes to take effect on the instrument. You must save the pattern before you can load or burst it on the digital pattern instrument. The Digital Pattern Editor prompts you to save modified files before loading and bursting a pattern.

Related reference:

- System View

Bursting Patterns and Viewing Results

The Digital Pattern Editor bursts the pattern on and collects the results for all enabled sites. Use the **Site** drop-down menu to select the site for which you want to view History RAM results in overlay mode. Click the Enabled Sites button to launch the

Sites dialog box, which you use to enable and disable sites defined in the active pin and channel map, typically for debugging purposes.Refer to the specifications document for the digital pattern instrument for more information about the number of samples supported in History RAM. You can also use the **Site** drop-down menu on the toolbar to select the site to determine when to stop bursting when you select the **Burst Until Fail** or **Burst Until Pass** burst option.

Select the **Run** » **Connect Pins on Burst** menu item to enable or disable automatically connecting I/O switches on burst operations.

Use the **Burst** button pull-down menu to determine how to execute, or run, the loaded pattern. Bursting a pattern from the editor always starts at the first vector of the pattern. Click the **Abort** button to halt pattern execution.

Viewing Results

The **Overlay Last History RAM Results** button on the pattern document toolbar is enabled by default. The editor displays the subset of burst results that corresponds to vectors in the current pattern based on the settings you specify in the Instrument Settings pane. Rebursting a pattern overwrites the History RAM result data in the pattern document in History RAM overlay mode.

The **Passed**/**Failed**/**Disabled** indicator displays whether the last pattern burst includes any failures, regardless of what you log to History RAM, and whether the site was disabled for the last burst of the pattern. Disabled sites do not return results. The **Total Failures** indicator displays the number of failures in the pattern. The **Burst Complete** indicator displays the timestamp of when the last pattern burst completed. Pin state data cells with a red background and red text indicate a failure.

Column heading tooltips display the total number of pin failures for the pattern burst for each pin and the number of failures for the pin recorded in History RAM results in parentheses. Cell tooltips display expected and actual values for failures. Pin state data cells with a white or grey background indicate that no data was available for the cycle. Use the navigation buttons on the toolbar to move among the first, previous, next, and last failures in History RAM overlay mode. Expand a pin group to navigate to the individual pins in the group.

Vectors with a blue background display available History RAM results. Vectors with a

white background do not include History RAM results. The pattern document displays a dash (–) for the cycle count in the **Vector/Cycle** column for vectors without History RAM results. For vectors that execute on more than only one cycle, the pattern document in History RAM overlay mode displays the superset of failures for all cycles with History RAM results.

Click the **Show History RAM View** button on the toolbar or select **View » History RAM** to display in a separate document the burst results for the flow of pattern execution across all executed patterns. The History RAM View shows the results for all cycles of a vector logged to History RAM, including those resulting from looping and `repeat` opcodes. The digital pattern editor automatically displays the History RAM View when the **Overlay Last History RAM Results** button is disabled and the digital pattern instrument returns History RAM results for any site of the pattern burst.

Click the **Show Digital Scope** button on the pattern document toolbar or right-click a cell in the pattern document and select **Use Digital Scope on Selected Pins and Vectors** from the context menu to launch the digital scope, which displays a 2D plot of the actual waveform using the pattern, timing set, and levels.

Refer to the ***Related reference*** section at the end of this topic for a link to more information about History RAM and the digital scope.

Bursting Patterns with Source and Capture Waveforms

The Digital Pattern Editor prompts you to load waveforms on the instrument before bursting a pattern if the pattern directly uses a waveform, including waveforms used in patterns that the pattern jumps to. You can also click the **Load Waveform** button on the toolbar or right-click the waveform file in the Project Explorer window and use the context menu to manually load the current waveform on the instrument. You must specify a name and at least one pin for the waveform. You cannot load a waveform with the same name as another already loaded waveform. Click the **Unload All Patterns and Waveform Configurations** button to remove all source and capture waveforms and patterns from the instrument. If you modify and save the waveform, you must unload and reload the waveform for the changes to take effect on the instrument. The Digital Pattern Editor prompts you to reload modified files before bursting a pattern.

Select **View » Capture Results** or click the **Show Capture Results View** button on the

capture waveform configuration document toolbar or on the pattern document toolbar to view the capture waveform data from the instrument for the last pattern burst. The capture waveform data updates with each burst. Use the `capture_start`, `capture`, and `capture_stop` opcodes to store values when you burst a pattern. Use the `source_start` and `source` opcodes to send waveform data when you burst the pattern.

Refer to the ***Related reference*** section at the end of this topic for a link to more information about source and capture functionality.

Related reference:

- Loading, Unloading, and Modifying a Pattern
- History RAM View
- Instrument Settings Pane
- Digital Scope
- Source Waveform Configurations
- Capture Waveform Configurations
- Digital Source and Capture Opcodes
- Capture Results View
- Learn Failures from History RAM

History RAM View

Use this view to see the History RAM results for the flow of pattern execution for the last pattern burst across all executed patterns for the number of samples you specify in the Instrument Settings pane, including vectors with `repeat` opcodes. The view also includes the corresponding time sets, labels, opcodes, pattern names, vectors/cycles, pin state data, and comments.Rebursting a pattern overwrites the data in the History RAM View.

The digital pattern editor automatically displays the History RAM View when the **Overlay Last History RAM Results** button is disabled and the digital pattern instrument returns History RAM results for any site of the pattern burst. Click the **Show History RAM View** button on the toolbar or select **View » History RAM** to launch the History RAM View.

The Digital Pattern Editor collects results for all enabled sites. Refer to the

specifications document for the digital pattern instrument for more information about the number of samples supported in History RAM. Use the **Site** drop-down menu on the toolbar to select the site for which you want to view results. Click the Enabled Sites button to launch the Sites dialog box, which you use to enable and disable sites defined in the active pin and channel map, typically for debugging purposes.

Viewing Results

The **Passed**/**Failed**/**Disabled** indicator displays whether the last pattern burst includes any failures, regardless of what you log to History RAM, and whether the site was disabled for the last burst of the pattern. Disabled sites do not return results. The **Total Failures** indicator displays the number of failures in the pattern. The **Burst Complete** indicator displays the timestamp of when the last pattern burst completed. Pin state data cells with a red background and red text indicate a failure.

Column heading tooltips display the total number of pin failures for the pattern burst for each pin and the number of failures for the pin recorded in History RAM results in parentheses. Cell tooltips display expected and actual values for failures. Pin state data cells with an asterisk indicate that no data was available for the cycle. This can happen when pins are mapped to multiple instruments with History RAM configured only to capture failures or to trigger on failure. Each instrument independently captures the History RAM results and not all instruments capture the same samples when capturing failures because each instrument only considers the failures on that particular instrument. The same rationale applies for triggering on failure. Use the navigation buttons on the toolbar to move among the first, previous, next, and last failures in History RAM overlay mode. Expand a pin group to navigate to the individual pins in the group.

Click the **Find in Pattern** button on the History RAM View toolbar or press <Ctrl+F> to launch a dialog box in which you can specify the text you want to search for. The History RAM View highlights cells that include the search text. Use the **Find Previous** and **Find Next** buttons in the dialog box or on the History RAM View toolbar to navigate among the results. If you close the **Find in Pattern** dialog box, you can continue to browse through results by selecting **Shift** » **F3** to see previous results or **F3** to see the next results.

Right-click a pin state cell in the History RAM View and select **Use Digital Scope on Selected Pins and Cycles** from the context menu to launch the digital scope, which

displays a 2D plot of the actual waveform using the pattern, timing set, and levels.

Saving Results

Click the **Save History RAM Results** button on the History RAM View toolbar or press <Ctrl+S> to save all the History RAM results in the view in a `.csv` file. The file displays the results for each cycle in a row and a separate column for the actual and expected pin states for each pin, including pins from pin groups.

Click the **Save as Pin Failures List** button on the History RAM View toolbar to save a list of only the pin failures in a `.csv` file. The file displays the results for each failure in a separate row with columns that show the cycle and pin at which the failure occurred and the actual and expected pin state results.

You can also select and copy values from the History RAM View to paste into another application.

Keyboard Shortcuts within History RAM View

Action	Shortcut
Find in pattern	<Ctrl+F>
Find next	F3
Find previous	<Shift+F3>
Save all History RAM results in the view in a `.csv` file	<Ctrl+S>

Related reference:

- Instrument Settings Pane
- Digital Scope
- Flow Control Opcodes
- Learn Failures from History RAM

Instrument Settings Pane

Select **Instrument Settings** » **History RAM and Signal Settings** to launch the

Instrument Settings pane.

History RAM Setup

Configure the following options for bursting a pattern and logging History RAM results to display. Refer to the specifications document for the digital pattern instrument for more information about the number of samples supported in History RAM.

- **Trigger on** — Specifies the conditions on which to begin logging results to History RAM.
  - **Failure** — Begins logging History RAM results on the first cycle that fails. This is the default value.

    > **Note** The editor logs failures on a per-site and per-instrument basis, returning only partial results for a site that spans multiple digital pattern instruments where not all instruments for a site had History RAM results logged for the same cycles. When an instrument does not include all pins of a site and no History RAM results exist for those pins and cycles on another digital pattern instrument in the active pin and channel map, those pin state data cells are indicated with an asterisk in the History RAM View.

  - **Cycle** — Begins logging History RAM results starting with the cycle you specify.
    - **Cycle** — Specifies the cycle number on which to trigger.
  - **Global Label + Offsets** — Specifies to begin logging History RAM results at a particular global label with a cycle offset or a vector offset from the label that you specify.
    - **Label or Pattern** — Specifies the global label at which to start logging History RAM results. A global label is a label that a pattern exports. All pattern names are also global labels for the first vector of the pattern.
    - **Vector Offset** — Specifies the number of vectors to offset from the label or pattern before logging History RAM results.
    - **Cycle Offset** — Specifies the number of cycles to offset from the label or pattern before logging History RAM results.
- **Pretrigger Samples** — Specifies the number of samples to log before the **Trigger on** event occurs. The default value is `0`.
- **Stop** — Specifies the condition on which to stop logging results to History RAM.
  - **When Full** — Stops logging History RAM results when the History RAM is full

and discards any additional results after the History RAM is full. This is the default value.
    - **After X Samples** — Stops logging History RAM results when the number of samples you specify have been logged.
        - **X** — Specifies the number of samples to log.
- **Cycles to Capture** — Specifies the cycles to log when the **Trigger on** event occurs.
    - **Failures Only** — Logs only cycles with a failure. This is the default value.
    - **All** — Logs all cycles.

Export Pattern Opcode Event Signals

You can route up to four events to any of the eight PXI chassis backplane trigger lines per digital pattern instrument based on the signal opcodes you use in a pattern.

Note PXI trigger lines are system-wide resources that other instruments in the test system can use or reserve. When you export pattern opcode events to these output terminals, ensure that another instrument is not using or has not reserved the PXI trigger line.

Assign Pattern Opcode Conditional Jump Triggers

Configure each digital pattern instrument or group of digital pattern instruments to conditionally jump on a specified trigger. The conditional jump triggers can be software signals or PXI trigger lines.

The **Instrument or Group** drop-down menu displays one of the following options:

- The name of a single, unsynchronized instrument.
- The group name for an instrument or group of instruments. Specify a group name for one or more instruments in the Pin Map Editor.
- The chassis name for a set of instruments synchronized through the DPE's **Synchronized** > **Yes** menu option and not specified as a group in the Pin Map Editor.

Frequency Counter

Configure the frequency counter and enable hysteresis for digital pattern instruments.

The Measurement Mode drop-down displays the following options:

Option	Description	Hysteresis supported?
Banked	Frequency measurements are made serially for groups of channels associated with a single frequency counter for each group.	No
Parallel	Frequency measurements are made by multiple frequency counters in parallel.	Yes

Note Frequency counter hysteresis filters out noise by requiring the corresponding rising and falling edges of a signal to pass through both V_{ol} and V_{oh} in the correct order.

Learn Failures from History RAM

Use Learn Failures from History RAM to automatically edit the pin states of a pattern to change failing pin states to ones that will pass.

Configuring Learn Failures from History RAM

To execute this command, use the toolbar button on the pattern document while viewing History RAM failures in the Grid view of the document with the History RAM Overlay feature enabled. Ensure that there is at least one failure to learn. Before beginning to edit a pattern, the Digital Pattern Editor displays a dialog box with the options described below.

Once you confirm your options, the DPE will edit the pin states for the failures currently in History RAM. If your pattern has more failures than will fit in History RAM, burst the pattern again after confirming your options and execute the command again to learn additional failures.

The **Learn Failures from History RAM** dialog box includes the following options:

- **Which Pin States** — Specifies the pin states from which to learn failures.
  - **All Failures in History RAM** — Specifies to learn failures from all failed pin states in History RAM.

  - **Selected Failures** — Specifies to learn failures from selected failed pin states in History RAM.

- **Replace With** — Specifies what to replace the specified failures with.
  - **Actual from History RAM** — Replaces the specified failures with the pin states that match the actual measured results from History RAM.
  - **X** — Replaces the specified failures with the X pin state.

- **On Conflict** — Specifies how to replace specified failures when there is a conflict. The DPE considers a failure to be a conflict in the following circumstances:
  - The existing pin state is a – or E.
  - There is more than one result for the failure in History RAM and at least one of those results conflicts with the other results
  - At least one actual measured result for the failure in History RAM is both high and low. This can happen when the V_{OH} and V_{OL} levels are inverted, or with a compare strobe located at a fast rising or falling edge.
  - The actual measured result in History RAM is an M and the failure is on a scan output pin in a scan vector. Scan output pins only support the L, H, and X pin states.
  - The failure is on a parallel pin in a scan vector.
  - **Stop** — Specifies to stop learning failures at the first conflict. The DPE selects the failure with the conflict and scrolls it into view.
  - **Replace With X** — Specifies to replace the conflicting failures with the X pin state.
  - **Skip Conflicting** — Specifies to skip over any conflicting failures.

- **Add Vector Comment** — Specifies a comment to append to each vector in which the DPE modifies a pin state for the Learn Failures from History RAM operation. This field does not accept the <Enter> or <Tab> keys.

  For scan vectors, comments can only be added to the template rows of the vector, so the DPE also appends the ranges of scan indices in which a pin state was learned. For example, if a pin state was learned on scan cycles 0, 1, 2, and 4, the default comment appended would be: Learned (scan indices: 0-2, 4). If the vector already has an existing comment, the DPE will append the comment you specify using a comma as a delimiter to separate it from the existing comment. If you leave this setting blank, the DPE does not append any comments for this operation.

Debugging Techniques

To interactively debug patterns on digital pattern instruments, you typically pause the execution of the test program and open a project in the Digital Pattern Editor that has the same active pin and channel map as the test program uses. If you previously created instrument sessions with the digital pattern editor, disconnect those sessions before executing the test program. The digital pattern editor automatically connects to the available instruments defined in the active pin and channel map or returns a detailed error message explaining why it cannot connect to the sessions. You can also select the **Instruments** » **Connect** menu item in the digital pattern editor to connect to the existing instrument sessions the test program application created.

Once you have established a connection to the instrument sessions, you can complete the following troubleshooting steps. Refer to the ***Related reference*** section at the end of this topic for a link to more information about each of these components.

- Use the System View to view or modify pin settings and measurements.
- Use the Pin View pane to view or modify pin settings.
- Unload, load, and burst patterns after optionally modifying specifications, levels, timing, or pattern components.
- View History RAM results for bursts in overlay mode in pattern documents or in the History RAM View. Use the Instrument Settings pane to customize settings for logging results to History RAM.
- Mask pins to ignore failures in a pattern burst for those pins.
- Disable the sites you want to ignore while debugging.
- Execute Shmoo or digital scope operations to review results.

Note The Pin View pane and System View display a yellow warning icon to indicate measurements that are not up to date when the digital pattern editor is connected to existing instrument sessions and an instrument session owner, typically a test program application, is actively using the instrument session. Click the icon to take control of the instrument and acquire new measurement values in the background. The warning icon returns when the instrument session owner actively uses the instrument sessions again.

Related reference:

System View

Use System View to edit the settings and measurements for all pins and relays in the active pin and channel map connected to a digital pattern instrument, NI-DCPower instrument, or relay driver module.

This view contains a list of patterns and source waveforms loaded on the digital instruments. The following table displays the information contained in each list type.

Note For more information on vector memory and source waveforms, refer to the relevant section of the PXIe-6570 User Manual and Specifications.

List Type	Details
Patterns	Contains the following information: • Number of fast vectors • Number of cache vectors • Number of large vectors • Total vector usage
Source Waveforms	Contains the following information: • Number of bytes used by each loaded source waveform • Total memory usage

You can use this view to edit pattern sequencer flags and registers and interactively

debug and make changes to the state of your digital pattern instruments, NI-DCPower instruments, or relays.

> **Note** As of 2024 Q2, the System View displays NI-DMM, NI-SCOPE, NI-FGEN, NI-RFSA, NI-RFSG, and NI-DAQmx pins. These pins do not support all attributes.

You must install drivers to access additional behaviors in the Digital Pattern Editor:

- Install the NI-DCPower driver for NI-DCPower-related functionality.
- Install the NI-DAQmx driver for relay driver functionality.

Select **Instruments** » **New System View** or **View** » **New System View** to launch this view. You can configure a custom layout by resizing columns, by dragging the column heading to move the column to a new location, and by clicking the column heading to sort the rows in the table. The layout customizations do not persist when you close the view. You can copy data from the grid and paste it into spreadsheet applications and text editors that support tab-delimited clipboard text.

The Pin View pane and System View display a yellow warning icon to indicate measurements that are not up to date when the Digital Pattern Editor is connected to existing instrument sessions and an instrument session owner, typically a test program application, is actively using the instrument session. Click the icon to take control of the instrument and acquire new measurement values in the background. The warning icon returns when the instrument session owner actively uses the instrument sessions again.

Rows for disabled sites display in grey. When you remove the checkmark from the **Pattern Burst** checkbox of a site in the Sites dialog box, rather than disabling it, you can still view the site in the System View and modify pins in the site in the Pin View or System View pane, but bursting the pattern has no effect on the site.

Edit individual cells to change the state of your digital pattern instruments, NI-DCPower instruments, or relays. To edit multiple cells at once, copy a cell that includes the desired value, select the cells you want to edit, and paste the value into that selection.

Use the **Disconnect All Digital Pins** button in the Function column to change the state

of all digital pins to disconnect from another supported state.

You can toggle the state of a relay through the corresponding checkbox in the State column of the Relays section of the document. Adding a checkmark to the state checkbox will set the relay to Closed, while removing the checkmark will set the relay to Open. Change the state of multiple relays at once by selecting multiple checkbox cells in the grid and changing the state with the mouse or by pressing the spacebar key.

You can toggle the state of a pattern sequencer flag through the corresponding checkbox in the Pattern Sequencer Flags section of the document. You can change the value of a pattern sequencer register by editing the corresponding cell in the Pattern Sequencer Registers section of the document.

Use System View commands to navigate to other locations outside of Digital Pattern Editor.

- Right-click a source measure unit (SMU), data acquisition, digital multimeter, radio frequency (RF), oscilloscope, and waveform generator pin and select **Go to InstrumentStudio** to launch InstrumentStudio for that item.
- Right-click a pin map file in the Project Files View and select **Go to InstrumentStudio** to launch InstrumentStudio for that pin map.
- Right-click a Digital or DCPower pin and select **Go to Pin View** to navigate to the Pin View pane for that pin.

Related reference:

- Table View Filters

Related information:

- PXIe-6570 Specifications
- Vector Memory
- Sourcing and Capturing Waveforms

Pin View Pane

Use the Pin View pane during debugging to interactively view and modify the current

state of the pin driver, active load, comparators, PPMU, and NI-DCPower instrument settings for a single pin at a time in the active instrument session.

Note You must install the NI-DCPower driver to access NI-DCPower-related functionality in the Digital Pattern Editor.

The pane displays by default when you create or open a project. Click the collapse » or expand « arrows or hover over the Pin View Pane icon when the pane is collapsed to toggle the display of the pane. You can also select **Instruments » Pin View** or **View » Pin View** to display the Pin View pane.

Changes you make in the Pin View pane apply only to the currently connected digital pattern instruments. You must manually make changes in the levels and timing documents if you want the changes to take effect the next time you burst a pattern. The pane reflects the changes you make in levels and time sets when you reapply levels and timing values on the instrument.

Note The Pin View pane and System View display a yellow warning icon to indicate measurements that are not up to date when the digital pattern editor is connected to existing instrument sessions and an instrument session owner, typically a test program application, is actively using the instrument session. Click the icon to take control of the instrument and acquire new measurement values in the background. The warning icon returns when the instrument session owner actively uses the instrument sessions again.

Configuring the Pin View Pane

Use the **Site** drop-down menu to select a site for this operation. Select **System** to view the system pins in the **Pin** control. When you remove the checkmark from the **Pattern Burst** checkbox of a site in the Sites dialog box, rather than disabling it, you can still view the site in the System View and modify pins in the site in the Pin View or System View pane, but bursting the pattern has no effect on the site.

Use the **Pin** control to select the pin in the pin and channel map for which you want to view or modify settings. Use the tooltip to display the instrument and channel that

correspond to the pin. Select **System** in the **Site** drop-down menu to view the system pins in the **Pin** control menu. Click the switch icon  , when available, to connect or disconnect the pin.

Use the expander buttons to expand or collapse each section in the pane. The pane dims the sections that are not active. Refer to the ***Related reference*** section at the end of this topic for a link to more information about each section.

A green icon next to the Drive section indicates that the digital pattern instrument is in digital mode for the current pin when connected. A green icon next to the PPMU or DCPower section indicates that the PPMU or the NI-DCPower instrument is forcing a voltage or a current.

You can drag a document by its tab away from the main workspace window and drop the document without highlighting a docking guide location to detach the document from the main workspace and create a new instance of the application window. All instances of the application window refer to the same project file and share the same state. Using this technique, you can display the Pin View pane in each instance of the application to debug multiple pins at the same time. Selecting **View » Reset Workspace** in a window in this case closes all the other windows.

Related reference:

- Pin View Pane Drive Section
- Pin View Pane Active Load Section
- Pin View Pane Compare Section
- Pin View Pane PPMU Section
- Pin View Pane NI-DCPower Section
- Getting Started with Digital Pattern Editor
- Pin View Pane Frequency Section
- Pin View Pane DCPower Section

Pin View Pane Drive Section

Use this section of the Pin View pane to specify pin driver options. A green icon next to the Drive section indicates that the digital pattern instrument is in digital mode for the current pin when connected.

- **Termination Mode** — Specifies the behavior of the pin when the pin function is set to digital and the pin state for the current cycle is not a drive state.
  - **High Z** — Specifies that, for non-drive pin states (L, H, X, V, M, E), the pin driver is put in a high-impedance state and the active load is disabled. This is the default setting.
  - **Vterm** — Specifies that, for non-drive pin states (L, H, X, V, M, E), the pin driver terminates the pin to the configured V_{TERM} voltage through a 50 Ω impedance. V_{TERM} is adjustable to allow for the pin to terminate at a set level. This is useful for devices that might operate incorrectly if an instrument pin is unterminated and is allowed to float to any voltage level within the instrument voltage range. To address this issue, enable V_{TERM} by configuring the V_{TERM} pin level to the desired voltage and selecting the V_{TERM} termination mode. Setting V_{TERM} to 0 V and selecting the V_{TERM} termination mode has the effect of connecting a 50 Ω termination to ground, which provides an effective 50 Ω impedance for the pin. This can be useful for improving signal integrity of certain DUTs by reducing reflections while the DUT drives the pin.
  - **Active Load** — Specifies that, for non-drive pin states (L, H, X, V, M, E), the active load is connected and the instrument sources or sinks a defined amount of current to load the DUT. The amount of current sourced by the instrument and therefore sunk by the DUT is specified by I_{OL}. The amount of current sunk by the instrument and therefore sourced by the DUT is specified by I_{OH}. The voltage at which the instrument changes between sourcing and sinking is specified by V_{COM}. Configure V_{COM}, I_{OL}, and I_{OH} in the Active Load section of the pane, which expands when you select this option.
- V_{TERM} — Specifies the termination voltage the instrument applies during non-drive cycles when the termination mode is set to V_{TERM}. The instrument applies the termination voltage through a 50 Ω parallel termination resistance.
- V_{IH} — Specifies the voltage that the instrument applies to the input of the DUT when the instrument drives a logic high (1).
- V_{IL} — Specifies the voltage that the instrument applies to the input of the DUT when the instrument drives a logic low (0).
- **Driver Symbol** — Click the box to the left of the driver symbol, as shown in the following figure, to set the control to **X** to terminate driving a value or to set the control to **1** or **0** to specify a value to temporarily drive. Bursting a pattern overrides this temporary setting.

- **Enable Button** — Enables the driver and disables any other active enable button. Disabling the driver puts the pin into a high-impedance state until you enable the driver, the PPMU Force Voltage mode, or the PPMU Force Current mode.
- **Clock**—Enables generation of digital clock signal on the pin. Specify the clock frequency in the adjacent numeric edit box. **Clock** only generates the clock signal when the pin is in digital mode.

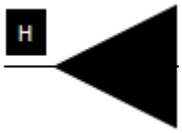
Pin View Pane Active Load Section

Use this section of the Pin View pane to specify active load options.

- **Active Load Enable** — Enables or disables the active load. Disabling the active load sets the termination mode to **High Z**.
- V_{COM} — Specifies the commutating voltage level at which the active load circuit switches between sourcing current and sinking current.
- I_{OL} —Specifies the current that the DUT sinks from the active load while outputting a voltage below V_{COM}.
- I_{OH} —Specifies the current that the DUT sources to the active load while outputting a voltage above V_{COM}.

Pin View Pane Compare Section

Use this section of the Pin View pane to specify comparator options.

- **Comparator Symbol** — The box to the left of the comparator symbol, as shown in the following figure, indicates the comparator results. 

  - **H** — Indicates a value above V_{OH}.
  - **L** — Indicates a value below V_{OL}.
  - **M** — Indicates a value that is below V_{OH} and above V_{OL}.
  - **V** — Indicates a value that is above V_{OH} and below V_{OL}, which can occur when you set V_{OL} higher than V_{OH}.

  - - — Indicates that the pin is disconnected or cannot be read.
- V_{OH} — Specifies the DUT output voltage above which the comparator on the instrument pin interprets a logic high (H).
- V_{OL} — Specifies the DUT output voltage below which the comparator on the instrument pin interprets a logic low (L).

Pin View Pane PPMU Section

Use this section of the Pin View pane to specify pin parametric measurement unit (PPMU) options. A green icon ● next to the PPMU section indicates that the PPMU is forcing a voltage or a current.

> **Note** The collapsed PPMU section displays the PPMU voltage indicator in the section heading. When you expand the section, the voltage indicator displays below the circuit diagram. The PPMU is always measuring voltage when the pane is visible.

- **Voltage and Current Indicator** — Displays the actual voltage and current. The PPMU always measures the voltage on the pin but measures the current only when forcing a voltage or a current.
- **I Limit Range/I Level Range/I Range** — Specifies the current range to use. Use the pull-down menu to select a valid range. Displays as **I Limit Range** when you force a voltage, **I Level Range** when you force a current, and **I Range** when idle.
- V_F Enable Button ⏻ — Enables PPMU Force Voltage and disables any other active enable button. Disabling PPMU Force Voltage puts the pin into a high-impedance state until you enable the pin driver, enable PPMU Force Voltage, or enable PPMU Force Current.
- V_F — Specifies the voltage level that the PPMU forces to the DUT pin. Takes effect when the PPMU is in Force Voltage mode.
- I_F Enable Button ⏻ — Enables PPMU Force Current and disables any other active enable button. Disabling PPMU Force Current puts the pin into a high-impedance state until you enable the pin driver, enable PPMU Force Voltage, or enable PPMU Force Current.
- I_F — Specifies the current level that the PPMU forces to the DUT. Takes effect when the PPMU is in Force Current mode.
- V_{CH} — Specifies the nominal voltage at the pin at which the high side voltage

clamp activates when the PPMU forces current to the DUT. Voltage clamp high is enabled only when the PPMU is in Force Current mode.
- V_{CL} — Specifies the nominal voltage at the pin at which the low side voltage clamp activates when the PPMU forces current to the DUT. Voltage clamp low is enabled only when the PPMU is in Force Current mode.

Pin View Pane NI-DCPower Section

Use these sections of the Pin View pane to specify when you select a pin connected to an NI-DCPower instrument to specify voltage and current options. A green icon ● next to the DCPower section indicates that the instrument is forcing a voltage or a current. The pane does not display the switch  icon to connect or disconnect the pin when the digital pattern instrument does not support this type of functionality.

NI-DCPower Section

The NI-DCPower section displays the voltage indicator in the section heading. In this section, the voltage indicator displays below the circuit diagram. When the pane is visible, the NI-DCPower instrument is measuring voltage and current.

> **Note** You must install the NI-DCPower driver to access related functionality in the Digital Pattern Editor. For more information on the driver, refer to the NI-DCPower User Manual.

- **Voltage and Current Indicator** — Displays the actual voltage and current. The instrument always measures the voltage and current on the pin.
- **I Range** — Specifies the current range to use. Use the pull-down menu to select a valid range. Displays **I Level Range** when you force a current, **I Limit Range** when you force a voltage, and **I Range** when idle.
- **V Range** — Specifies the voltage range to use. Use the pull-down menu to select a valid range. Displays **V Limit Range** when you force a current, **V Level Range** when you force a voltage, and **V Range** when idle.
- V_F Enable Button ⏻ — Enables Force Voltage and disables any other active enable button. Disabling Force Voltage puts the pin into a high-impedance state until you enable Force Voltage or enable Force Current.
- I_C — Specifies the maximum limit for the current at the pin when the instrument

forces voltage to the DUT. This value is enabled when the instrument is in Force Voltage mode.

- I_F Enable Button — Enables Force Current and disables any other active enable button. Disabling Force Current puts the pin into a high-impedance state until you enable Force Voltage or enable Force Current.
- V_C — Specifies the maximum limit for the voltage at the pin when the forces current to the DUT. Enabled only when the instrument is in Force Current mode.
- Sense—Specifies the method used when collecting output and voltage measurements.
  - Local—Use a single set of connections for output and voltage measurement.
  - Remote—Use two sets of connections for output and voltage measurement.

> **Note** Remote sense measurements are appropriate for high-current applications. Remote sense enables more accurate voltage output and measurements when the output lead voltage drop is significant.

Driver Attributes Section

The Driver Attributes section displays a list of attribute names and values. Use this section to check or modify the configuration of a miscellaneous setting on an NI-DCPower SMU without launching InstrumentStudio.

Related information:

- NI-DCPower User Manual

Pin View Pane Frequency Section

Use this section of the Pin View pane to measure the frequency of a signal on a digital pin.

> **Note** The collapsed Frequency section displays the frequency indicator and measure frequency button in the section heading. When you expand the section, the frequency and period indicators and the interval control display.

- **Frequency and Period Indicator** — Displays the measured frequency and period.

The measurement is taken by counting the number of rising edges that cross over V_{OL} that occur over a specified interval time.

> **Note** You can take measurements only when the pin is in digital mode. You cannot take measurements while using the PPMU.

- **Interval** — Specifies the interval time to use. The interval time is the period during which the frequency counter is measuring the frequency. The larger the interval time, the better the resolution. Select short interval times for faster measurement speed.
- **Measure Frequency Button** — Takes a new frequency measurement and updates the frequency and period indicators.

Masking Compares on Pins or Pin Groups

In the pattern document, select cells in the columns for the pins or pin groups for which you want the digital pattern instrument to ignore failures when you burst the pattern, right-click, and select **Mask Compares on Selected Pins** from the context menu. Select cells in the columns for masked pins or pin groups, right-click, and select **Unmask Compares on Selected Pins** from the context menu to remove the setting. These settings persist only until you close the instrument sessions.

When you mask compares on pins or pin groups, the digital pattern editor dims the column background and compare pin states for the pin or pin group in the pattern document, including in History RAM overlay mode, and in the History RAM View. The background is dimmed when the pin or pin group is set to mask compares on the next burst, and the compare pin states for a pin or pin group are dimmed if compares were masked when the pattern was last burst. If not all pins in a pin group are masked, the editor dims only the compare pin states for the masked pins, not the entire pin group, and the header of the pin group column displays the text `partially masked`.

Digital Scope

Use the digital scope to view a graph of the expected waveform underneath the actual waveform using the pattern, timing set, and levels, including compare strobe locations, V_{OH} and V_{OL} levels, and expected drive levels. You can use the digital scope to help debug failures to verify that the actual waveform accurately reflects what you

expect based on what the pattern and timing define.

The Digital Pattern Editor generates digital scope waveforms by repeatedly bursting a pattern while changing levels and the timing of the strobe edge to determine the level of the waveform at each point in time.

Note The cycles or vectors on which you generate digital scope waveform points must be repeatable for every burst of the pattern. Using opcodes that result in non-deterministic pattern execution, such as using conditional jumps based on the failed or matched state of the pattern, can result in errors and potentially inaccurate results. Remove the pins with such compares from the pin list for the digital scope to avoid such inaccuracies. For example, if you configure the digital scope to start at a cycle, all cycles in the range you specify must always be at the same vectors on every burst. If you configure the digital scope to start at a vector, that vector does not have to always execute on the same cycle, but the number of cycles you specify to plot after the start vector cycle for each burst must correspond to the same vectors on every burst.

Launching the Digital Scope

Use the following techniques to launch the digital scope:

- Select **View » Digital Scope**.
- Click the **Show Digital Scope** button on the pattern document toolbar.
- Right-click a cell in the pattern document and select **Use Digital Scope on Selected Pins Data** from the context menu.
- After you burst a pattern, right-click a cell in the History RAM View and select **Use Digital Scope on Selected Pins Data** from the context menu.

Running the Digital Scope

Use the **Run Digital Scope** and **Abort** buttons on the toolbar to control the operation.

Note When you run the Digital Pattern Editor in demo mode and execute a digital scope operation, the generated results reflect the actual pin states

> from the target pattern and the currently active timing values but do not reflect a realistic execution of patterns. Use the pattern waveform view to see a graph of the pattern without requiring the use of digital pattern instruments.

Reading the Digital Scope

The digital scope graph can contain the following components:

- **Sampled Point** — Indicates a particular sample. These points disappear if the number of samples available provides a sufficient resolution to draw a smooth signal.
- **Non-Converging Point** — Indicates a particular sample that might be inaccurate or noisy, such as when a signal has a high amount of noise, the pattern is non-deterministic, or the point occurs on sharp rising or falling edges of a signal.
- **Cycle Boundary** — Indicates the cycle boundaries.
- **VOH and VOL Levels** — Indicates the V_{OH} and V_{OL} levels per pin.
- **Driving** — Indicates a driving pin state (0, 1, D).
- **Not Driving** — Indicates a non-drive pin state (L, H, X, V, M, E). An up or down arrow placed at the strobe time the time set specifies indicates the strobe edge of compare states.
- **Unknown State** — Indicates that sufficient drive state information does not exist for a particular section of the digital signal, such as before the Drive On edge at the beginning of a pattern, after a discontinuity, in situations where a dash (–) pin state cannot be resolved, or when information outside the pattern determines the drive state, such as a source waveform.

Configuring the Digital Scope

Use the **Pattern**, **Pins/Pin Groups**, and **Site** controls in the Digital Scope Configuration section to configure digital scope execution settings. Use the type-ahead drop-down menu to select valid pins and pin groups. Click the Enabled Sites button to launch the Sites dialog box, which you use to enable and disable sites defined in the active pin and channel map, typically for debugging purposes.Disabled sites do not return results. Use the **Start Vector**, **Cycle Offset**, **Number of Cycles**, **Number of Steps**, and **Voltage Resolution** controls in the Sweep Ranges section to configure sweep operation settings. The project stores the settings for the digital scope but not the plot

data itself.

> **Note** Setting the number of steps to a value that results in a resolution smaller than the timing accuracy the digital pattern instrument supports does not yield any additional improvements in the waveform accuracy.

Use the expander buttons to expand or collapse the settings section to create a larger display area for the digital scope.

When you right-click a cell in the pattern document and select **Use Digital Scope on Selected Pins and Vectors** from the context menu or right-click a cell in the History RAM View and select **Use Digital Scope on Selected Pins and Cycles** from the context menu, the digital scope preconfigures the **Start Vector**, **Cycle Offset**, **Number of Cycles**, **Pins/Pin Groups**, and **Pattern** settings for the digital scope operation based on the vector or cycle you selected.

You can also use the **Site** drop-down menu on the toolbar to select the site for which you want to display plot results.

While the digital scope is running, the vertical red points mark the point in time for which the levels on each pin are currently being evaluated. Green circles indicate fully evaluated points and always display while executing.

Configuring a small number of steps might result in incomplete waveform information if the signal changes rapidly.

Instrument settings restore to their original values when the digital scope operation completes or when you abort the operation.

Use the two cursors on the digital scope to display the timing and measured voltage values for all pins. You can drag the two cursors independently from each other to compute the time offset difference between the two locations and display the value in the top left corner of the digital scope. The cursors reset to the beginning of the digital scope plot when you restart the digital scope operation.

Use the **Toggle Expected Waveforms on Top** button to change whether the actual waveform overlays the expected waveform. Use the **Always Show Sampled Points**

button  to toggle the display of all points on the plot. The graph always displays the sampled points while executing. Use the **Show Levels at Cursor** button to toggle the display of the per pin voltage levels at the locations of the cursors.

Use the buttons above the digital scope to control panning and zooming. The pan and zoom locations reset to center when you restart the digital scope operation.

The document stores the specified digital scope settings, but not the graph data itself, in the corresponding `.digiprojcache` file for the project.

Note Once the Digital Pattern Editor calculates and applies TDR values, the Digital Scope's timing is now relative to the DUT. The time displayed in the Digital Scope represents the time the DUT drove or received the signal, not the time a digital pattern instrument drove or received the signal. Use the Digital Scope to validate signals driven from the DUT. The Digital Scope displays the expected waveform signals driven the DUT in blue.

Use the **Halt Vector** control to specify the vector index at which to halt the pattern in the digital scope. This vector should not have any opcodes.

Note The default value for this control is -1 and results in bursting the entire pattern.

Saving the Digital Scope Information

Use the **Save Plot Data to CSV File** button  on the toolbar or <Ctrl+S> to save the actual and expected plot data for the active site to a `.csv` file. Use the **Save Screenshot to PNG** button  on the toolbar to save a screenshot of the state and plot data of the digital scope as an image file. You must expand the settings section to include it in the image.

Keyboard Shortcuts within the Digital Scope

Action	Shortcut
Run Digital Scope	F5
Control panning	<Ctrl+left-click> and drag

Action	Shortcut
Zoom on a region you select	<Shift+left-click> and drag
Save actual and expected plot data for the active site to a `.csv` file	<Ctrl+S>

Related reference:

- Bursting Patterns and Viewing Results
- Pattern Grid View
- History RAM View

Shmoo Plot

Use the Shmoo plot to view a dynamically updated plot of pass and fail values for a sweep of up to two variables you specify using the specifications, timing, levels, pin and channel map, and pattern files in the project. An asterisk (`*`) next to a filename in the Project Explorer window and in the document tab indicates that the file has been modified but not saved.

Select **File** » **New** or click the **Add Item** button on the Project Explorer window toolbar to create a new document. Double-click a `.digishmoo` file in the Project Explorer window or right-click the file and select **Open** from the context menu to launch the Shmoo plot.

Running the Shmoo Plot

Select the **Run** » **Connect Pins on Burst** menu item to enable or disable automatically connecting I/O switches on burst operations. Use the **Run Shmoo** and **Abort** buttons on the toolbar to control the Shmoo operation.

Configuring the Shmoo Plot

Use the **Site** drop-down menu on the toolbar or below the **Settling Time** control to select the site for which you want to display plot results. Click the Enabled Sites button to launch the Sites dialog box, which you use to enable and disable sites defined in the active pin and channel map, typically for debugging purposes. You can change the site for which you want to display plot results before the operation completes. You can

also use the **Site** drop-down menu to specify the site to use for edge detection in edge traversal mode. In edge traversal mode, changing the site once the Shmoo operation starts does not change the site used for the edge traversal. Disabled sites do not return results.

You can sweep up to two levels, voltages, currents, edges, or specifications variables at a time. When you sweep a specification variable, all the levels and edges in the active timing and levels sheets affected by the change to the variable are updated at each sweep interval. You can specify the numeric start, stop, and number of step values to use for each variable. The start, stop, and number of step options determine the number of iterations the Shmoo operation executes. The start and stop values must use the same SI prefix. Using a large number of steps might generate too many points and negatively affect performance.

Note The Shmoo operation does not validate that the variables you select remain within reasonable ranges. For example, if specifications formulas result in $V_{IH} < V_{IL}$ for some points, the Shmoo operation continues to generate the plot, even though the points in this situation are not reasonable configurations.

You can execute the Shmoo operation in sweep, zigzag, progressive resolution, or edge traversal mode. The default Shmoo plot mode is progressive resolution, which provides a high-level view of the preliminary results and then increases the level of detail iteratively until the final results display. Digital pattern instrument settings restore to their original values when the Shmoo operation completes or when you abort the operation.

For specification variables, the Shmoo operation uses the values in the active levels file and the active timing file listed in the Project Explorer window.

The comma-separated list of pins or pin groups and the time set you specify must match valid values in the corresponding pin and channel map and timing file. Use the type-ahead drop-down menu to select valid pin, pin group, and time set values.

Note You must install the NI-DCPower driver to access NI-DCPower-related functionality in the Digital Pattern Editor.

You can select **<None>** for the **Type** option on either axis to create a 1D Shmoo.

You can also define a settling time value to add a delay between when the Digital Pattern Editor applies the values to the instruments and when it bursts the pattern. If the Shmoo plot seems to take more time than you expect, ensure that the settling time value is not set to an unnecessarily high value.

You can apply the settings at a specific point on the Shmoo plot and burst the corresponding pattern by double-clicking the point on the Shmoo plot or by using the cursors to navigate to the point on the Shmoo plot and double-clicking the cursor target. The Digital Pattern Editor applies the settings specified at that point, initiates a burst, and retains those settings. The editor brings the pattern grid view or the History RAM view to the front and briefly highlights the Passed/Failed status icon to indicate that the pattern finished bursting. Alternatively, you can drag and dock the pattern document so you can view the pattern and Shmoo plot in the workspace at the same time.

Use the expander buttons to expand or collapse the settings section to create a larger display area for the Shmoo plot.

Use the buttons above the Shmoo plot to control panning and zooming, and use the cursor on the Shmoo plot to display the value for an individual point.

You can configure and save the settings for multiple Shmoo plots per project. The document stores the Shmoo settings but not the plot data itself.

You can select among various options to display simulated results when you run the Digital Pattern Editor in demo mode.

Saving the Shmoo Plot Information

Use the **Save to PNG** button on the toolbar to save the current state of the Shmoo plot as an image file. You must expand the settings section to include it in the image.

Use the **Save results** button on the toolbar to save Shmoo settings and Shmoo plot results in a text file.

Keyboard Shortcuts within the Shmoo Plot

Action	Shortcut
Run Shmoo	F5
Control panning	<Ctrl+left-click> and drag
Zoom on a region you select	<Shift+left-click> and drag
Zoom out	<Shift+right-click>

Related reference:

- Getting Started with Digital Pattern Instruments
- NI-Digital Pattern Driver Examples

Table View Filters

Table view filters enable you to visualize items in document tables based on specified values. The filter inputs appear at the top of each section within the document you want to filter. Enter a search term in the filter to display rows that contain that term.

Column-based Filters

To filter a specific column, enter the column name followed by a colon and one or more search terms. Separate multiple search terms with a semi-colon. All search terms you specify must be present in a row for it to appear in your filtered view.

> **Note** If you want to use character-sensitive searches, click the **Match case** button next to the filter entry field. If you want to search for whole words only, click the **Match whole word** button.

A space starts a new search string. If the column name includes a space, refer to the ***Quotation Marks*** section to learn how to filter with search strings containing spaces. Refer to the following examples to see how to filter by column name:

```
columnName:CellValueA;CellValueB;CellValueC
```

```
pingroups:pingroupA;pingroupB
```

```
"termination mode":vterm

function:ppmu
```

Quotation Marks

To filter using terms containing spaces, colons, or semicolons, use quotation marks. Refer to the following examples to see how to use quotation marks in your search strings:

```
"termination mode":"High Z"

"abc:def"

"pin groups":pingroupA
```

Note If you need to use a quotation mark as part of a search string, put two adjacent quotation marks inside a search string. The filter string `"abc""def"` will search for abc"def

Operators

To filter your tables based on certain conditions, use the following operators:

- NOT—Use the `NOT` operator before a keyword to invert the truth condition for it. For example, the following string hides all rows with disabled sites: `NOT site:disabled`
- OR—Use the `OR` operator to match any member of a group of search keywords. For example, the following string shows all rows that contain both vterm and digital, or ppmu and voltage: `vterm digital OR ppmu voltage`.

Combining Filters

To narrow your results, use a combination of filter types. Use a search string delimited by spaces to specify multiple search terms. For example, the following search string will show all pins in the mygroup pin group that are not on disabled sites and have termination mode set to vterm: `NOT site:disabled pingroups:mygroup terminationmode:vterm`

Related concepts:

- Register Maps
- Register Maps

Related reference:

- System View
- System View

Exporting Patterns

Select **File** » **Export Pattern** to save the binary version of the pattern file (`.digipat`) as a text pattern file (`.digipatsrc`) or as a compressed text pattern file (`.digipatsrc.gz`) that you can store in a source code control system or use with a third-party application to compare revisions of the pattern file.

Related reference:

- Importing Patterns
- Text Pattern File Syntax

Keep Alive Patterns

Keep alive patterns are specialized, simple looping patterns designed to keep the DUT from unlocking clocks and PLLs while you perform other tasks, such as loading or unloading patterns, changing time sets, changing digital pattern instrument configurations, or moving from one test to another in the test program.

Select **File** » **New** or click the **Add Item** button + on the Project Explorer window toolbar to create a new document. Double-click a `.digipat` file in the Project Explorer window or right-click the file and select **Open** from the context menu to launch the pattern document. The toolbar for a keep alive pattern includes only buttons relevant to keep alive patterns. The keep alive pattern document includes a status indicator and a **Keep Alive Pattern** indicator.

To burst a keep alive pattern, first load the keep alive pattern and then burst a

standard pattern that jumps to a keep alive pattern.

You must also use the `keep_alive` opcode to switch to a keep alive pattern from a standard pattern. You must use the `keep_alive` opcode on the last vector of a keep alive pattern to loop the keep alive pattern until you burst another pattern or until you abort the keep alive pattern. You can load or unload only one keep alive pattern on or from the digital pattern instrument at a time.

History RAM results for the standard pattern that jumped to the keep alive pattern do not change while the keep alive pattern executes.

If the keep alive pattern does not use all the pins referenced in the standard pattern that was executing before you called the keep alive pattern, the pins excluded from the keep alive pattern maintain their previous state.

Related reference:

- Pattern Grid View
- Pattern Declaration

Scan Patterns

Scan patterns are specialized patterns designed to test DUT circuitry using scan chains. Scan patterns contain scan pins and scan vectors in addition to DUT pins and vectors.

Typically scan patterns are generated as text or text pattern files (`.digipatsrc`) that can be imported into a Digital Pattern Editor project. You also can create scan patterns directly in the Digital Pattern Editor by creating a regular pattern and adding scan pins and scan vectors to that pattern.

To add a scan pin to your pattern, add the desired pin, select **Change Pin Type** from the context menu, and change the pin type to **Scan Input** or **Scan Output**. You can change a scan pin to a parallel pin (a regular DUT pin) by selecting **Input/Output (default)** from the context menu. The pattern grid view displays scan pins in the pin's column header.

Convert any vector to a scan vector by using the `scan` opcode. Specify the number of

scan cycles in the scan vector using the single argument of the `scan` opcode. In the pattern grid view, a scan vector consists of a template item followed by the specified number of scan cycle items. Edit the pin state values of parallel pins in the template rows and the pin state values of scan pins in individual scan rows. Parallel pin state values are repeated in the scan rows, but you cannot edit parallel pin state values within scan rows.

You can insert, delete, copy, and paste scan rows like you would any other rows in the pattern grid view. Collapse or expand the scan rows in a scan vector using the button to the right of the scan opcode. You can expand or collapse all scan vectors in the pattern using the toolbar buttons in the pattern grid view and the pattern waveform view.

Treat scan patterns like regular patterns in the Digital Pattern Editor. You can burst scan patterns, view their waveforms, use the digital scope, and perform other expected tasks. During pattern execution, the scan cycles for a scan vector execute. The scan template scan row defines the behavior of the parallel pins on the scan cycles but does not execute. The behavior of the scan pins on the scan cycles is defined by the scan pin states.

Register Maps

Use Digital Pattern Editor to view, create, modify, and save register map files (`.digireg`).

> **Note** As of 2024 Q2, you can use Digital Pattern Editor to create and fully edit register map files.

Register maps display information about DUT registers in a table that contains field names, values, types, and comments.

Related tasks:

- Creating a Register Map

Related reference:

- Table View Filters

Creating a Register Map

You can create, modify, or import a register map in your Digital Pattern Editor project.

1. Determine the method you will use to create your project's register map with the following table.

Method	Tasks
Create a new register map in Digital Pattern Editor.	Open a project in Digital Pattern Editor and select **File** » **New Register Map**.
Create a new register map externally in a text editor.	a. Open a new file in a text editor. b. Edit the register map to fit your requirements. c. Save the file with the `.digireg` file extension d. Open a project in Digital Pattern Editor and select **File** » **Add file**.
Use an example register map.	a. Open a project in Digital Pattern Editor. b. Click the **home button** and select the **Learning** tab. c. Click Register Maps and launch one of the examples. d. In the Project Files tab, double-click the register map file (`.digireg`). e. Edit the register map to fit your requirements.
Import an existing SDC register map.	Open a project in Digital Pattern Editor and select **File** » **Import Files** » **Register Maps** » **Semiconductor Device Control**.

2. Using the datasheet for your DUT, configure the registers and fields in the register map.
3. Save your document.

Related concepts:

- Register Maps

Related tasks:

- Using a Register Map

Using a Register Map

Use a register map to view and modify DUT register values.

Complete the following tasks before using a register map:

- Add and define a pin map for the DUT.
- Create, reuse, or import a register map to your project.
- Create a source waveform and a capture waveform.
- Create patterns to read from and write to the DUT registers.

1. Open your project.
2. Open the register map (`.digireg`) from the Project Files pane.
3. On the toolbar, select the site of the register map you want to configure.
4. Enter the names of the read and write patterns.
   Match the pattern names specified in your read and write pattern (`.digipat`) documents.
5. Select and edit the field values in the register map table you want to update.
   You can only modify fields in the table that are writeable and not read-only. The values you modify will display as blue. Read-only fields are grayed out.
6. Depending on your changes, complete the following tasks:

Task	Action required
Write field values	◦ Click **Write all changes to registers** at the top of the document to write values to all fields. ◦ Click **Write changes to selected fields** in the context menu to write values to selected fields.
Clear edited field values	◦ Click **Clear all pending changes to registers** at the top of the document to return all fields to their previous values. ◦ Click **Clear pending changes to selected fields** in the context menu to return the selected fields to their previous values.
Read field values	◦ Click **Read all register values** at the top of the document to read all field values. ◦ Click **Read selected field values** in the

Task	Action required
	context menu to read selected field values.

Related tasks:

- Creating a Register Map

Related reference:

- Pin Map Tab
- Register Map Components

Register Map Components

Refer to the following information to successfully execute a register map in Digital Pattern Editor.

Read Pattern

The read pattern must be designed to read register values from the DUT. In most protocols this involves sending the register address and other bits like command bits and parity bits to the DUT, and receiving the register values and other bits from the DUT. To achieve this, the read pattern sources bits to send from a source waveform and captures bits received in a capture waveform.

Digital Pattern Editor packs the source waveform and parses the capture waveform based on the `SourceWaveformRegisterPacket` and `CaptureWaveformRegisterPacket` elements specified in the register map file `ReadTemplate`. If you do not specify a `SourceWaveformRegisterPacket` element, Digital Pattern Editor packs register addresses contiguously in chunks of length defined by `addressBitWidth`.

Note It is invalid to reference more than one source and one capture waveform in the read pattern.

When initiating a read operation in the register map document, Digital Pattern Editor

populates the source waveform and bursts the read pattern after setting the following pattern sequencer registers:

- `reg0`-- number of registers to read.
- `reg1`-- register address bit width. The `addressBitWidth` attribute in the register map file defines this value.
- `reg2`-- register value bit width. The `valueBitWidth` attribute in the register map file defines this value.
- `reg3`-- number of bits sent to the DUT to read one register. The `SourceWaveformRegisterPacket` element specified in the register map file `ReadTemplate` defines this value. If you do not specify this element, this is equal to the `addressBitWidth` attribute.
- `reg4`-- number of bits received from the DUT to read one register. The `CaptureWaveformRegisterPacket` element specified in the register map file `ReadTemplate` defines this value. If you do not specify a this element, this is equal to the `valueBitWidth` attribute.

The read pattern uses these pattern sequencer registers to iteratively send bits from the source waveform and receive bits in the capture waveform. On completion of the read pattern execution, Digital Pattern Editor parses the received register values from the capture waveform and updates affected fields in the register map document with new values.

Write Pattern

You must design the write pattern to write register values to the DUT. In most protocols, this involves sending the register address, register value, and other bits like command bits and parity bits to the DUT. To achieve this, the write pattern sources the bits to send from a source waveform.

Digital Pattern Editor packs the source waveform based on the `SourceWaveformRegisterPacket` element specified in the register map file `WriteTemplate`. If you do not specify a `SourceWaveformRegisterPacket` element, Digital Pattern Editor registers addresses and values contiguously in chunks of length defined by the sum of `addressBitWidth` and `valueBitWidth`.

Note It is invalid to reference more than one source waveform in the write

pattern.

When initiating a write operation in the register map document, Digital Pattern Editor populates the source waveform and bursts the write pattern after setting the following pattern sequencer registers:

- `reg0`-- number of registers to read.
- `reg1`-- register address bit width. The `addressBitWidth` attribute in the register map file defines this value.
- `reg2`-- register value bit width. The `valueBitWidth` attribute in the register map file defines this value.
- `reg3`-- number of bits sent to the DUT to write to one register. The `SourceWaveformRegisterPacket` element specified in the register map file `WriteTemplate` defines this value. If you do not specify this element, this is equal to the sum of the `addressBitWidth` and `valueBitWidth` attributes.

The write pattern iteratively sources bits from the source waveform using the above pattern sequencer registers and sends them to the DUT.

Source Waveform

Read and write template patterns require a source waveform. The patterns must not use more than one source waveform. The use of additional source waveforms will result in ambiguity when Digital Pattern Editor uses the source waveform to supply register addresses, values, and other bits like command bits and parity bits.

Note Read and write patterns do not have to use the same source waveform.

You must configure source waveforms with a sample width and sample count large enough to read all the registers in the register map. They must have a capacity equal to or greater than the product of the register count and the number of bits required to read or write one register.

For read patterns, this is specified by the `SourceWaveformRegisterPacket` element in the register map file `ReadTemplate` (or `addressBitWidth`, if `SourceWaveformRegisterPacket` is not specified). For write patterns, this is specified by the `SourceWaveformRegisterPacket` in the register map file

`WriteTemplate` (or the sum of `addressBitWidth` and `valueBitWidth`, if `SourceWaveformRegisterPacket` is not specified).

If possible, configure the sample count to equal the register count and the sample width to equal the number of bits sent to the DUT to read and write one register.

Note You must configure the source waveform with Broadcast data mapping and Serial waveform type.

Capture Waveform

Read template patterns require a capture waveform. The pattern must not use more than one capture waveform. The use of additional capture waveforms will result in ambiguity when Digital Pattern Editor uses the capture waveform to read the register values.

Note The write pattern is not restricted by the number of capture waveforms because Digital Pattern Editor does not use capture waveforms for write operations.

The read pattern stores received register value bits and other bits like parity bits in the capture waveform. You must configure the read pattern to capture and store the expected number of bits. This is the number of registers to read and the number of bits to be received per register determined by the `CaptureWaveformRegisterPacket` element in the register map file `ReadTemplate` (or `valueBitWidth` if `CaptureWaveformRegisterPacket` is not specified).

Digital Pattern Editor returns an error if the number of bits in the capture waveform does not match the expected number of bits. If possible, it is simpler to configure the sample width to equal the number of bits received from the DUT to read one register.

Note You must configure the capture waveform with Serial waveform type.

Related tasks:

- Using a Register Map

Register Map XML File Structure

Refer to the following information when configuring an XML file for use with a Digital Pattern Editor Register Map.

The register map XML schema, located at `<Program Files>\National Instruments\Digital Pattern Editor\RegisterMap.xsd`, defines the structure for a register map XML file.

Configuring a Register Map XML File

The root element of a register map file is a `RegisterMap` element. This element must specify the following required attributes:

- `schemaVersion` — this must have a value of `"1.0"`.
- `xmlns` — this must have a value of `http://www.ni.com/Semiconductor/RegisterMap`

The following sections describe elements within the `RegisterMap` element to configure.

Registers

`Registers` contains zero or more `Register` elements. You must configure the `addressBitWidth` and `valueBitWidth` for this element. For each `Register` element, you must specify the register `name`, hexadecimal `address`, and read-write access `type`.

Note Set the read-write type attribute to `ReadWrite`, `Read`, or `Write`.

Fields

`Fields` contains zero or more `Field` elements representing register fields in a register map document. Each `Field` element contains an `Comment` element, and zero or more `RegisterBits` elements. You must configure the `name` attribute for each `Field` element. Each `Field` element specifies the following attributes:

- `format`— set to one of the following values:
  - `Hexadecimal` — this is the default value if this attribute is not specified.

    - `Binary`
    - `Decimal`
    - `UnsignedDecimal`
    - `Boolean`
    - `Enumeration` — the `Field` element must specify an enum definition in the `enumDefinition` attribute.
- `enumDefinition`
- `reserved`— specifies the field is reserved for internal or future use

EnumDefinitions

`EnumDefinitions` contains zero or more `EnumDefinition` elements representing enums specified in the register map's `Field` element. Each `EnumDefinition` element specifies a list of `EnumMember` elements and the following attributes:

- `name`— the name of the enum definition.
- `format`— set to one of the following values:
    - `Hexadecimal` — this is the default value if this attribute is not specified.
    - `Binary`
    - `Decimal`
    - `UnsignedDecimal`
    - `Boolean`

You must specify the following attributes for each `EnumMember`:

- `name` — the name of the enum member.
- `value` — the value of the enum member.

RegisterBits

`RegisterBits`— these elements describe how a `Field` is composed from the bits of registers in the register map. You must specify values for the `register` and `bitRange` attributes for each `RegisterBits` element. The `bitRange` value is composed of two numbers separated by a colon. The number on the left of the colon specifies the most significant bit, and the number on the right of the colon specifies the least significant bit. For example, a specified value of `7:0` indicates bits 7 through 0 with bit 7 in the MSB position and bit 0 in the LSB position.

Note These numbers are restricted by the value of the `valueBitWidth` attribute.

Protocol

These elements specify how to read and write register values. The Protocol element contains the following:

- A `ReadTemplate` element specifying how to read the registers. This element contains:
  - `patternName`— an attribute specifying the name of the pattern to be burst to read register values.
  - `PatternSequencerRegisters`— an element containing zero or more pattern sequencer register values to set before executing a read operation.
  - `SourceWaveformRegisterPacket`— an element specifying how to format the source waveform for reading register values. If this element is not specified, Digital Pattern Editor will pack the source waveform with the addresses of the registers to read. This element contains the following:
    - `Comment`— an element describing the packet.
    - One or more `RegisterPacketSegment` elements that contain the following:
      - `Comment`
      - `CustomBits`— specifies a custom bit value, with an `comment` attribute.
      - `AddressBitRange`— the address bit range must be formatted as 2 numbers separated by a colon.
      - `ValueBitRange`— the value bit range must be formatted as 2 numbers separated by a colon.
      - `ParityBit`— set to either `EvenParity` or `OddParity`. Only one `ParityBit` element is allowed in a `RegisterPacketSegment`.
  - `CaptureWaveformRegisterPacket`— an element specifying how to parse the capture waveform for reading register values. If this element is not specified, then Digital Pattern Editor will parse the capture waveform assuming it is packed with register value bits. This element contains the following:
    - `Comment`
    - One or more `RegisterPacketSegment` elements that contain the following:

        - Comment
        - CustomBits— specifies a custom bit value, with an comment attribute.
        - AddressBitRange— the address bit range must be formatted as 2 numbers separated by a colon.
        - ValueBitRange— the value bit range must be formatted as 2 numbers separated by a colon.
        - ParityBit— set to either EvenParity or OddParity. Only one ParityBit element is allowed in a RegisterPacketSegment.
- An WriteTemplate element specifying how to write to registers. This element contains:
    - patternName— an attribute specifying the name of the pattern to be burst to write register values.
    - SourceWaveformRegisterPacket— an element specifying how to format the source waveform for writing register values. If this element is not specified, Digital Pattern Editor will pack the source waveform with the address and value of the registers to read. For example, when reading two registers named regA and regB, Digital Pattern Editor packs the source waveform with the address and value of regA, then the address and value of regB.
    - PatternSequencerRegisters— an element containing zero or more pattern sequencer register values to set before executing a write operation.

PatternSequencerRegister

PatternSequencerRegisters— an element containing zero or more pattern sequencer register values to set before executing a read operation. Each PatternSequencerRegister contains:

- Comment
- index— a required attribute specifying the pattern sequencer register index. Specify values from 8 to 15. reg0 to reg7 are reserved by Digital Pattern Editor.
- value— a required attribute specifying the integer value to set in the pattern sequencer register. Specify values from 0 to 65535.

Source Waveform Configurations

Use the source waveform configuration document to view, modify, save, and load

source waveform files (`.tdms`). Use the source functionality when the data you need to use is site-specific or only determined at run time, such as when you need to write registers or test converters. You can send multiple source waveforms with the same configuration in a single pattern burst. Use the `source_start` and `source` opcodes to send waveform data when you burst the pattern. An asterisk (`*`) next to a filename in the Project Explorer window and in the document tab indicates that the file has been modified but not saved.

Select **File** » **New** or click the **Add Item** button + on the Project Explorer window toolbar to create a new document. Create a single `.tdms` file for each individual source waveform you require. Double-click a `.tdms` file in the Project Explorer window or right-click the file and select **Open** from the context menu to launch the source waveform configuration document. The Digital Pattern Editor opens only `.tdms` waveform files that have the same structure as the waveform files the editor creates. You can also use the NI-Digital LabVIEW, .NET, or C API to configure and load source waveforms you create or edit with the source waveform configuration document. Refer to the ***Related reference*** section at the end of this topic for a link to more information about source and capture functionality.

Configuring Waveforms

You must specify values for the following components of the waveform. Options that do not apply dim based on the values you specify.

- **Waveform Name** — Specifies the name of the waveform.
- **Status** — Indicates one of the following states of the waveform:
  - **Not Loaded** — The waveform has not been loaded on the digital pattern instrument.
  - **Loaded** — The waveform has been loaded on the digital pattern instrument.
  - **Out of Sync** — The waveform has been loaded on the digital pattern instrument but has been modified since it was loaded.
- **Waveform Type** — Specifies one of the following types of waveforms:
  - **Serial** — Specifies that the waveform uses the serial configuration. When you configure the source functionality for serial use, each vector that contains a `source` opcode serially shifts 1 bit on the pin using the drive from source pin state (D) to indicate what value to drive. Only one pin per vector can source in this mode.

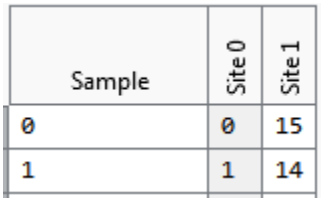
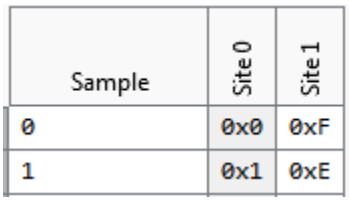
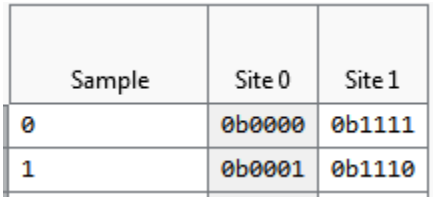
    - **Parallel** — Specifies that the waveform uses the parallel configuration. When you configure the source functionality for parallel use, the entire waveform sample is sourced on all specified pins in parallel for each vector that contains a `source` opcode. Use the drive from source pin state (D) for each pin on each such vector that corresponds to the pins to source.
- **Sample Format** — Specifies one of the following options to display data in decimal (no prefix), hexadecimal (0x prefix), or binary (0b prefix) format.
- **Sample Width** — Specifies the width in bits of each serial sample when you select **Serial** for the **Waveform Type**. Valid values are 1-32.
- **Bit Order** — Specifies whether to evaluate the most significant bit (MSB) or the least significant bit (LSB) first when sourcing a sample of a serial waveform.
- **Pins/Pin Groups** — Specifies the pins or pin groups configured for the waveform. Use the type-ahead drop-down menu to select valid pins and pin groups.
- **Data Mapping** — Specifies one of the following data mapping options:
    - **Broadcast** — Specifies to broadcast the waveform data in the **Site 0** column of the Source Data table to all sites.
    - **Site Unique** — Specifies to source unique waveform data for each site from the separate corresponding column for each site in the Source Data table. If using a pin with a shared pin connection, the waveform data must be identical for each of the sites that a pin is shared on.

The **Source Data** tab of the document displays editable source waveform data per site in columns. When you set **Data Mapping** to **Broadcast**, only the column for Site 0 is used. You can edit the data directly in the Source Data tab, or you can edit the source waveform file in another application and then click the **Import Source Data** button on the toolbar or the **Import From File** button on the Source Data tab to update the data to source. Use the **Number of Sites** control or right-click the source data table and select **Set Number of Sites** from the context menu to specify the number of sites to use.

Importing Source Waveform Data

To import data to a source waveform, click the **Import Source Data** button on the toolbar or click the **Import From File** button on the Source Data tab and select a `.csv` file of source waveform data in decimal, hexadecimal, or binary format to store and load with the waveform configuration.

The following table shows examples of text files and the corresponding Source Data tab.

Text File	Source Data tab
`0, 15` `1, 14`	
`0x0, 0xF` `0x1, 0xE`	
`0b0000, 0b1111` `0b0001, 0b1110`	

Loading Waveforms

The Digital Pattern Editor prompts you to load waveforms on the instrument before bursting a pattern if the pattern directly uses a waveform, including waveforms used in patterns that the pattern jumps to. You can also click the **Load Waveform** button on the toolbar or right-click the waveform file in the Project Explorer window and use the context menu to manually load the current waveform on the instrument. You must specify a name and at least one pin for the waveform. You cannot load a waveform with the same name as another already loaded waveform. Click the **Unload All Patterns and Waveform Configurations** button to remove all source and capture waveforms and patterns from the instrument. If you modify and save the waveform, you must unload and reload the waveform for the changes to take effect on the instrument. The Digital Pattern Editor prompts you to reload modified files before bursting a pattern.

When you load a parallel source or capture waveform through the Digital Pattern Editor, the order of pins you specify in the **Pins/Pin Groups** configuration determines the mapping for the bits in each sample. For example, the left-most bit maps to the

left-most pin you specify, and each subsequent bit maps to each subsequent specified pin. For pin groups, the left-most bit maps to the top pin of the pin group, and each subsequent bit maps to the subsequent pin in the pin group.

Keyboard Shortcuts within the Source Waveform Configuration File

Action	Shortcut
Load a source waveform file	<Ctrl+L>

Related reference:

- Digital Source and Capture Opcodes

Capture Waveform Configurations

Use the capture waveform configuration document to view, modify, save, and load capture waveform files (`.digicapture`). Use the capture functionality when the data you need to acquire is site-specific or only determined at run time, such as when you need to read registers or test converters. You can receive multiple capture waveforms with the same configuration in a single pattern burst. Use the `capture_start`, `capture`, and `capture_stop` opcodes to store values when you burst a pattern.An asterisk (`*`) next to a filename in the Project Explorer window and in the document tab indicates that the file has been modified but not saved.

Select **File** » **New** or click the **Add Item** button + on the Project Explorer window toolbar to create a new document. Create a single `.digicapture` file for each individual capture waveform you require. Double-click a `.digicapture` file in the Project Explorer window or right-click the file and select **Open** from the context menu to launch the capture waveform configuration document. You can also use the NI-Digital LabVIEW, .NET, or C API to load capture waveform configurations you create with the capture waveform configuration document. Refer to the ***Related reference*** section at the end of this topic for a link to more information about source and capture functionality.

Configuring Waveforms

You must specify values for the following components of the waveform. Options that do not apply dim based on the values you specify.

- **Waveform Name** — Specifies the name of the waveform.
- **Status** — Indicates one of the following states of the waveform:
  - **Not Loaded** — The waveform has not been loaded on the digital pattern instrument.
  - **Loaded** — The waveform has been loaded on the digital pattern instrument.
  - **Out of Sync** — The waveform has been loaded on the digital pattern instrument but has been modified since it was loaded.
- **Waveform Type** — Specifies one of the following types of waveforms:
  - **Serial** — Specifies that the waveform uses the serial configuration. When you configure the capture functionality for serial use, each vector that contains a `capture` opcode serially shifts 1 bit on pins using the pin state V.
  - **Parallel** — Specifies that the waveform uses the parallel configuration. When you configure the capture functionality for parallel use, the entire waveform sample is captured on all specified pins in parallel for each vector that contains a `capture` opcode. Use the pin state V for each pin on each such vector that corresponds to the pins for which you want to capture waveform samples.
- **Sample Format** — Specifies one of the following options to display data in decimal (no prefix), hexadecimal (0x prefix), or binary (0b prefix) format. Saving capture data uses the format you specify for the waveform file.
- **Sample Width** — Specifies the width in bits of each serial sample when you select **Serial** for the **Waveform Type**. Valid values are 1-32.
- **Bit Order** — Specifies whether to evaluate the most significant bit (MSB) or the least significant bit (LSB) first when capturing a sample of a serial waveform.
- **Pins/Pin Groups** — Specifies the pins or pin groups configured for the waveform. Use the type-ahead drop-down menu to select valid pins and pin groups.

Loading Waveforms

The Digital Pattern Editor prompts you to load waveforms on the instrument before bursting a pattern if the pattern directly uses a waveform, including waveforms used in patterns that the pattern jumps to. You can also click the **Load Waveform** button on the toolbar or right-click the waveform file in the Project Explorer window and use the

context menu to manually load the current waveform on the instrument. You must specify a name and at least one pin for the waveform. You cannot load a waveform with the same name as another already loaded waveform. Click the **Unload All Patterns and Waveform Configurations** button to remove all source and capture waveforms and patterns from the instrument. If you modify and save the waveform, you must unload and reload the waveform for the changes to take effect on the instrument. The Digital Pattern Editor prompts you to reload modified files before bursting a pattern.

When you load a parallel source or capture waveform through the Digital Pattern Editor, the order of pins you specify in the **Pins/Pin Groups** configuration determines the mapping for the bits in each sample. For example, the left-most bit maps to the left-most pin you specify, and each subsequent bit maps to each subsequent specified pin. For pin groups, the left-most bit maps to the top pin of the pin group, and each subsequent bit maps to the subsequent pin in the pin group.

Capture Results

Select **View » Capture Results** or click the **Show Capture Results View** button on the capture waveform configuration document toolbar or on the pattern document toolbar to view the capture waveform data from the instrument for the last pattern burst. The capture waveform data updates with each burst.

Related reference:

- Capture Results View

Capture Results View

Use this view to see capture waveform data per site in columns for the last pattern burst. The capture waveform data updates with each burst. Use the `capture_start`, `capture`, and `capture_stop` opcodes to store values when you burst a pattern.

Use the **Capture Waveform** pull-down menu to select the capture waveform for which you want to view data. Disabled sites do not return results. The pull-down menu displays multiple captures of the same waveform within the same burst as multiple items.

Click the **Save Capture Data** button on the Capture Results View toolbar or press <Ctrl+S> to save a `.csv` file that contains the data for the currently selected capture waveform.

Keyboard Shortcuts within Capture Results View

Action	Shortcut
Save all data for the currently selected capture waveform in a `.csv` file	<Ctrl+S>

DUT Bring-Up

Use the DUT Bring-up document (`.dutbringup`) to view, create, modify, save, and bring up your devices. The DUT Bring-up document displays the relay configuration, continuity test, power sequence, and bring-up pattern required to bring up your DUT for further operation. Complete the following steps to configure the DUT bring-up.

1. Click **File** » **New** » **DUT Bring-up**.
2. Define and apply the relay configuration.
   a. Specify the name of the relay configuration that you want to apply when bringing up the DUT.
   b. Specify optional delay to apply after configuring the relays.
   c. Click **Apply Relay Configuration** to apply specified relay configuration without executing other sections in this document.
3. Define the continuity test. A continuity test measures voltage drop on the specified pins on the DUT without executing other sections of the DUT Bring-up document.
   a. Specify the aperture and settling times to apply to measurements taken on all channels.
   b. Add a continuity configuration row specifying the pin or pin group, its sites, current level, and voltage limit. Specify the minimum and maximum voltage drops to determine whether the test passes or fails.
   c. Run the test by clicking the **Run Continuity Test** button in the Continuity Test window.
4. Define the power sequence.
   The table displays configurations for each pin, including voltage level, sourcing state, live current, and live voltage. Pin items are powered up in the order they

appear in the table from top to bottom. Re-arrange rows to change their order.
   a. Specify the NI-DCPower and NI-Digital PPMU pin items you want to use to power up your DUT.
   b. Specify optional delay to apply after powering up each pin.
   c. Specify power levels for these pins in the active levels sheet.
   d. Click **Power Up** to apply power to a pin without executing other sections in this document.
   e. Click **Power Down** to power down these pins. Pin items are powered down in the reverse order they appear in the table, from bottom to top, without applying specified delays.
5. Define the bring-up pattern.
   a. Specify the pattern to burst to bring up your DUT.
6. Click **Run DUT Bring-up** on the toolbar.

Configuring Time Domain Reflectometry

Create a time domain reflectometry (`.digitdr`) document to save a TDR configuration for your project. TDR measures latency introduced by your set-up, and applies the measured values to the channels you select to ensure accurate measurements.

The TDR document groups pins by channel, rather than by site, enabling faster test configuration across sites that share pins.

1. Select **File** » **New** » **TDR Configuration**.
2. Expand **Comments** to add information about the TDR test configuration.
3. Check the boxes next to each channel you want to include in the TDR measurement.
4. Depending on your goals, use the following table to determine which actions to take once you set up your TDR measurement document.

Goal	Button
Apply TDR offset to the specified channel.	Apply TDR
Execute a TDR test on shorted channels.  **Note** Only select this option if the selected channels are	Run TDR to Short

Goal	Button
shorted.	
Execute a TDR measurement on open channels.	Run TDR to Open
Update the TDR document to display channels from the active pin map.	Update from Pin Map

5. To remove channels no longer present in the active pin map, right-click the TDR configuration table and select **Remove Unmapped TDR Entries**.

Keyboard Shortcuts

Table 1. Keyboard Shortcuts Within the Digital Pattern Editor

Action	Shortcut
Throughout the Digital Pattern Editor	
Launch search dialog box	<Ctrl+F>
Zoom in and out	<Ctrl+scroll>
Launch a specific topic within the Digital Pattern Editor Help	F1
Launch the Digital Pattern Editor Help	<Ctrl+F1>
Unload all patterns and waveforms	<Ctrl+Shift+U>
Save	<Ctrl+S>
Save all	<Ctrl+Shift+S>
Show/hide panes	<Ctrl+\>
Close a document	<Ctrl+W>
Close a project	<Ctrl+Shift+W>
Exit the Digital Pattern Editor	<Alt+F4>
Within the Levels Document	
Load and apply levels sheets	<Ctrl+L>
Within the Timing Document	

Action	Shortcut
Apply time sets	<Ctrl+L>
Within the Pattern Document	
Load a pattern	<Ctrl+L>
Burst a pattern	F5
Launch a dialog box in which you can indicate a vector number to highlight and locate in the pattern document	<Ctrl+G>
Select active editor	<Ctrl+E>
Expand all scan vectors	<Ctrl+Shift+Add Sign> **Note** Use the Add Sign (+) on the numeric keyboard.
Collapse all scan vectors	<Ctrl+Shift+Minus Sign> **Note** Use the Minus Sign (-) on the numeric keyboard.
Within Pattern Grid View	
Start a new line inside a cell in the comments column	<Ctrl+Enter>
Start in-place editing	F2
Find next	F3
Find next occurrence of current selection	<Ctrl+F3>
Find previous	<Shift+F3>
Find previous occurrence of current selection	<Ctrl+Shift+F3>
Expand selected scan vector	<Ctrl+Add Sign> **Note** Use the Add Sign (+) on the numeric keyboard.
Collapse selected scan vector	<Ctrl+Minus Sign>

Action	Shortcut
	Note Use the Minus Sign (-) on the numeric keyboard.
Select all	<Ctrl+A>
Copy	<Ctrl+C>
Cut	<Ctrl+X>
Paste	<Ctrl+V>
Undo	<Ctrl+Z>
Redo	<Ctrl+Y>
Within Pattern Waveform View	
Control panning	Scroll bars, mouse wheel, or drag
Control zooming	<Ctrl+mouse wheel>
Expand all scan vectors	<Ctrl+Shift+Add Sign> **Note** Use the Add Sign (+) on the numeric keyboard.
Expand selected scan vector	<Ctrl+Add Sign> **Note** Use the Add Sign (+) on the numeric keyboard.
Collapse all scan vectors	<Ctrl+Shift+Minus Sign> **Note** Use the Minus Sign (-) on the numeric keyboard.
Collapse selected scan vector	<Ctrl+Minus Sign> **Note** Use the Minus Sign (-) on the numeric keyboard.

Action	Shortcut
Within History RAM View	
Find in pattern	<Ctrl+F>
Find next	F3
Find previous	<Shift+F3>
Save all History RAM results in the view in a `.csv` file	<Ctrl+S>
Within the Digital Scope	
Run Digital Scope	F5
Control panning	<Ctrl+left-click> and drag
Zoom on a region you select	<Shift+left-click> and drag
Save actual and expected plot data for the active site to a `.csv` file	<Ctrl+S>
Within the Shmoo Plot	
Run Shmoo	F5
Control panning	<Ctrl+left-click> and drag
Zoom on a region you select	<Shift+left-click> and drag
Zoom out	<Shift+right-click>
Within the Source Waveform Configuration File	
Load a source waveform file	<Ctrl+L>
Within Capture Results View	
Save all data for the currently selected capture waveform in a `.csv` file	<Ctrl+S>

Programming with the NI-Digital Pattern Driver

You can call pin and channel map, specifications, levels, timing, pattern, and waveform files you create in the Digital Pattern Editor in the NI-Digital Pattern Driver LabVIEW, .NET, or C API to configure and control digital pattern instruments.

Note You must install the NI-Digital Pattern Driver for content to display in this section of the help file.

Related reference:

- Getting Started with Digital Pattern Instruments
- NI-Digital Pattern Driver Examples
- NI-Digital Pattern Driver C Function Reference

Session State Model

As illustrated in the following image of a pattern state model, an NI-Digital Pattern Driver session can be in one of three programming states:

- Uncommitted
- Committed
- Running

Most digital pattern instrument settings do not take effect immediately when configured with the NI-Digital Pattern Driver. Rather, settings take effect when they are committed.

Note The PPMU has a simpler session state model than the pattern sequencer. PPMU settings related to sourcing are committed when you call the niDigital PPMU Source VI, the DigitalPpmu.Source .NET method, or the niDigital_PPMU_Source C function. PPMU settings related to measuring take effect when you call the niDigital PPMU Measure VI, the DigitalPpmu.Measure

.NET method, or the niDigital_PPMU_Measure C function.

Figure 3. NI-Digital Pattern Driver Session State Diagram

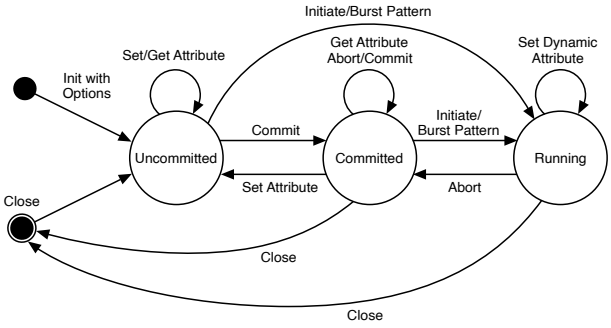


Table 2. Session States

State	Description
Uncommitted	A session enters this state after you initialize or reset. In this state, the instrument remains configured as it was the last time the session was committed. Settings you configure in this state are not applied to the instrument until it enters the Committed or Running states.
Committed	A session enters this state after you explicitly commit the session or call a VI, .NET method, or C function that commits the session. In this state, previously configured settings and certain automatically configured settings are applied to the instrument. **Note** Bursting a pattern automatically commits the pattern sequencer settings.

State	Description
	Note Reading or writing static data automatically commits driver, comparator, and active load settings. This includes V_{IH}, V_{IL}, V_{OH}, V_{OL}, active load, and termination mode.
Running	The session enters this state after you burst a pattern or initiate. In this state, you can dynamically reconfigure certain settings. For example, you can dynamically reconfigure all driver, comparator, and active load settings. This includes V_{IH}, V_{IL}, V_{OH}, V_{OL}, active load, and termination mode. The session exits this state after you call the niDigital Abort VI, the DigitalPatternControl.Abort .NET method, the niDigital_Abort C function, the niDigital Abort Keep Alive VI, the DigitalPatternControl.AbortKeepAlive .NET method, or the niDigital_AbortKeepAlive C function. **Note** Closing or resetting a session automatically aborts the session.

NI-Digital Pattern Driver LabVIEW Reference

Refer to the ***NI-Digital Pattern Driver LabVIEW Reference*** to view the VIs and properties included with the NI-Digital Pattern Driver. Use the NI-Digital VIs and properties in LabVIEW to configure and control the digital pattern instrument. The NI-Digital Pattern Driver LabVIEW API is installed by default when you run the installer.

The NI-Digital Pattern Driver palette VIs are available on the LabVIEW Functions palette

by browsing to **Measurement I/O » NI-Digital** or **Instrument I/O » Instrument Drivers » NI-Digital**.

Related information:

- NI-Digital Pattern Driver LabVIEW Reference

NI-Digital Pattern Driver .NET Reference

Refer to the ***NI-Digital Pattern Driver .NET Class Library*** to view the classes and types included with the NI-Digital Pattern Driver .NET class library. You can optionally install and use the NI-Digital Pattern Driver .NET API to configure and control the digital pattern instrument.

You can use the NI-Digital Pattern .NET class library by adding a reference to NationalInstruments.ModularInstruments.NIDigital.Fx40 or NationalInstruments.ModularInstruments.NIDigital.Fx45 and any dependent class libraries from within the Solution Explorer in Visual Studio.

If you are using the NI-Digital .NET Class Library, the .NET 4.0 Framework (minimum) is required. The NI-Digital .NET Class Library can be used with the Visual C# or Visual Basic .NET programming languages in any Visual Studio version that can target the .NET 4.0 Framework or the .NET 4.5 Framework.

Note Ensure that you select **.NET Framework 4.0 Languages Support** or **.NET Framework 4.5 Languages Support** in the NI-Digital Pattern Driver installer to install .NET support for the NI-Digital Pattern Driver software.

Related information:

- NI-Digital Pattern Driver .NET Class Library

Using the NI-Digital Pattern Driver .NET Class Library

You can use the NI-Digital Pattern Driver .NET class library to configure and control an NI Digital Pattern Instrument.

In the **Start** menu, navigate to **NI Digital Pattern Examples** in the **National Instruments** folder to access example applications.

Refer to `ni.com/mstudio` or visit `ni.com/info` and enter the Info Code `NIdotNET` for additional information on developing applications using NI drivers and the .NET Framework.

Note NI-Digital Pattern Driver documentation for .NET Framework 4.5 is available in the `<IVIinstalldir>\Drivers\niDigital\Documentation\English` directory. The ***Digital Pattern Help*** links to the .NET Framework 4.0 documentation.

Related information:

- NI-Digital Pattern Driver .NET Class Library

Working with Pin Sets

Use a DigitalPinSet object in the NI-Digital Pattern Driver .NET API to configure settings on your pins, such as the digital levels or the selected function. A pin set is a set of channels, pins, pin groups, or site-specific pins. Use the GetPinSet method on the PinAndChannelMap property of an NIDigital session to get a DigitalPinSet.

The pin map must define the pins and pin groups a pin set uses.

Creating a Pin Set

In order to perform operations only on specific pins, you can create a specific, named subset of pins, called a pin set.

A pin set can be composed of an individual channel, list of channels, individual pin, list of pins, individual pin group, list of pin groups, or a site-specific pin; however, you cannot mix channels and pins. You can mix pins, pin groups, and site-specific pins.

Note Operations apply to all items within the pin set. For example, if you set the termination mode on a DigitalPinSet object, the same termination mode

applies to all pins defined in the pin set.

The following code creates a DigitalPinSet that represents pins `DUTPin1` and `DUTPin2` and sets the selected function to Ppmu.

```
Visual C#

DigitalPinSet dutPins = session.PinAndChannelMap.GetPinSet("DUTPin1, DUTPin2");
dutPins.SelectedFunction = SeletedFunction.Ppmu;
```

```
Visual Basic .NET

Dim dutPins As DigitalPinSet = session.PinAndChannelMap.GetPinSet("DUTPin1,
DUTPin2")
dutPins.SelectedFunction = SelectedFunction.Ppmu
```

Pin Maps

A pin and channel map defines the relationships between DUT pins and system pins to the channels on an NI Digital Pattern Instrument. You can load an existing `.pinmap` file or programmatically create a new `.pinmap` file. Use the pins and pin groups defined in the `.pinmap` file to create pin sets.

You can use the Create Pin Map example to learn how to create a pin map programmatically. The .NET example applications are installed in the following location: `<Public Documents>\National Instruments\NI-Digital\Examples\DotNET 4.x`.

Note You can load only a single pin map during an NI-Digital Pattern Driver session. To switch pin maps, you must reset the device or create a new session.

Creating Pin Set Strings

The following table shows the correct syntax for creating a pin set string to use with GetPinSet or other methods that take a `siteList` parameter:

Group	Format	Example

Channels	<channelNumber>	`0`
		`0,1,7`
		`0-6`
		`0,1,7,8,9-14`

Pins	<underscore or any alphanumeric string>	`DUTPin1`
PinGroup	<underscore or any alphanumeric string>	`PinGroup1`
Site-Specific Pin	site<site number>/<pin name>	`site0/DUTPin1`
All Channels	empty string	`string.Empty`

Refer to DigitalPinSet for more information on creating pin set strings.

Selected Function

SelectedFunction is a property on the DigitalPinSet object that you can use to determine which instrument function controls the pin(s):

- **Digital**—The pin is connected to the driver, comparator, and active load functions. The PPMU is not sourcing, but can make voltage measurements. The state of the digital pin driver when you change the selected function to Digital is determined by the most recent call to WriteStatic or the last vector of the most recently executed pattern burst, whichever happened last. Use WriteStatic to control the state of the digital pin driver through software. Use BurstPattern to control the state of the digital pin driver through a pattern. Set the `selectDigitalFunction` parameter of BurstPattern to true to automatically switch the selected function of the pins in the pattern burst to Digital. You must set the termination mode and input and output voltages when SelectedFunction is set to Digital for the instrument.
- **Ppmu**—The pin is connected to the PPMU. The driver, comparator, and active load are off while this function is selected. Call DigitalPpmu.Source to source a voltage or current. DigitalPpmu.Source automatically switches the selected function to the

PPMU state and starts sourcing from the PPMU. Changing the selected function to Disconnect, Off, or Digital causes the PPMU to stop sourcing. If you change the selected function to PPMU using SelectedFunction, the PPMU is initially not sourcing.

> **Note** You can make PPMU voltage measurements using DigitalPpmu.Measure from within any selected function.

- **Off**—The pin is electrically connected, and the PPMU and digital pin driver are off while this function is selected.
- **Disconnect**—The pin is electrically disconnected from instrument functions. Selecting this function causes the PPMU to stop sourcing prior to disconnecting the pin.

> **Caution** In the Disconnect state, some I/O protection and sensing circuitry remains exposed. Do not subject the instrument to voltage beyond its operating range.

DigitalLevels

If the SelectedFunction of the pin set is Digital, use the DigitalLevels property to set the termination mode and the voltage levels.

PPMU

DigitalPinSet.Ppmu gets the DigitalPpmu object that contains the properties used to set the PPMU voltage and current levels, aperture time, and output function when the SelectedFunction for the DigitalPinSet is Ppmu.

You can use the PPMU to make DC parametric measurements on the DUT pins. Set the OutputFunction to determine whether to Source or Measure voltage or current. If the OutputFunction is DCVoltage, use the DCVoltage property on the DigitalPpmu to change DCVoltage settings. If the OutputFunction is DCCurrent, use the DCCurrent property on the DigitalPpmu to change DCCurrent settings.

The following code creates a DigitalPinSet and configures it to source DCVoltage.

```
Visual C#

DigitalPinSet ppmuPins = session.PinAndChannelMap.GetPinSet(“DUTPin1, DUTPin2”);
ppmuPins.SelectedFunction = SelectedFunction.Ppmu;
ppmuPins.Ppmu.OutputFunction = OutputFunction.DCVoltage;
ppmuPins.Ppmu.Source();
```

```
Visual Basic .NET

Dim ppmuPins As DigitalPinSet = session.PinAndChannelMap.GetPinSet("DUTPin1,
DUTPin2")
ppmuPins.SelectedFunction = SelectedFunction.Ppmu
ppmuPins.Ppmu.OutputFunction = PpmuOutputFunction.DCVoltage
ppmuPins.Ppmu.Source()
```

DigitalPpmu.Source() starts sourcing voltage or current on a per-pin basis from the PPMU to the DUT, depending on whether you specify DCCurrent or DCVoltage for the OutputFunction. Changing DigitalPpmu.OutputFunction or any other PPMU setting does not take effect until you call DigitalPpmu.Source, and DigitalPpmu.Source immediately commits the changes even if the PPMU is already sourcing. DigitalPpmu.Measure behaves the same way.

IVI Entry Points Not Supported in the NI-Digital Pattern Driver .NET API

The NI-Digital Pattern Driver .NET API does not support the following IVI entry points:

- DigitalDriverIdentity.GetGroupCapabilities
- DigitalDriverIdentity.GetSupportedInstrumentModels
- DigitalDriverOperation.Coercion
- DigitalDriverOperation.InterchangeCheckWarning
- DigitalDriverOperation.InvalidateAllAttributes
- DigitalDriverOperation.LogicalName
- DigitalDriverOperation.ResetInterchangeCheck
- DigitalDriverOperation.Warning
- DigitalDriverUtility.Disable
- DigitalDriverUtility.ResetWithDefaults

NI-Digital Pattern Driver C Function Reference

Refer to the ***NI-Digital Pattern Driver C Function Reference*** to view the functions and properties included with the NI-Digital Pattern Driver C API, which you can use to configure and control the digital pattern instrument. The NI-Digital Pattern Driver C API is installed by default when you run the installer.

You can use the NI-Digital Pattern C dynamically linked library by adding a reference to `C:\Program Files (x86)\IVI Foundation\IVI\Bin\niDigital_32.dll` for 32-bit development or to `C:\Program Files\IVI Foundation\IVI\Bin\niDigital_64.dll` for 64-bit development.

Related information:

- NI-Digital Pattern Driver C Function Reference