COMPREHENSIVE SERVICES

We offer competitive repair and calibration services, as well as easily accessible documentation and free downloadable resources.

SELL YOUR SURPLUS

We buy new, used, decommissioned, and surplus parts from every NI series. We work out the best solution to suit your individual needs.

Sell For Cash Get Credit Receive a Trade-In Deal

OBSOLETE NI HARDWARE IN STOCK & READY TO SHIP

We stock **New, New Surplus, Refurbished,** and **Reconditioned** NI Hardware.

APEX WAVES

Bridging the gap *between the manufacturer and your legacy test system.*

1-800-915-6216

www.apexwaves.com

sales@apexwaves.com

Request a Quote CLICK HERE *NI-9351*

SAFETY MANUAL

C Series Functional Safety

This document provides information about developing, deploying, and running Functional Safety systems using C Series Functional Safety modules.

C Series Functional Safety modules include the NI 9350 and the NI 9351. You can identify C Series Functional Safety modules by the yellow enclosure, yellow backshell, and SIL certification mark.

This is the May 2018 release version of the C Series Functional Safety Manual. Refer to the following table for release version information.

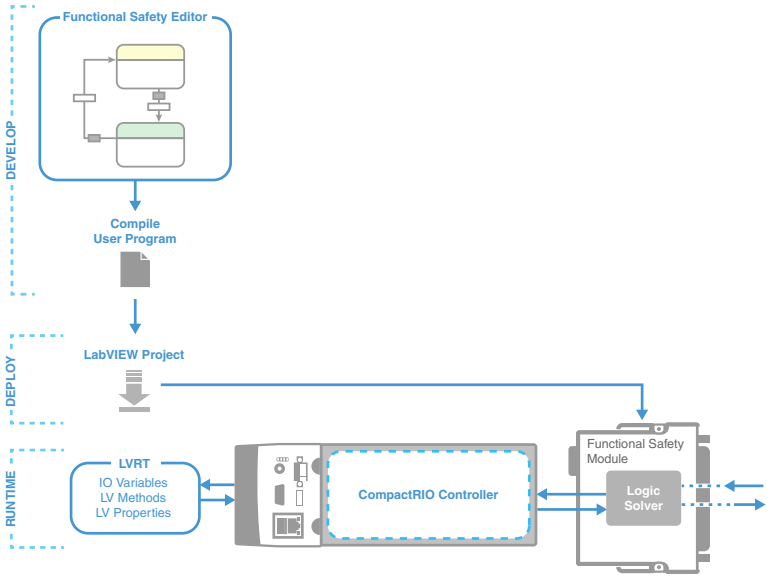
Table 1. C Series Functional Safety Manual Release Versions

Part Number	Release Date	Release Notes
377937A-01	September 2017	This is the initial release version. This version includes support for the NI 9350 and the Functional Safety Editor 2017.
377937C-01	May 2018	This version includes support for the NI 9351, the Functional Safety Editor 2018, and known issues resources for module firmware.

Contents

NATIONAL INSTRUMENTS

C Series Functional Safety Systems



Related Information

Finding Resources on page 99

Develop

Offline Development Tools

- LabVIEW—provides a platform for deploying and monitoring User Programs.
- Functional Safety Editor—provides a platform to facilitate the creation of safety User Programs.

What to Do

- Create a User Program in the Functional Safety Editor that implements the safety logic required by your safety instrumented function (SIF).
- Create a project in LabVIEW to download User Programs to the C Series Functional Safety module.
- (Optional) Develop a VI in LabVIEW to monitor module and channel status and to set outputs through digital passthrough.

Deploy

Offline, Non-Safety Support Tools

- CompactRIO controller—provides a hardware connection for deploying Safety Programs.
- LabVIEW—provides a software platform for deploying User Programs.

Functional Safety System Components

- C Series Functional Safety module—contains a logic solver that runs User Programs and provides I/O that connects to inputs, final elements, and a power supply.
- User Program—contains a set of user-defined logic and actions that run in the logic solver. The User Program defines the system's responses to inputs and detected faults.
- Cabling, sensors, final elements (actuators)—allows the C Series Functional Safety module to connect, monitor, and control safety critical systems.
- External LPS power supply—powers the C Series Functional Safety module.

What to Do

- Install and connect hardware components, including the CompactRIO controller, the C Series Functional Safety module, power supply, cabling, sensors, and final elements (actuators)
- Use the LabVIEW project to download the User Program to the logic solver on the C Series Functional Safety module while the module is not executing safety functionality.
- Validate the system by verifying system response to faults and system safety response time.

Runtime

Online, Non-Safety Support Tools

- CompactRIO controller—provides a hardware connection for monitoring Safety Programs and setting outputs through digital passthrough.
- LabVIEW—provides a platform for monitoring User Programs.

Functional Safety System Components

- C Series Functional Safety module—contains a logic solver that runs User Programs and provides I/O that connects to inputs, final elements, and a power supply.
- User Program—contains a set of user-defined logic and actions that run in the logic solver. The User Program defines the system's responses to inputs and detected faults.
- Cabling, sensors, final elements (actuators)—allows the C Series Functional Safety module to connect, monitor, and control safety critical systems.
- External LPS power supply—powers the C Series Functional Safety module.

What to Do

- Operate the safety User Program on the logic solver as part of your safety instrumented function (SIF).
- (Optional) Monitor the Functional Safety system through LabVIEW.

Functional Safety Overview

Safety design, process, and validation conducted for the C Series Functional Safety modules followed the standards outlined in IEC 61508:2010.

C Series Functional Safety modules are certified SIL3 capable Type B devices for use in continuous demand applications in simplex deployment configurations. The certification only applies to the C Series Functional Safety module. The CompactRIO chassis and LabVIEW are not safety-certified.

To view the IEC 61508 certificate with failure rates and assessment report from exida, go to *ni.com/info* and enter Info Code `safetycert`.

FMEDA Assumptions

The FMEDA results assume the C Series Functional Safety modules are used as logic solvers in De-Energize to Trip safety functions. All external circuits connected to the C Series Functional Safety module must apply the De-Energize to Trip principle.

Caution The De-Energize to Trip principle must be applied both to safety inputs and outputs.

Minimum Required Competency

All persons involved with planning, installing, connecting, or configuring software and hardware for use in safety systems that employ C Series Functional Safety modules must meet the following minimum competency requirements:

- Be informed about dependencies, risks, and consequences associated with safe operation, failure, and unsafe system conditions of any system employing C Series Functional Safety Modules.
- Have appropriate training and knowledge in the operation and implementation of industrial processes, measurement and control, automation, electrical engineering, and safety compliance.
- Have sufficient knowledge of all applicable codes, laws, regulations, and standards, including IEC 61508:2010.
- Be familiar with and have access to all requirements, conditions, specifications, and guidelines in all applicable NI documentation including hardware documentation for the C Series Functional Safety module and CompactRIO chassis and the C Series Functional Safety manual.

C Series Functional Safety Requirements

User Responsibilities

When deploying the safety system, users must:

- Create and configure the system HMI
- Define the system response for diagnostics in the User Program
- Be aware of and account for all documented known issues

- Validate and test the safety system prior to deployment
- Verify the safety response time of the system
- Document the validation test plan and results to demonstrate 100% test coverage.
- Change the module's mode to Operational Mode

When operating the safety system, users must:

- Monitor the HMI and/or module LEDs
- Conduct periodic proof tests as required by the application
- Respond to faults and detected unsafe conditions according to the safety plan
- Call National Instruments if the Internal Fault LED flashes more than three times then pauses.

Hardware Requirements

- Follow all documented installation instructions, connection guidelines, and operating requirements for C Series Functional Safety modules and CompactRIO controllers employed in the safety system.
- Apply the De-Energize to Trip principle to all external circuits connected to the C Series Functional Safety module.
- You must use a limited power source (LPS) supply suitable to the safety needs and configuration of the implemented system. Implement one of the following options to ensure continued compliance with IEC 61010-1.
  - The Vsup must be powered from a Class 2 or Limited Power Source (LPS), SELV source, 30 V DC maximum.
  - The Vsup must be powered from a SELV source, 30 V DC maximum, with supplementary overcurrent protection in series, 8 A maximum breaking capacity at 120 s.
  - The C Series Functional Safety module and associated controller must be installed in an end-use fire enclosure.

Software Requirements

- Install application software and device drivers appropriate to your hardware configuration. Refer to the following table for software applications and device drivers that are compatible with C Series Functional Safety modules.

Table 2. C Series Functional Safety Software Compatibility

NI 9350	NI 9351
Functional Safety Editor 2017 or later	Functional Safety Editor 2018 or later
LabVIEW 2017 or later	LabVIEW 2017 SP1 or later
LabVIEW Real-Time Module 2017 or later	LabVIEW Real-Time Module 2017 SP1 or later
CompactRIO Device Drivers 17.0 or later	CompactRIO Device Drivers 17.6 or later

- You must download a compiled User Program to the C Series Functional Safety module. You can create a User Program using the NI Functional Safety Editor. To download the necessary software, go to *ni.com/info* and enter Info Code `safetydownload`.
- You must have a computer running 64-bit Windows 7, Windows 8.1, or later to install and use the Functional Safety Editor. The application is not compatible with 32-bit Windows versions.
- The LabVIEW Real-Time Module is only available in 32-bit. If you are using the LabVIEW Real-Time Module, you must download 32-bit application software and device drivers to a computer running a 64-bit operating system.
- You must verify and formally document that your safety application is not affected by any documented known issue. For a complete list of resources for determining the known issues for your software and firmware, refer to the *Finding Resources* section of this manual.

Note For minimum software support information, visit *ni.com/info* and enter the Info Code `swsupport`.

Related Information

Functional Safety Editor on page 15

Security Requirements

Implement the following measures to protect against manipulation or corruption of the safety system.

- Determine and implement levels of access for hardware and software elements of the safety system.
- Transfer data only over secure connections.
- Limit personnel access to the C Series Functional Safety modules and the CompactRIO controller.
- Use locked enclosures to house the C Series Functional Safety modules and the CompactRIO controller.
- Implement operator authentication protections for software and network connections.
- Apply network segmentation strategies, such as firewalls or VPN.

Note For detailed information about security best practices for CompactRIO systems, visit *ni.com/info* and enter the Info Code `safetysecurity`.

Proof Test

The C Series Functional Safety module does not require a proof test. You do not need to include the module in a proof test plan for low-demand applications.

Non-Safety Functionality

RIO Scan Interface downloads to the CompactRIO controller FPGA when you configure your system in the NI Measurement & Automation Explorer (MAX). Scan Interface manages non-safety communication between the C Series Functional Safety module and LabVIEW Real-Time.

Scan Interface allows you to do the following:

- Read the values of inputs, outputs, and variables
- Read the status of fault diagnostics
- Monitor and set the module's Operating Mode
- Set output values with the digital passthrough

Related Information

Installing Hardware on page 8
Module Operating Modes on page 13
Passthrough on page 37

Installing Functional Safety Tools

Installing Hardware

1. Follow the instructions and guidelines in the getting started guides, datasheets, user manuals, and other hardware documentation for CompactRIO controllers and the C Series Functional Safety modules on *ni.com/manuals*.
2. Install the CompactRIO controller and C Series Functional Safety module(s).
3. Configure the system in the Measurement & Automation Explorer (MAX).
4. Connect the C Series Functional Safety module(s) to sensors, devices, and final elements as dictated by system requirements.
5. Connect the C Series Functional Safety module(s) to an external power supply.

Installing Software

1. Refer to the LabVIEW Installation Guide on *ni.com/manuals* to install LabVIEW and the NI-RIO device drivers.

   Note Select **NI 935x Functional Safety Module Support** from the *LabVIEW Real-Time Software Wizard* when installing drivers on the controller.

2. Go to *ni.com/info* and enter Info Code `safetydownload`.
3. Download and install the Functional Safety Editor.

Developing a Functional Safety System

Running the Safety Editor

1. Launch the Functional Safety Editor.
2. Select **File»New»Safety State Machine**.

   Note To begin with an example state machine, navigate to **Help»Open examples...** and double-click the example of your choice.

Creating User Programs

1. The State Machine editor opens to the I/O Configuration table.
2. Select the **Module and Diagram** tab in the configuration pane.
   a) Specify the **NI Safety Module**.
   b) Update the Document name and the State Machine name.
3. Define properties for all inputs and outputs wired to the module based on the system configuration.
4. Press <Ctrl-E> to open the Diagram.
5. Add states and connect transitions as required by the safety plan.

Note To add additional state machines to a User Program, click the pull-down menu at the top of the state machine tab and select **Add New State Machine**.

Compiling User Programs

Follow these steps to compile documents and output User Programs in the Functional Safety Editor.

1. Verify that there are no alerts in the **Errors and Warnings** pane.
2. Press <Ctrl-S> to save the state machine.
3. Click the **Compile** button.
4. Verify the User Program has compiled correctly.

   If the compile fails, do the following:
   a) Review the Errors and Warnings pane for compile errors.
   b) Address all errors and warnings.
   c) Repeat steps 1 through 4.
5. Verify that all inputs, outputs, and variables configured in the I/O Configuration table are used in the state machine diagram.
6. Verify that all diagnostics listed in the **Faults** table have **Module failsafe** selected or are used in the state machine diagram.

Note You can review the following files to verify your User Program:

- *<filename>*`.json`
- *<filename>*`_errors.json`
- *<filename>*`_report.log`

Related Information

Saving and Compiling on page 49

Creating a Functional Safety Project in LabVIEW

1. Launch LabVIEW.
2. Click the **Create Project** button to display the Project Explorer window. You can also select **File»New Project** to display the Project Explorer window.

3. Double-click **Blank Project**.
4. Right-click the top-level project item in the Project Explorer window and select **New»Targets and Devices** from the shortcut menu to display the **Add Targets and Devices** dialog box.
5. Ensure that the **Existing target or device** radio button is selected.
6. Expand **Real-Time CompactRIO**.
7. Select the CompactRIO controller to add to the project and click **OK**.
8. Click Continue. LabVIEW adds the controller and all the modules to the project.
9. Click **Discover** in the **Discover C Series Modules?** dialog box if it appears.
10. Select **File»Save Project** and save the project.

Creating a Functional Safety Monitoring VI in LabVIEW

1. Right-click the Real-Time CompactRIO target item in the Project Explorer window.
2. Select **New»VI** from the shortcut menu to open a new VI front panel and block diagram.
3. Add channels or variables to block diagram to monitor inputs and outputs.
   a) Select the channel or variable nested under the module item in the Project Explorer window.

      Available channels and variables include:
      - Analog input (NI 9351 only)
      - Digital input
      - Digital output
      - State machine variables
      - User-configurable LED

   b) Drag and drop the channel or variable onto the block diagram.
4. Add the Invoke Node to the block diagram to monitor the module status, diagnostics, set the module mode, or manually start the User Program.
5. Add the Property Node to the block diagram to monitor the firmware version, User Program GUID, User Program version, or other information about the C Series Functional Safety module.

Note For detailed information about using method and variables with C Series Functional Safety modules, open the LabVIEW Help and navigate to **NI CompactRIO Device Drivers»Devices»Functional Safety Modules**.

Starting a User Program from LabVIEW

You can use an **Invoke Node** in LabVIEW to start the User Program on your C Series Functional Safety module. You must start the User Program from LabVIEW in the following situations:

- You disable auto start in the User Program by deselecting the box on the **Module and Diagram** tab of the Functional Safety Editor.
- User-configurable faults in the User Program trigger Fail-safe Mode.

What to Do

1. Drag the C Series Functional Safety module (NI 935x) from the LabVIEW project and drop it onto the block diagram to create a reference constant.
2. Right-click the reference constant and select **Create»Method for 935x Class»Start Program** to place the Invoke Node.
3. Wire the reference constant to the reference terminal on the Invoke Node.

Deploying a Functional Safety System

Downloading User Programs

Follow these steps to download your User Program to the C Series Functional Safety module.

Note This procedure assumes you are interacting with the chassis in Scan Interface mode only. If your chassis is running in hybrid mode, stop any LabVIEW program running on the RT target in your LabVIEW project, set the chassis to Scan Interface mode, and deploy the chassis before downloading your User Program.

1. Open the LabVIEW project (`.lvproj`) created to monitor the safety system.
2. Right-click the module in the LabVIEW project and select **Properties**.
3. Click the **Read Module** button in the **Current User Program** section.
4. Verify current **Build Number** and **Program GUID**.

   If no User Program has been downloaded to the module, the fields will display as follows:
   - **Build Number**: **0**
   - **Program GUID**: **{00000000-0000-0000-000000000000}**
   - **Mode**: **Unprogrammed**
5. Click the folder icon next to the **Path to New User Program** field in the **New User Program** section.
6. Locate and double-click the User Program (`.bin`).
7. Click the **Download Program** button to deploy the selected User Program to the C Series Functional Safety module.

   The **Download Program** window will open.
8. Type `yes` and click **OK**.

   The **Download Message** field will indicate successful completion or error. In the case of an error, click the **Details** button for more information.
9. Verify the **Build Number** and **Program GUID** fields have updated to match the build number and program GUID of the new User Program.

   In the Functional Safety Editor, the build number and program GUID are displayed on the **Module and Diagram** tab of the configuration pane.
10. Verify the module mode has updated to `Verification Mode` in the **Mode** field.

11. Click **OK**.

Related Information

Saving and Compiling on page 49

Verifying User Programs

Complete the following steps to change the mode to Operational Mode.

Note Verify that the User Program responds as expected for all configured faults.

Note Verify the safety response time for all configured faults.

1. Open the LabVIEW Project (`.lvproj`) created to monitor the safety system.
2. Right-click the module in the LabVIEW Project and select **Properties**.
3. Click the **Change Mode to** button.
4. Type `verify` and click **OK**.
5. Verify the module mode has updated to `Operational Mode` in the **Mode** field.

Validating a Functional Safety System

1. Perform necessary system tests before implementation as required by safety plan.

   Note System testing must provide 100% coverage for all transition statements and signal values in the User Program.

2. Create formal documentation to record system test plan and test results and to demonstrate 100% coverage.

Functional Safety Hardware

Module Independence

The C Series Functional Safety module is independent of the CompactRIO controller. The module must be powered by an external power supply. Loss of controller power or communication with the controller does not affect the safety functionality of the module.

Module Logic Solver (FPGA-based)

The primary safety function of the C Series Functional Safety module is to read inputs and set outputs based on safety logic defined in the User Program. A logic solver runs the User Program on an FPGA in the C Series Functional Safety module.

Note Any instance of the term *FPGA* in this manual refers to the FPGA internal to the C Series Functional Safety module that runs the module firmware, the logic solver, and the User Program, unless the instance explicitly indicates the controller FPGA.

Module Operating Modes

- The module runs in Unprogrammed Mode when you first install and power on the module.
- While the User Program is downloading, the module runs in User Program Download mode.
- After a successful download, the module changes to Verification Mode.

Note In Verification Mode, the User Program is running normally.

- Perform validation procedures on your system while in Verification Mode.
- Change the mode to Operational Mode from your project in LabVIEW once validation of the system is complete.

The module will run in Operational Mode until one of the following things happen:

- You change the mode back to Verification Mode in LabVIEW.
- You cycle external power to the module.
- User-configured diagnostics or automatic self-diagnostics trigger Fail-safe Mode.

Note The module FPGA stops the User Program when the module changes from Verification Mode to Operational Mode or from Operational Mode to Verification Mode. If you enable auto start, the User Program will restart after the module changes modes. If you do not enable auto start, you will need to restart the User Program from LabVIEW.

Note Latched faults persist when the module changes operating mode. For more information on fault latching, refer to the *Fault Latching* section.

The RIO Scan Interface monitors and returns the module operating mode. You can view or change the operating mode in the Properties window in the LabVIEW project or with the Invoke Node in your LabVIEW VI.

Table 3. Module Operating Modes

Mode	What Is Happening	What to Do Next
Unprogrammed Mode	• Hardware state out of the box • User Program is not written to the module • Vsup/Status LED flashes	• Develop the User Program in the Functional Safety Editor • Download the User Program to the module
User Program Download Mode	• User Program is downloading to the module • Vsup/Status LED flashes	• Verify the mode updates to Verification Mode • Verify the Build Number and the Program GUID update

Table 3. Module Operating Modes (Continued)

Mode	What Is Happening	What to Do Next
Verification Mode	• User Program has downloaded to module and is running normally • User Program requires verification • Vsup/Status LED flashes	• Use this mode to perform necessary verifications based on system design • Monitor system for detected faults • Set module to Operational Mode
Operational Mode	• User Program is running on the module • Vsup/Status LED is on	• Perform maintenance and proof tests as determined by your safety plan • Monitor system for detected faults
Fail-safe Mode	• All outputs are de-energized • User Program stops running • Vsup/Status LED flashes • Internal Fault LED flashes • LabVIEW returns fault status information	• Respond to fault as determined by user safety plan • Cycle external V_{sup} to the module • Restart the User Program • Return the module to Operational Mode as defined by your safety plan

Related Information

Starting a User Program from LabVIEW on page 10

Fail-Safe Mode

Fail-safe Mode de-energizes all outputs from the C Series Functional Safety module and stops the User Program. You can still read diagnostics, inputs, and the module status in Scan Interface, but the User Program is no longer running. However, depending on the condition that triggered Fail-safe Mode, the data returned by Scan Interface may not be correct.

You can configure the User Program to trigger Fail-safe Mode in response to faults in the I/O Configuration table in the Functional Safety Editor. If a user-configurable fault triggers Fail-safe Mode, you must cycle external V_{sup} power to the module and restart the User Program using the Invoke Node in the monitoring VI in LabVIEW.

Automatic self-diagnostics will trigger Fail-safe Mode independently of User Program. If an automatic self-diagnostic triggers Fail-safe Mode, identify the condition causing the fault and remove it. For more information on automatic self-diagnostics, refer to the Automatic Self-Diagnostics section. Then, to exit Fail-safe Mode, cycle external V_{sup} power to the module.

The User Program will start automatically if auto start is enabled. Otherwise, restart the User Program using the Invoke Node in the monitoring VI in LabVIEW.

Related Information

Setting Faults to Trigger Fail-Safe Mode on page 33
Starting a User Program from LabVIEW on page 10

Power Down Mode

In Power Down Mode, the C Series Functional Safety powers off and ceases all operation. The User Program stops running, all outputs de-energize, and the module no longer communicates with LabVIEW.

Automatic self-diagnostics can trigger Power Down Mode in the following situations:

- Short condition on both DO FETs on a single output channel
- Overvoltage on V_{sup}
- Internal overvoltage faults

If the module goes into Power Down Mode, follow these steps:

1. Inspect all inputs and outputs to verify they are within specifications.
2. Cycle external V_{sup} to the module.
3. Contact NI if the module goes into Power Down Mode a second time.

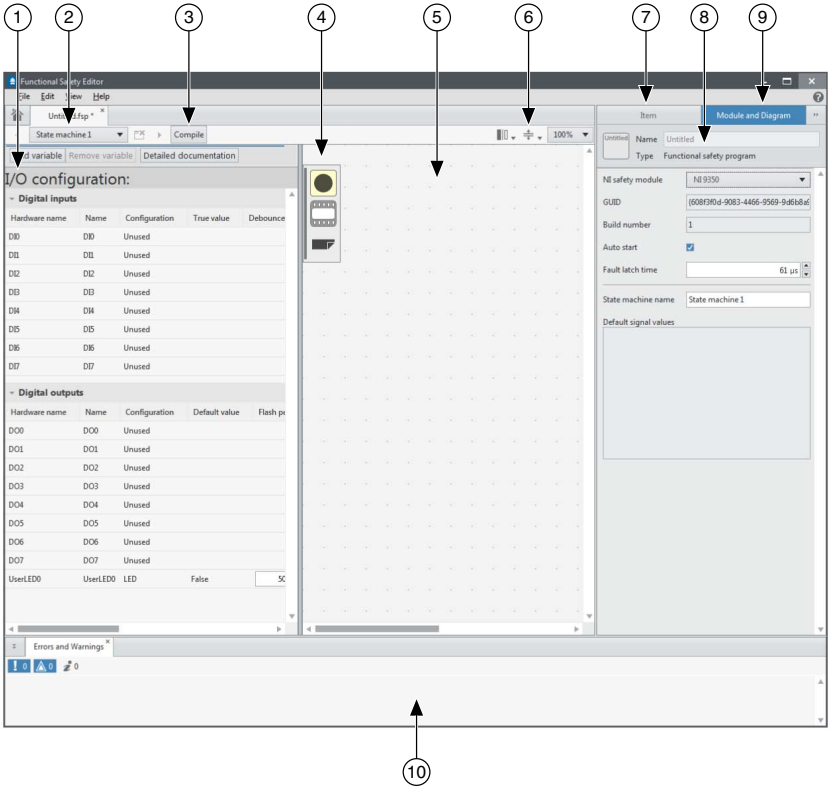
Functional Safety Editor

The Functional Safety Editor provides an interface to create and compile User Programs that implement the safety logic for your application. The compiled User Program deploys and runs on the module logic solver. Each User Program supports multiple state machines that run in parallel. Create up to eight state machines for the NI 9350 and create up to four state machines for the NI 9351.

The Functional Safety Editor allows users to do the following:

- Add states from the palette and define output behavior for those states
- Connect states with transitions and define input triggers for those transitions
- Configure input and output channels and variables in the I/O Configuration table
- Set default output values and variables for state machines and compound states

Figure 1. Functional Safety Editor



Use the following elements to navigate and configure the Functional Safety Editor.

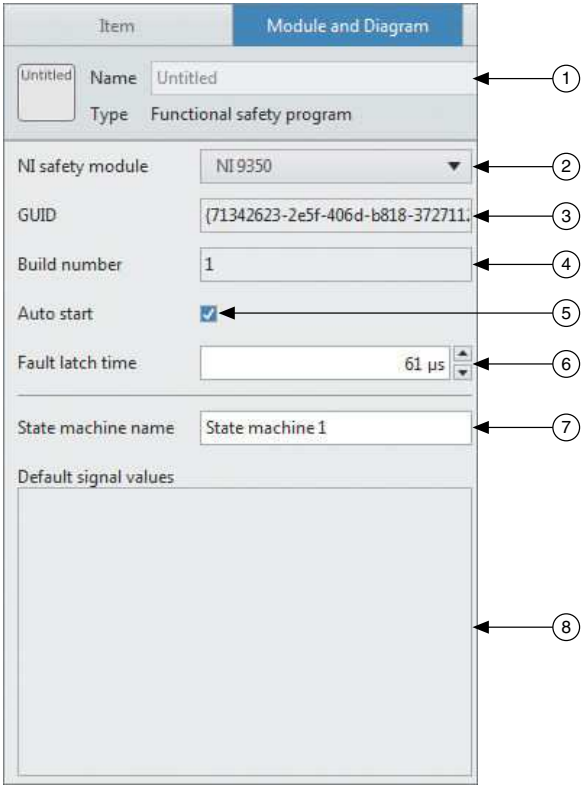
1. **I/O Configuration table**—Use this table to configure the parameters for all inputs, outputs, variables, and faults used in your User Program.
2. **State machine menu**—Use this pull-down menu to switch between state machines or add state machines to the User Program.
3. **Compile button**—Click this button to compile your User Program. The compiler will generate a binary file you can download to your C Series Functional Safety module.
4. **Palette**—Use the palette to drag and drop simple states, compound states, and comments.
5. **State machine diagram**—Use this diagram to build your state machine. Add states from the palette and connect them with transitions.
6. **Switch view button**—Click this button to switch between the state machine diagram, the I/O Configuration table, or a split view. You can also switch between the I/O Configuration table and the state machine diagram by pressing <Ctrl-E>.
7. **Item tab**—Select this tab to update properties or access help documentation for the currently selected item in the state machine diagram.
8. **Configuration pane**—Use this pane to view the **Item** tab or the **Module and Diagram** tab.

9. **Module and Diagram tab**—Select this tab to update properties for the module and the User Program.
10. **Errors and Warnings pane**—Refer to this pane for possible issues with syntax or design of the User Program.

Module and Diagram Tab

The **Module and Diagram** tab allows you to configure settings for the C Series Functional Safety module and for the User Program.

Figure 2. Module and Diagram Tab



1. **Name**—Displays the filename of the `.fsp`
2. **NI safety module**—Selects the C Series Functional Safety module that will run the User Program
3. **GUID**—Displays the unique ID of the User Program
4. **Build Number**—Displays the build number of the User Program
5. **Auto start**—Disables or enables the auto start function for the User Program
6. **Fault Latch Time**—Sets the fault latch time for the User Program

7. **State machine name**—Sets the name for the current state machine in the User Program
8. **Default signal values**—Displays the default signal values set on the I/O Configuration table

Build Number

- Build number allows you to track versions of your User Program.
- When you create a new User Program, the initial build number on the **Module and Diagram** tab is **1**.
- The binary file includes the current build number when it compiles.
- You can verify the build number sent to the compiler by checking the JSON.
- The build number on the **Module and Diagram** tab increments when you first edit a User Program that has successfully compiled.
- When you download a binary file to the module, you can confirm the build number and GUID of the binary file in the **Properties** dialogue in the LabVIEW project.

Auto Start

- The auto start function starts the User Program under the following conditions:
  - When you cycle external power to the module
  - After successful download of a User Program
  - On power up
  - When you change operating modes
- Auto start is enabled by default. You can disable or enable auto start with the **Auto start** checkbox on **Module and Diagram** tab.
- If auto start is enabled, the User Program starts when the module changes to Verification Mode after a successful download.
- If auto start is disabled, users must restart the User Program from the Start Program Method in LabVIEW.
- Auto start disables when the User Program triggers Fail-safe Mode.

  Tip When the User Program triggers Fail-safe Mode, auto start disables, preventing Fail-safe Mode loops and allowing you to download a new User Program.
- Cycling external power twice after the module goes into Fail-safe Mode re-enables auto start.
- Auto start does not disable when automatic self-diagnostics trigger Fail-safe Mode.

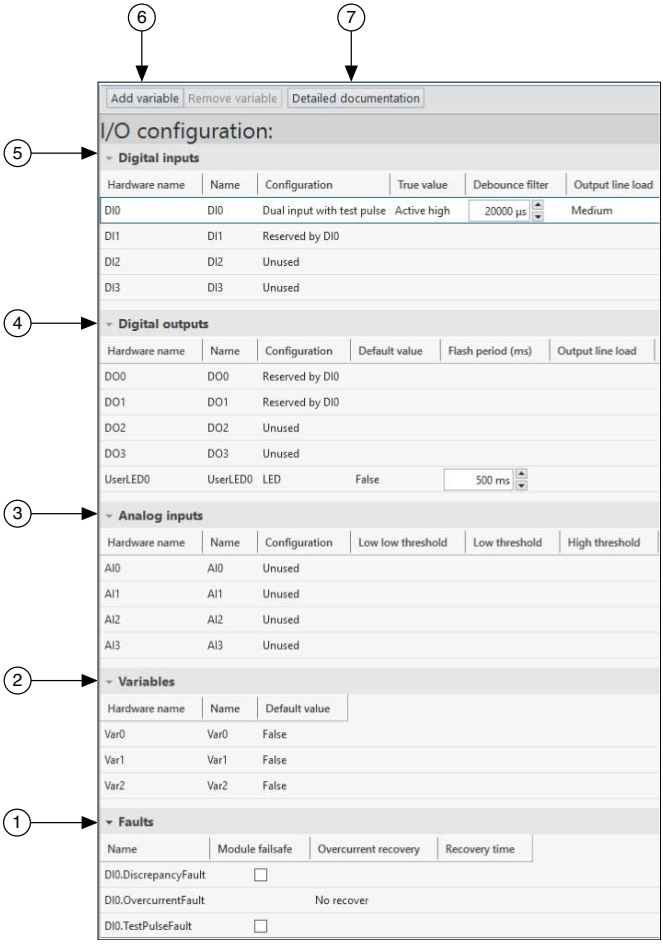
Related Information

Starting a User Program from LabVIEW on page 10

I/O Configuration Table

The I/O Configuration table allows you to configure parameters for all inputs, outputs, variables, and faults on the C Series Functional Safety module.

Figure 3. I/O Configuration Table



1. Faults table
2. Variables table
3. Analog inputs table
4. Digital outputs table
5. Digital inputs table
6. Add variable button
7. Detailed documentation button

A new functional safety program opens to the I/O Configuration table for the NI 9350, showing only the digital output and digital input tables.

- To create a functional safety program for the NI 9351, select NI 9351 from the NI safety module pull-down menu on the Module and Diagram tab. This will add the analog input table to the I/O Configuration.
- To switch between the I/O Configuration table and the state machine diagram, press <Ctrl-E> or click the **Switch View** button at the top of the state machine tab.

- To add and populate the variable table, click the Add variable button at the top of I/O Configuration table. To remove a variable, select the variable in the variable table and click the Remove variable button.
- To add and populate the **Faults** table, start configuring inputs and outputs. The **Faults** table will populate based on the configurations selected.
- To view an online version of the C Series Functional Safety manual, click the Detailed documentation button.

Related Information

Configuring I/O Channels on page 36

Digital Configurations

Table 4. Digital Input Configurations

Configuration	Channels		Notes
	NI 9350	NI 9351	
Single input	Available on any digital input channel.		—
Single input with test pulse	Available on any digital input channel.		Test pulse on DIn reserves DOn to generate the test pulse.
Dual input	Available on these channel pairs: [DI0, DI1], [DI2, DI3], [DI4, DI5], [DI6, DI7].	Available on these channel pairs: [DI0, DI1], [DI2, DI3].	A dual input on DIn, reserves DIn +1.
Dual input with test pulse	Available on these channel sets: [DI0, DI1, DO0, DO1], [DI2, DI3, DO2, DO3], [DI4, DI5, DO4, DO5], [DI6, DI7, DO6, DO7].	Available on these channel sets: [DI0, DI1, DO0, DO1], [DI2, DI3, DO2, DO3].	Test pulse on DIn reserves DIn+1, DOn and DOn+1.

Table 5. Digital Output Configurations

Configuration	Channels		Notes
	NI 9350	NI 9351	
Single output	Available on any digital output channel.		—
Single output with external readback	Available on any digital output channel.		Readback on external DOn reserves DIn.

Table 5. Digital Output Configurations (Continued)

Configuration	Channels		Notes
	NI 9350	NI 9351	
Single output with internal test pulse	Available on any digital output channel.		—
Single output with external test pulse	Available on any digital output channel.		Outputs a test pulse on DOn and reserves DIn to monitor test pulse.
Dual output	Available on these channel pairs: [DO0, DO1], [DO2, DO3], [DO4, DO5], [DO6, DO7].	Available on these channel pairs: [DO0, DO1], [DO2, DO3].	Dual output on DOn reserves DOn+1.
Dual output with Internal test pulse	Available on these channel pairs: [DO0, DO1], [DO2, DO3], [DO4, DO5], [DO6, DO7].	Available on these channel pairs: [DO0, DO1], [DO2, DO3].	Dual outputs with test pulses on DOn and DOn+1.
Dual output with external test pulse	Available on these channel pairs: [DO0, DO1, DI0, DI1], [DO2, DO3, DI2, DI3], [DO4, DO5, DI4, DI5], [DO6, DO7, DI6, DI7].	Available on these channel pairs: [DO0, DO1, DI0, DI1], [DO2, DO3, DI2, DI3].	Dual outputs with test pulses on DOn and DOn+1 and reserves DIn and DIn +1 to monitor test pulses.

Note Dual input and dual output configurations are only available on the even-numbered channel. Only the even-numbered channel will be available in the **Faults** table or on the state machine diagram.

Related Information

Digital Input Configurations on page 72
Digital Output Configurations on page 74
User-Configurable Digital Diagnostics on page 78

Analog Configurations (NI 9351 Only)

The NI 9351 has four analog input channels. You can use them to create the following configurations.

Table 6. Analog Configurations

Configuration	Channels	Notes
Single input (1oo1)	Available on any analog input channel.	Monitors current ranges for a single analog signal.
Dual input (1oo2)	A dual input configuration available on the following channel pairs: [AI0, AI1] and [AI2, AI3].	Establishes a 1oo2 voting strategy on two analog input channels. Configuring AI0 reserves AI1 and configuring AI2 reserves AI3.
Triple input (2oo3)	A triple input configuration is only available on AI0.	Establishes a 2oo3 voting strategy on three analog input channels. Configuring AI0 reserves AI1 and AI2.

Note Dual input (1oo2) configurations are only available on the even-numbered channel. Only the even-numbered channel will be available in the **Faults** table or on the state machine diagram.

Note A triple input (2oo3) configuration is only available on AI0. Only AI0 will be available in the **Faults** table or on the state machine diagram.

Related Information

Analog Input Configurations (NI 9351 Only) on page 76
User-Configurable Analog Diagnostics (NI 9351 Only) on page 86

Variables

- Variables are Boolean values used to communicate between individual state machines in a User Program and with Scan Interface.
- The User Program supports up to 24 variables.
- You can create variables in the I/O Configuration table by clicking the **Add variable** button.
- You can remove variables by selecting the variable you want to delete and clicking the **Delete variable** button.
- Only one state machine can write to a given variable.
- You can use variables as both signal values and transition conditions.
- Variables are read-only in Scan Interface.

Naming Channels and Variables in the I/O Configuration Table

Follow these guidelines when naming channels and variables in the I/O Configuration table:

- Rename the channel or variable by double-clicking the default name in the **Name** column.
- Use only Unicode 5.0 language-type characters.
- Do not use Boolean operators as names.

- Do not use spaces in channel or variable names. Replace spaces with underscores.
- Refer to the following table for a list of common keywords and operators that are not allowed for use in channel or variable names.

Note The Functional Safety Editor will not allow you to enter forbidden characters.

Table 7. Forbidden Keywords and Operators

Keywords		Operators		
after	or	||	+	!
true	and	&&	*	.
false	not	^^	( )	=

Digital I/O Parameters

When you select a configuration for a channel, the I/O Configuration table enables the appropriate parameters. Refer to the following table for the parameters associated with each configuration.

Table 8. I/O Configuration Parameters

Signal Type	Configuration	Parameters
Digital Inputs	Single input	True value, Debounce filter
	Single input with test pulse	Test pulse period, Test pulse width, True value, Debounce filter, Output line load
	Dual input	True value, Discrepancy time, Debounce filter, Complementary
	Dual input with test pulse	Test pulse period, Test pulse width, True value, Discrepancy time, Debounce filter, Complementary, Output line load

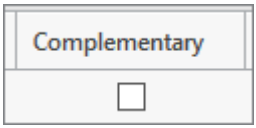
Table 8. I/O Configuration Parameters (Continued)

Signal Type	Configuration	Parameters
Digital Outputs	Single output	Default value, Output line load, Flash period
	Single output with external readback	Default value, Readback delay, Output line load, Flash period, Debounce filter
	Single output with internal test pulse	Default value, Test pulse period, Test pulse width, Output line load, Flash period
	Single output with external test pulse	Default value, Test pulse period, Test pulse width, Output line load, Flash period, Debounce filter
	Dual output	Default value, Output line load, Flash period
	Dual output with Internal test pulse	Default value, Test pulse period, Test pulse width, Output line load, Flash period
	Dual output with external test pulse	Default value, Test pulse period, Test pulse width, Output line load, Flash period, Debounce filter
UserLED0	LED	Default value, Flash period

Complementary

The complementary parameter configures how the User Program evaluates dual inputs. Check the complementary box to configure the dual inputs as complementary. Leave the box unchecked to configure the dual inputs as equivalent. The complementary parameter is available on the even-numbered channel.

Figure 4. Complementary



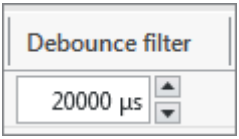
Related Information

Discrepancy Diagnostics for Digital Inputs on page 83

Debounce Filters

You can set debounce filters on any digital input channel.

Figure 5. Debounce Filter



Debounce filters are timers that debounce mechanical switches or filter noise and transitions.

The filter timer begins at the rising or falling edge of the unfiltered input signal. The User Program reads the previous value of the signal for the duration of the filter time. After the filter time elapses and no new edges on the input signal have occurred, the User Program reads the new signal value. The filter timer restarts at the next edge of the of the unfiltered input signal.

Figure 6. Debounce Filter on an Active High Input

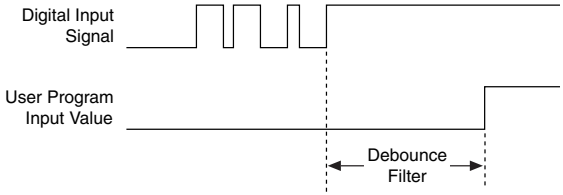
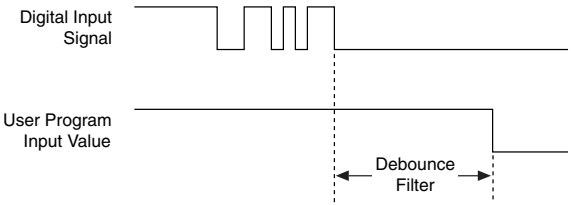


Figure 7. Debounce Filter on an Active Low Input



Refer to the following table to calculate maximum filter times. For information on calculating input signal response times, refer to the *Input Signal Response Time* section.

Table 9. Calculating Debounce Filter Times

DI Configuration	Filtered Signal Time Maximum	Detected Signal Time Minimum
Single input and dual input	*Debounce filter time* - 15 µs	Input signal response time (0 to 1)
Single input with test pulse and dual input with test pulse	*Debounce filter time* - (2 × *test pulse width*) - (2 × *debounce constant*) - 43 µs	

Tip To turn off filters, set filter value to 0.

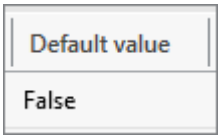
Note To use debounce filters with test pulses, refer to the *Filter Times for Test Pulses* section for maximum and minimum debounce filter values.

Note A debounce filter on digital inputs clears when the User Program first starts. Digital inputs that are true when the User Program starts will read false until the debounce filter time elapses.

Default Value

Default value is a required parameter that defines the default signal value for outputs, variables, and the UserLED0.

Figure 8. Default Value



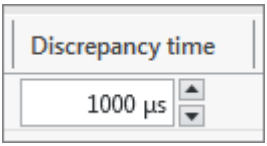
Related Information

Default Signal Values on page 36
Output Signal Value Syntax on page 41

Discrepancy Time (Digital Configurations)

Discrepancy time defines the delay before the User Program checks whether the signals are complementary or equivalent, based on your configuration.

Figure 9. Discrepancy Time



Dual input configurations introduce additional discrepancy due to signal routing and counter timebases. This results in a maximum tolerable discrepancy which is shorter than the configured parameter by the amount of an FPGA-based minimum discrepancy timer.

Maximum tolerable discrepancy = discrepancy time - minimum discrepancy timer

Refer to the following table to calculate the minimum discrepancy timer values based on the configuration.

Table 10. Calculating Minimum Discrepancy Timer Values

Debounce Filter Time	Dual Input	Dual Input with Test Pulse
0 µs < *debounce filter time* ≤ 50 µs	100 µs	—
50 µs < *debounce filter time*	2 × *debounce filter time*	(2 × *debounce filter time*) + *test pulse width*

Note You cannot set *debounce filter time* < 108 µs when using dual input with test pulse.

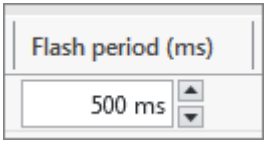
Related Information

Discrepancy Diagnostics for Digital Inputs on page 83

Flash Period

You can set the flash period for any output.

Figure 10. Flash Period

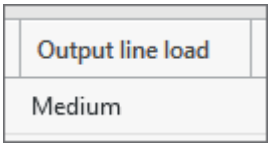


- The flash period is defined by the time the output is on plus the time the output is off. The output on/off time equals half of the flash period.
- Set the signal value to `DOn = flash` in the state machine diagram to use the flash period.
- Set the flash period large enough to allow the readback diagnostic to run: *Flash period* > 2 × *Readback response time*
- When using test pulses, set the flash period large enough to allow the test pulse to run: *Flash period* > 2 × *Test pulse period*

Output Line Load

You can set the line load for digital outputs or digital inputs with test pulses.

Figure 11. Output Line Load



Setting an appropriate output line load is necessary for test pulse and readback diagnostics. Heavy output line loads work for all applications within module specifications but will result

in slower response times. Reducing output line loads will enable shorter test pulses, readback delays, and faster response times.

There are two ways to set output line load:

- Calculate the discharge time using the following equation and the *Output Line Load for Input Discharge Times* table.
- Approximate the discharge time based on the configuration, external load, cable length and capacitance using the *Output Line Load Recommendations* table.

$$t = -R \times \left(C + 600\,pF\right) \times \ln\left(\frac{R \times 0.8\,mA + 5.7\,V}{R \times 0.8\,mA + 30\,V}\right)$$

Table 11. Output Line Load Discharge Times

Input Discharge Time	Output Line Load
Discharge time < 40 µs	Very Light
40 µs < *discharge time* < 1,000 µs	Light
1,000 µs < *discharge time* < 10,000 µs	Medium
10,000 µs < *discharge time* < 100 ms	Heavy

Table 12. Output Line Load Recommendations

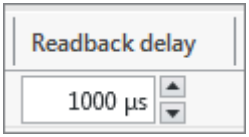
Configuration	External Load[1]	Cable Length/ Capacitance	Output Line Load
• Single output • Dual output • Single output with internal test pulse • Dual output with internal test pulse	High Impedance	≤10 m and ≤1.8 nF	Light
	High Impedance	≤50 m and ≤9 nF	Medium
	High Impedance	>50 m	Heavy
	≤3 kΩ	≤10 m and ≤1.8 nF	Very Light
	≤3 kΩ	≤50 m and ≤9 nF	Light
	≤3 kΩ	>50 m	Medium
• Single output with external test pulse • Dual output with external test pulse • Single output with external readback • Single input with test pulse • Dual input with test pulse	>3 kΩ	≤50 m and ≤9 nF	Light
	>3 kΩ	>50 m	Medium
	≤3 kΩ	≤50 m and ≤9 nF	Very Light
	≤3 kΩ	>50 m	Medium

[1] When the output load on the DO channel is a DI channel on the same module, load is >3 kΩ.

Readback Delay

The readback delay parameter sets the maximum time for a signal to propagate from the configured output channel to the reserved input channel. Setting this value too low could result in a false readback fault.

Figure 12. Readback Delay



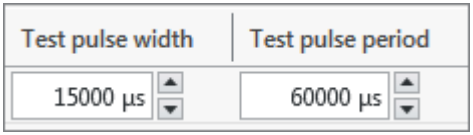
Related Information

Readback Diagnostics on page 82

Test Pulse Parameters

For channels configured with internal or external test pulses, you can configure the test pulse width and the test pulse period. For more information on configuring test pulses, refer to the *Test Pulses* section.

Figure 13. Test Pulse Parameters



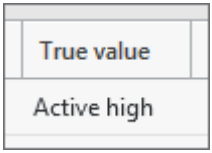
Related Information

Test Pulses on page 79

True Value

You can define the true value for input channels. The User Program will read the input signal as true when the channel returns the value configured by the parameter.

Figure 14. True Value



The options for true value are active high or active low.

Note Scan Interface reads the input signal, not the parameter in the User Program. If the input signal is high, Scan Interface will return a true value. If the input signal is low, Scan Interface will return a false value.

Analog I/O Parameters (NI 9351 Only)

When you select a configuration for a channel, the I/O Configuration table enables the appropriate parameters. Refer to the following table for the parameters associated with each configuration.

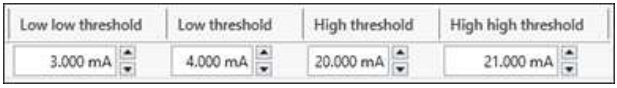
Table 13. I/O Configuration Parameters

Signal Type	Configuration	Parameters
Analog Inputs	Single input (1oo1)	Low low threshold, Low threshold, High threshold, High high threshold, Hysteresis
	Dual input (1oo2)	Low low threshold, Low threshold, High threshold, High high threshold, Hysteresis, Discrepancy time, Discrepancy current
	Triple input (2oo3)	Low low threshold, Low threshold, High threshold, High high threshold, Hysteresis, Discrepancy time, Discrepancy current

Thresholds

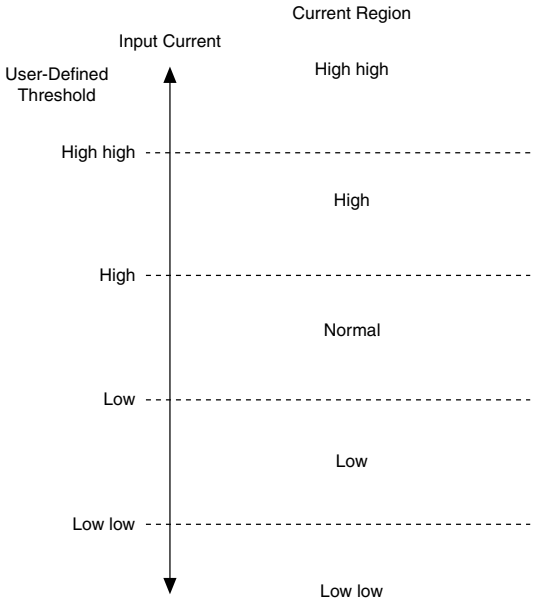
The Safety Editor allows you to set four current thresholds for each analog input configuration. The User Program applies the threshold values to every channel in the configuration.

Figure 15. Thresholds



Current thresholds define five regions that describe the state of the input. The module FPGA converts the current region for a channel into a Boolean value that can be read by the User Program. You can use the Boolean values for a current region as transition conditions in the state machine diagram.

Figure 16. Current Regions



Refer to the following guidelines when configuring thresholds:

- The four thresholds must be in a consecutive, increasing order. Low must have a larger value than low low, high must have a larger value than low, and high high must have a larger value than high.
- To ensure the module FPGA returns a low low region for a channel, set the low low threshold high enough to filter out inaccuracy and noise.
- The hysteresis ranges of two thresholds must not overlap. If you set the hysteresis range to 0.100 mA, the difference between any two thresholds must be >0.200 mA.
- If your system requires fewer than five configured current regions, you can conceal unnecessary regions, but the configured regions must be adjacent. For instance, you can configure low, normal, and high current regions, but not low, normal, and high high current regions.
- Set the high high and/or low low thresholds to their extreme values to conceal the outermost current regions.
- When concealing an outermost current region, use Boolean OR statements in transition conditions. For instance, if you don't want to use the high high current region, transitions that trigger on high should read `AIn.H or AIn.HH`.

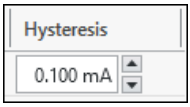
Related Information

Current Threshold Diagnostics on page 90
Analog Input Configurations (NI 9351 Only) on page 76

Hysteresis

You can set a hysteresis range that applies to all configured thresholds on an analog input configuration.

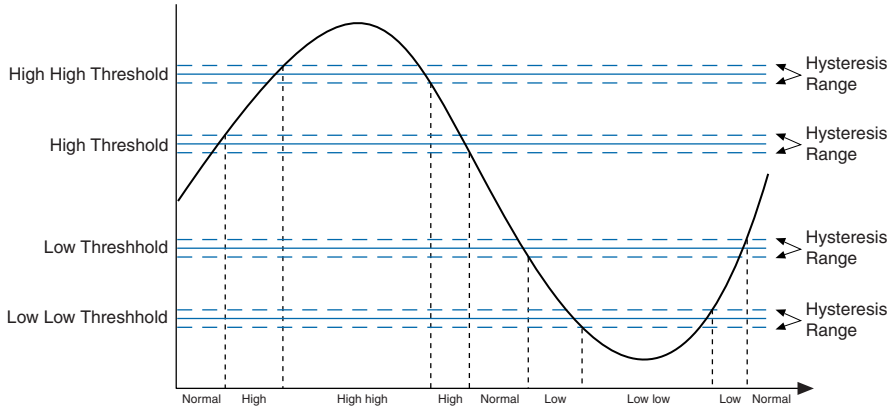
Figure 17. Hysteresis



Refer to the following guidelines when setting a hysteresis value for your analog configurations:

- The hysteresis range affects both the rising edge and the falling edge of the incoming signal. A hysteresis value of 0.100 mA filters the incoming signal between +0.100 mA and -0.100 mA of each configured threshold value.

Figure 18. Hysteresis Range for Input Signals



- The User Program applies the hysteresis value to all four thresholds. The hysteresis range of one threshold can not overlap the hysteresis range of another threshold.
- In dual input and triple input configurations, the hysteresis and threshold values apply to every channel in that configuration.

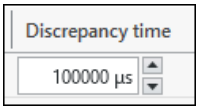
Related Information

Current Threshold Diagnostics on page 90

Discrepancy Time (Analog Configurations)

Discrepancy time sets the minimum time duration that a discrepancy current can exist between channels before a discrepancy warning or discrepancy fault is detected.

Figure 19. Discrepancy Time



Related Information

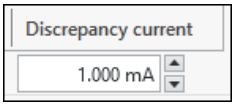
Discrepancy Faults for Analog Input Configurations on page 86

Discrepancy Warning on page 88

Discrepancy Current

You can define a discrepancy value for input currents on dual input and triple input configurations. If the input channels read currents that differ by more than the defined parameter after the discrepancy time has expired, the module FPGA will return a Discrepancy Fault or a Discrepancy Warning to the User Program, based on your configuration.

Figure 20. Discrepancy Current



Tip To maximize the effectiveness of discrepancy current detection, set the discrepancy current parameter as low as the system will allow. In most systems, the discrepancy current should be significantly less than the normal current range (*high threshold - low threshold*).

Note You must set the discrepancy current greater than 0 mA.

Related Information

Discrepancy Faults for Analog Input Configurations on page 86

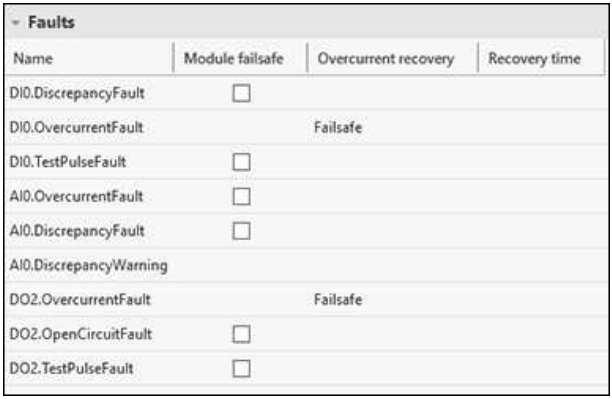
Discrepancy Warning on page 88

Analog Input Configurations (NI 9351 Only) on page 76

Setting Faults to Trigger Fail-Safe Mode

The **Faults** table populates based on the channel configurations you select.

Figure 21. Faults Table



If you check the **Module failsafe** box next to a fault, that fault will trigger the module to go into Fail-safe Mode. Checking the box also reserves that signal so it cannot be used in the state machine diagram. If you leave the box unchecked, you can use that signal as an input to trigger transitions in the state machine diagram.

To set **Module failsafe** for overcurrent faults on digital input and digital output configurations, select **Failsafe** from pull-down menu in the **Overcurrent recovery** column.

Caution All fault signals listed in the **Faults** table must have the **Module failsafe** box checked or must be used as transition conditions in the state machine diagram. If a fault occurs and that fault signal is not configured, the fault will not be handled by the User Program.

Tip You can copy and paste the fault name to avoid retyping it in the state machine diagram. Click on the fault name and press <Ctrl-C> to copy the fault name in the I/O Configuration table. If you're using the Functional Safety Editor 2018 or later, output signal values and transition conditions have a predictive text feature that allows you to choose from a list of available faults.

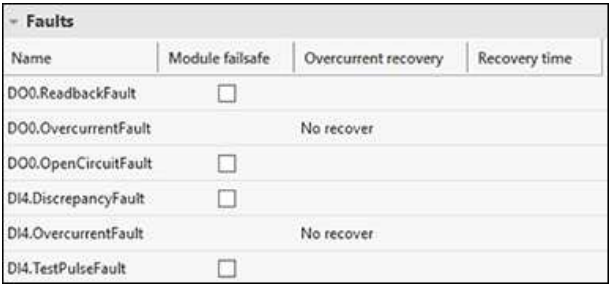
Related Information

Fail-Safe Mode on page 14
Diagnostics on page 71
Fault Response Time on page 61

Overcurrent Recovery (Digital Configurations)

When an overcurrent condition occurs on a digital channel, the channel de-energizes and the User Program returns an overcurrent fault. Configuring a digital input with test pulses or output populates the **Faults** table with an overcurrent fault for that channel.

Figure 22. Overcurrent Faults in the Faults Table



Overcurrent faults include an **Overcurrent recovery** pull-down menu that allows you to configure how the module responds when that channel reads an overcurrent condition.

- **Failsafe**—The module goes into Fail-safe Mode until you cycle external V_{sup} power to the module. This selection functions in the same way as checking the module failsafe box for other fault signals.

Figure 23. Overcurrent Fault Set to Failsafe

- **Auto recover**—The channel de-energizes. After the **Recovery time** elapses, the fault will clear, allowing the user program to energize the output again. If the current remains in an overcurrent state, the channel will de-energize again. The de-energize and auto recover cycle will continue until the module no longer reads an overcurrent condition.

Figure 24. Overcurrent Fault Set to Autorecover

- **No recover**—The channel de-energizes and remains de-energized until you cycle external V_{sup} power to the module. You can use the fault as a transition condition in the state machine diagram.

Figure 25. Overcurrent Fault Set to No Recover

Related Information

Overcurrent Diagnostics (Digital) on page 82

Configuring I/O Channels

1. Open the I/O Configuration table.
2. Select the appropriate channel in the digital inputs, digital outputs, or analog inputs table.
3. Click the channel name in the **Name** column to rename the channel, if necessary.

   Note You must use the channel name set in the I/O Configuration table when programming output values and transitions.
4. Click the cell in the **Configuration** column to select the configuration type for that channel.
5. Update the I/O parameters, as necessary.
6. Repeat steps 2 through 5 for all connected channels.
7. Verify that you have done the following:
   - Set a default value for all configured digital outputs.
   - Selected **Module failsafe** for applicable fault diagnostics.
   - Set a default value for all variables.

Related Information

I/O Configuration Table on page 18

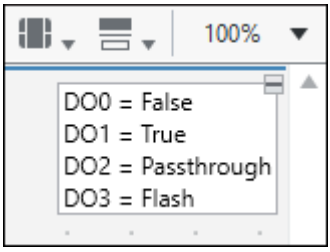
State Machine Diagram

Default Signal Values

You must set the default signal value for every output and variable you configure in the I/O Configuration table. When you use an output or variable in a state machine the default value appears in the **Default signal values** field on the **Module and Diagram** tab of the configuration pane for that state machine. The default values will apply when the User Program commences execution. If output values are not defined by the current state, the default value for that output will apply.

Default signal values appear in a pane in the upper right-hand corner of the state machine diagram.

Figure 26. Default Signal Values in State Machine Diagram

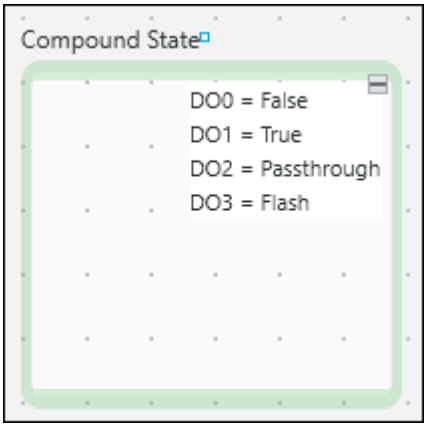


You can define default signal values for compound states by editing the **Signal values** field on the **Item** tab of the configuration pane. These default values will apply when the User Program

transitions into that compound state. If output values are not defined by the current simple state, the default value for that output will apply.

Default signal values for compound states appear in a pane in the upper right-hand corner of the compound state.

Figure 27. Signal Values for Compound States in the State Machine Diagram



Tip You can shrink or expand the default signal value pane by clicking the small square at the top of the pane.

Related Information

Compound States on page 40

Passthrough

Setting the digital output value to passthrough allows you to write directly to digital output channels through Scan Interface.

Use the following syntax to configure a digital output channel for passthrough:

<channel name> = passthrough, where <channel name> is the name of the digital output channel defined in the I/O Configuration table.

When communication to the controller is lost, the output value of the passthrough channel will be set to False. Once communication is restored, Scan Interface will be able to write the output value to the passthrough channel.

Note Digital passthrough may behave differently depending on which version of the firmware you have installed. For more information about differences in firmware versions, refer to the *C Series Functional Safety Firmware Versions* section.

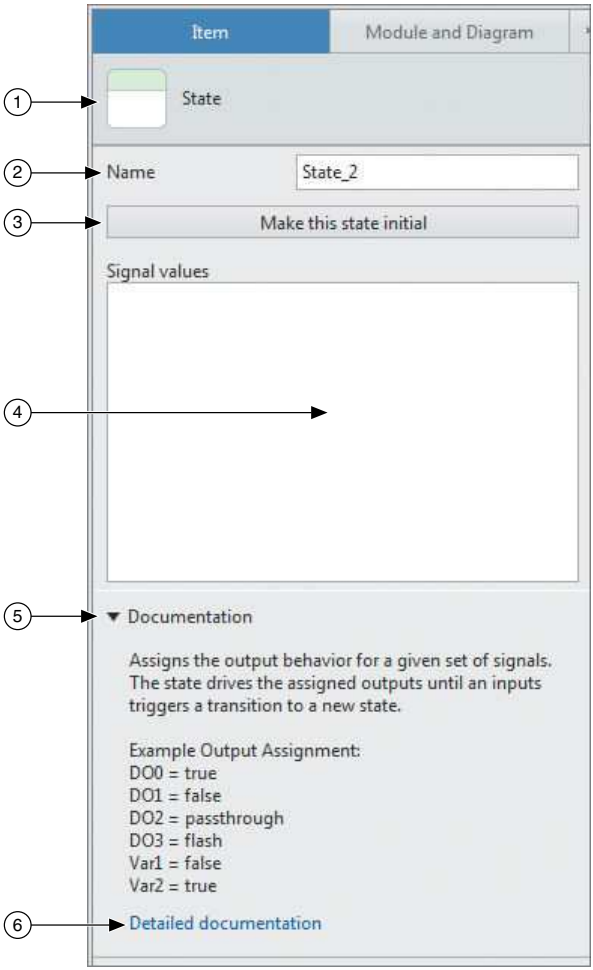
Caution The digital passthrough bypasses the User Program and should not be used for safety-critical outputs.

Tip Consider using passthrough during proof tests or when validating your system.

States

States represent a set of driven outputs that run until specified inputs trigger a transition. A single state machine supports up to 32 states. Drag and drop states from the palette in the state machine diagram and modify states in the diagram or on the **Item** tab of the Configuration pane.

Figure 28. State Item Tab



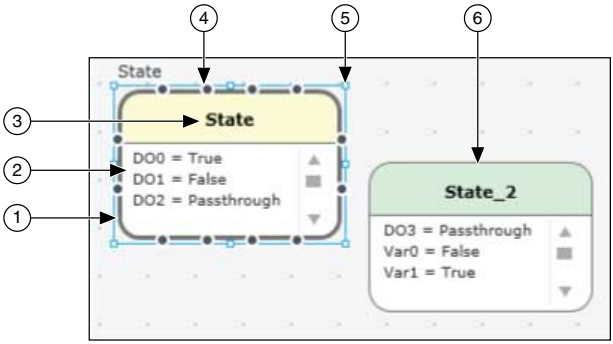
1. **State icon**—The icon and label indicate whether the state is simple or compound.
2. **State name**—This field allows you to rename the state.

3. **Make this state initial button**—This button allows you to set any intermediate state as the initial state for that state machine or compound state. Compound states can also be set as the initial state for a state machine.
4. **Signal states field**—This field contains the signal values for simple states or the default signal values for compound states.
5. **Documentation**—The documentation section provides helpful information about states.
6. **Detailed documentation link**—This link connects to the C Series Functional Safety Manual on *ni.com/manuals*.

Simple States

Simple states drive a specified list of outputs that run in response to system inputs.

Figure 29. Simple State Elements



1. **Initial state**—An initial state sets the signal values for the User Program or compound state when execution commences. All other states are intermediate states. Initial states are yellow and have thick gray borders.
2. **State output field**—This field displays the output values for a given simple state. You can type the output values directly into the field.
3. **State name field**—This field displays the state name. You can rename the state by clicking directly on the field.
4. **Terminal**—Terminals allow you to connect transitions between states. Each simple state has twelve terminals.
5. **Resize handle**—Resize handles allow you to increase or decrease the size of the state.
6. **Intermediate state**—An intermediate state is any simple state that is not an initial state. Intermediate states are green with a thin gray border.

Note To change an intermediate state to an initial state, right-click the state and select **Make this state initial**. You can also select **Make this state initial** on the **Item** tab of the configuration pane.

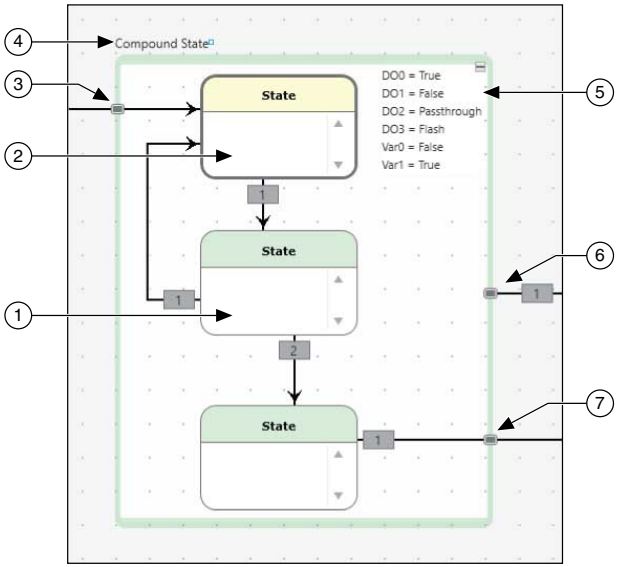
Related Information

Output Signal Value Syntax on page 41
Adding States on page 42

Compound States

Compound states are sub-state machines that contain simple states and transitions. Compound states can nest within other compound states.

Figure 30. Compound State Elements



1. **Intermediate state**—Intermediate states can serve as the destination for transitions from states inside or outside of the compound state.
2. **Initial state**—Transitions to terminals on the border of compound states will trigger the initial state.
3. **Terminal**—Terminals can connect external transitions to the border of the compound state. They can also act as tunnels to connect transitions with simple states inside the compound state. To create compound state terminals:
   - Double-click the edge of the compound state.
   - Connect a transition to the edge of a compound state.
   - Connect a transition to simple state within the compound state.
4. **Compound state name**—This field displays the name of the compound state. You can rename the compound state by clicking directly on the field.
5. **Default signal values**—This field displays the default signal values for the compound state. You can expand or collapse the field by clicking the box in the upper right corner.

6. **Transition from compound state**—Transition conditions can trigger transitions from the borders of compound state. If the statement evaluates as true, the User Program will transition out of the compound state regardless of the current simple state.
7. **Transition from simple state**—Transition conditions can trigger transitions from simple states within the compound state. If the statement evaluates as true, the User Program will transition out of the compound state.

Related Information

Output Signal Value Syntax on page 41
Adding States on page 42
Default Signal Values on page 36

Output Signal Value Syntax

States require Boolean statements to set output signal values. Statements include the channel or variable name and a keyword that defines the signal value.

Follow these guidelines when writing output signal values:

- You must use the channel name or variable name defined in the **Name** column of I/O Configuration table. Do not use the name defined in the **Hardware name** column.
- Keywords are not case-sensitive.
- Only one state machine can write to a given output channel or variable.
- The User Program resolves the innermost state for a given output or variable.

Table 14. Output Signal Value Syntax

Type	Syntax	Keywords	Notes
Output channel	<channel name> = <keyword>	`True`	Energizes channel
		`False`	De-energizes channel
		`Flash`	Output toggles at user-configurable interval
		`Passthrough`	Allows monitoring VI in LabVIEW to set output value
Variable	<variable name> = <keyword>	`True`	Sets variable value to true
		`False`	Sets variable value to false

Table 14. Output Signal Value Syntax (Continued)

Type	Syntax	Keywords	Notes
UserLED0	`UserLED0` = <keyword>	`True`	Sets LED on
		`False`	Sets LED off
		`Flash`	Sets LED flashing behavior

Related Information

Simple States on page 39
Compound States on page 40
Adding States on page 42

Adding States

Follow these steps to add simple states to the state machine diagram.

1. Select the state on the palette.
2. Drag the state from the palette and drop it onto the state machine diagram.
3. Update the state name in the state name field on the state or in the **Name** field on the **Item** tab of the configuration pane.
4. Configure output signal values for the state using the text field on the state or in the **Signal values** field on the **Item** tab of the configuration pane.

Note You must use the channel name or variable name defined in the **Name** column of I/O Configuration table. Do not use the name defined in the **Hardware name** column.

Tip In the Functional Safety Editor 2018 or later, output signal values have a predictive text feature. You can start typing or push <Ctrl-Space> to display a menu of possible channel names or output signal values based on how you configured the I/O configuration table.

Related Information

Simple States on page 39
Compound States on page 40
Output Signal Value Syntax on page 41

Naming States in the State Machine Diagram

Follow these guidelines when naming states in the state machine diagram:

- Use only Unicode 5.0 language-type characters.
- Do not use Boolean operators as names.
- Do not use numbers.

- Do not start the state name with a space or an underscore.
- Refer to the following table for a list of common keywords and operators that are not allowed for use as state names.

Note The Functional Safety Editor will not allow you to enter forbidden characters.

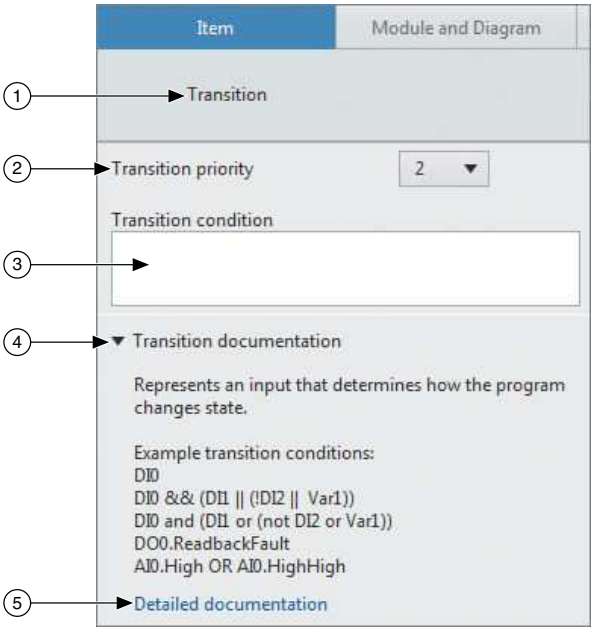
Table 15. Forbidden Keywords and Operators

Keywords		Operators		
after	or	||	+	!
true	and	&&	*	.
false	not	^^	( )	=

Transitions

Transitions determine how the User Program changes state. You can configure inputs, variables, and faults in the I/O Configuration table and use them as transition conditions. Transition conditions support most Boolean operators and timing statements.

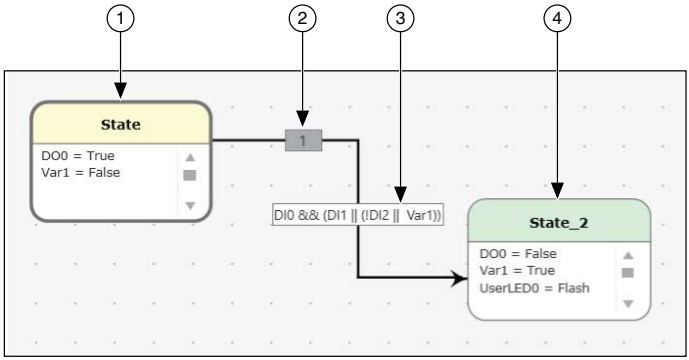
Figure 31. Transition Item Tab



1. **Transition icon**—The icon and label indicate that a transition is selected.
2. **Transition priority pull-down menu**—This menu allows you to set the priority number for the selected transition.

3. **Transition condition field**—This field contains the statement that triggers the selected transition.
4. **Documentation**—The documentation section provides helpful information about transitions.
5. **Detailed documentation link**—This link connects to the C Series Functional Safety Manual on *ni.com/manuals*.

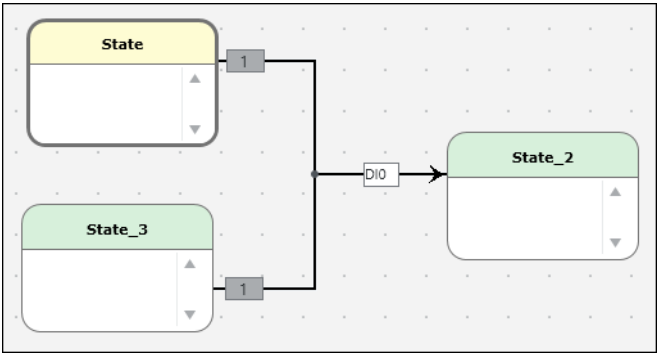
Figure 32. Transitions in the State Machine Diagram



1. Source state
2. Priority number
3. Transition condition
4. Destination state

Tip You can join transitions from two or more source states. The priority numbers apply to the individual source states. The transition condition will apply to all joined transitions.

Figure 33. Joined Transitions



Related Information

Transition Statements on page 45
Wiring Transitions on page 47

Transition Statements

The User Program requires transition statements to trigger the User Program to change states. Transition statements include the channel or variable name and a keyword that defines the condition that triggers the transition.

Follow these guidelines when writing transition statements:

- Type logic statements directly into the field on the transition in the state machine diagram or into the field on the configuration pane.
- You must use the channel or variable name set in the I/O Configuration table.
- Keywords and Boolean operators are not case-sensitive.
- Do not use = in transition statements.

Table 16. Digital Transition Statements

Type	Syntax	Keywords	Example
Input	<channel name>.<keyword>	`Discrepancyfault`	`DI2.DiscrepancyFault`
		`Overcurrentfault`	
		`Testpulsefault`	
Output	<channel name>.<keyword>	`Readbackfault`	`DO1.OpenCircuitFault`
		`Overcurrentfault`	
		`Opencircuitfault`	
		`Testpulsefault`	

Table 17. Analog Transition Statements (NI 9351 Only)

Type	Syntax	Keywords	Example
Input	<channel name>.<keyword>	`Overcurrentfault`	`AI0.DiscrepancyWarning`
		`Discrepancyfault`	
		`Discrepancywarning`	
		`Highhigh`, `HH`	`AI2.LowLow`, `AI2.LL`
		`High`, `H`	
		`Normal`, `N`	
		`Low`, `L`	
		`Lowlow`, `LL`	

Table 18. Boolean Logic

Type	Syntax	Keywords	Example
Boolean operators	<keyword><channel or variable name> <keyword><transition statement>	Not: `not`, `!`	`not DI2`, `!AI0.High`
	<channel or variable name> <keyword> <channel or variable name> <keyword><transition statement>	Or: `or`, `+`, `||`	`DI2 or Var2`, `AI0.High or AI0.HighHigh`
		And: `and`, `*`, `&&`	`DI2 and Var2`, `AI2.High && AI0.DiscrepancyWarning`

Related Information

Transitions on page 43
Wiring Transitions on page 47
Diagnostics on page 71

Timing Transitions

- Each state machine supports up to six timers.
- Timing transitions read as true after a specified time elapses.
- Maximum transition time is 4,096 hours.
- Minimum transition time is constrained by the maximum application processing time: 60 µs.

Table 19. Timer Accuracy

Timer Value	Timer Accuracy	
	Minimum	Maximum
Timer value ≤ 4,096 µs	*Timer value* + 30 µs	*Timer value* + 60 µs
4,096 µs < *timer value* ≤ 4,096 ms		*Timer value* + 1.06 ms
4,096 ms < *timer value* ≤ 4,096 s		*Timer value* + 1 s
4,096 s < *timer value* ≤ 4,096 hours		*Timer value* + 1 hour

Table 20. Timing Logic

Type	Syntax	Keywords
Timing Transition	After x <keyword>	`us`, `µs`, `ms`, `s`, `min`, `mins`, `hr`, `hrs`, `day`, `days`

Transition Priority

The User Program samples all inputs simultaneously. Transitions from a state are evaluated according to transition priority number. The priority number appears on the transition wire next to the transition condition and on the **Item** tab of the configuration pane. Default transition priority is determined by the order transitions are wired. To update the transition order, select the Transition Priority pull-down menu on the **Item** tab of the configuration pane.

The User Program evaluates the statement from transition 1 of the current state. If transition 1 evaluates as false, the User Program then evaluates the statement from transition 2. This continues until all transitions have been evaluated or until a transition evaluates as true. If no transition evaluates as true, the User Program remains in the current state for the next application processing loop. If a transition evaluates as true, the User Program updates to the configured state.

Transitions from nested compound states are evaluated from the outermost state to the innermost state. The User Program first evaluates the transitions originating at the border of outermost compound state, in priority order. If none of the transitions evaluate as true, the User Program evaluates the transitions exiting the next nested compound state. This continues until all transitions in the nested compound states have been evaluated. If no transitions from compound states evaluate as true, the User Program evaluates the transitions from the current simple state.

User Programs evaluate transitions in the following order:

1. Transitions from outermost compound state in transition priority order
2. Transitions from each nested compound state in transition priority order
3. Transitions from current state in transition priority order

Wiring Transitions

Follow these steps to wire transitions in the Functional Safety Editor.

1. Initiate the transition from the source simple state or compound state.
   - To create transitions from a simple state, click the terminal on the state border.
   - To create transitions from a compound state, double-click the border of a compound state to add a state machine tunnel. Then click the terminal on the state machine tunnel.

   Tip Simple states have twelve terminals for connecting transition. If you need more than twelve connections to or from a single state, consider using a compound state.
2. Complete the transition on the destination simple state or compound state.
   - To complete a transition to a simple state, click the terminal on the state border.
   - To complete a transition to a compound state, click the terminal on the state machine tunnel.

   Note Every transition must have at least one source state and only one destination state.

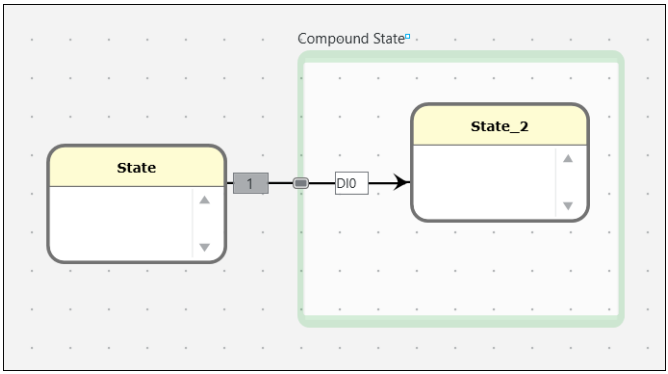
   Note Transition direction is dependent on wiring order. Make sure you click the source state first and the destination state second.

3. Right-click the transition and select **Create transition condition**.

   You can also click on the **Transition condition** field on the **Item** tab of the configuration pane.

Note When a transition wire crosses the border of a compound state, a state machine tunnel creates two wire segments. The Functional Safety Editor only allows transition statements on the wire segment closest to the target state.

Figure 34. Transition Crossing a Compound State Border



4. Type the transition statement into the transition condition box on the wire or the **Transition condition** field on the configuration pane. Always follow transition syntax rules.

Note You must use the channel name or variable name defined in the **Name** column of I/O Configuration table. Do not use the name defined in the **Hardware name** column.

Tip In the Functional Safety Editor 2018 or later, transition conditions have a predictive text feature. You can start typing or push <Ctrl-Space> to display a menu of possible channel names and transition statements based on how you configured the I/O configuration table.

Related Information

Transitions on page 43
Transition Statements on page 45

Saving and Compiling

User Programs save as `.fsp` files from the Functional Safety Editor. When you click the **Compile** button, the editor and compiler generate multiple files. All the generated files will share the filename assigned to the `.fsp` file.

- <*filename*>`.json`—Human-readable file that inputs to the compiler. Review this file for correctness.
- <*filename*>`.bin`—Compiled User Program that is generated by the compiler from the JSON. Download this file to the module from the LabVIEW project.
- <*filename*>`_errors.json`—Lists any errors generated by the compiler. This file populates the **Errors and Warnings** pane. This file is also human-readable and you can review this file for errors.
- <*filename*>`_report.log`—Lists additional information about the compiled User Program. You can review this file to verify the channel and fault configurations in your User Program.

You must download the binary User Program file to the C Series Functional Safety module from the properties dialogue of a target module in a project in LabVIEW.

Related Information

Compiling User Programs on page 9
Downloading User Programs on page 11

JSON Files

You can use the JSON file output from the editor to verify your User Program before downloading it to the module.

- Refer to the following tables for definitions and parameters when reviewing the JSON.
- Strings (keys and values) are not case sensitive.
- Any JSON element in the file that is not enumerated below will be ignored. If you manually write the JSON file and misspell a key, the key will be ignored as a missing element.

Type Definitions

- **Identifier string**—A valid UTF-8 Unicode sequence. Each identifier starts with a letter (Unicode general category L), followed by zero or more letters (Unicode general category L), numbers (Unicode general category N), marks (Unicode general category M), and punctuation connectors (Unicode general category Pc).
- **Time string**—A valid UTF-8 Unicode sequence that matches the following regular expression: `\s*\d+\s*[us|µs|ms|s|min|hr|day]\s*`, where `\s` means any spacing character (Unicode general category Z, and the vertical tab, newline, horizontal tab, form feed, and carriage return characters) and `\d` means any decimal digit (Unicode general category Nd).
- **High/Low string**—A string that is either `high` or `low`.

- **Boolean string**—A string that is either `true` or `false`.
- **OutputDrive string**—A string that is either `true`, `false`, `passthrough`, or `flash`.

JSON Definitions

Table 21. JSON Object Parameters

Term	Required?	Type	Notes
`version`	Yes	Numeric	Must be `1` or `2` if the `module_type` is `9350`. Must be `2` if the `module_type` is `9351`
`module_type`	Yes	String	Must be `9350` or `9351`
`program_info`	Yes	String	Specifies information and settings specific to the program

Table 22. Program Info Object Parameters

Term	Required?	Type	Notes
`build_number`	Yes	Numeric	Increments when you first edit a User Program that has successfully compiled
`guid`	Yes	GUID string	Formatted with hyphens and braces
`auto_start`	Yes	Boolean string	Specifies if the program should begin automatically when the module powers on
`led_flash_rate`	Yes	Time string	Specifies the rate UserLED0 blinks when set to Flash, must be between 50 ms and 12.8 s, inclusive, and be a multiple of 50 ms
`fault_latch_time`	Yes	Time string	—

Table 23. Module Type Object Parameters

Module_type	Term	Required?	Type	Number of Elements	Definition
`9350`	`digital_inputs`	Yes	Array	8	Digital input objects
	`digital_outputs`	Yes	Array	8	Digital output objects
	`variables`	Yes	Array	24	Variable objects
	`state_machines`	Yes	Array	0 to 7 (inclusive)	State tree node objects

Table 23. Module Type Object Parameters (Continued)

Module_type	Term	Required?	Type	Number of Elements	Definition
9351	digital_inputs	Yes	Array	4	Digital input objects
	digital_outputs	Yes	Array	4	Digital output objects
	analog_inputs	Yes	Array	4	Analog input objects
	variables	Yes	Array	24	Variable objects
	state_machines	Yes	Array	0 to 3 (inclusive)	State tree node objects

Table 24. Digital Input Object Parameters

Term	Required?	Type	Notes
alias_name	Yes	Identifier string	—
config_type	Yes	String	Must be one of: reserved, unused, basic, test_pulse, dual, or dual_and_test_pulse
discrepancy_time	Required if config_type is dual or dual_and_test_pulse	Time string	—
complementary	Required if config_type is dual or dual_and_test_pulse	Boolean string	—
owner	Required if config_type is reserved	Identifier string	—
test_pulse_width	Required if config_type is test_pulse or dual_and_test_pulse	Time string	—

Table 24. Digital Input Object Parameters (Continued)

Term	Required?	Type	Notes
test_pulse_period	Required if config_type is test_pulse or dual_and_test_pulse	Time string	—
true_value	Required if config_type is basic, test_pulse, dual, or dual_and_test_pulse	String	Must be high or low
debounce_filter_time	Required if config_type is basic, test_pulse, dual, or dual_and_test_pulse	Time string	—
load	Required if config_type is test_pulse, dual_and_test_pulse	Time string	Must have a value of 40 µs, 1 ms, 10 ms, or 100 ms (or equivalent).
fail_safe_on_over_current	Required if config_type is test_pulse or dual_and_test_pulse	Boolean string	Can not be true if auto_recover_on_over_current is true.If both failsafe_on_over_current and auto_recover_on_over_current are false, the module will not attempt to recover from the fault.
fail_safe_on_discrepancy	Required if config_type is dual or dual_and_test_pulse	Boolean string	—
fail_safe_on_test_pulse	Required if config_type is test_pulse or dual_and_test_pulse	Boolean string	—

Table 24. Digital Input Object Parameters (Continued)

Term	Required?	Type	Notes
auto_recover_on_over_current	Required if failsafe_on_over_current is false	Boolean string	Can not be true if failsafe_on_over_current is true. If both failsafe_on_over_current and auto_recover_on_over_current are false, the module will not attempt to recover from the fault.
over_current_auto_recover_time	Required if auto_recover_on_over_current is true	Time string	Must have a value between 3 s and 16384 s.

Table 25. Digital Output Object Parameters

Term	Required?	Type	Notes
alias_name	Yes	Identifier string	—
config_type	Yes	String	Must be reserved, unused, basic, test_pulse_internal, test_pulse_external, external_readback, dual, or dual_and_test_pulse_internal, dual_and_test_pulse_external
owner	Required if config_type is reserved	Identifier string	—

Table 25. Digital Output Object Parameters (Continued)

Term	Required?	Type	Notes
`test_pulse_width`	Required if `config_type` is `test_pulse_internal`, `test_pulse_external`, `dual_and_test_pulse_internal`, or `dual_and_test_pulse_external`	Time string	—
`test_pulse_period`	Required if `config_type` is `test_pulse_internal`, `test_pulse_external`, `dual_and_test_pulse_internal`, or `dual_and_test_pulse_external`	Time string	—
`readback_delay`	Required if `config_type` is `external_readback`	Time string	—
`load`	Required if `config_type` is `basic`, `test_pulse_internal`, `test_pulse_external`, `dual`, `dual_and_test_pulse_internal`, `dual_and_test_pulse_external`, or `external_readback`	Time string	Must have a value of 40 µs, 1 ms, 10 ms, or 100 ms (or equivalent).
`flash_rate`	Required if `config_type` is `basic`, `test_pulse_internal`, `test_pulse_external`, `dual`, `dual_and_test_pulse_internal`, `dual_and_test_pulse_external`, or `external_readback`	Time string	Must have a value that is a multiple of 50 ms, between 50 ms and 12.8 s, inclusive.

Table 25. Digital Output Object Parameters (Continued)

Term	Required?	Type	Notes
debounce_filter_time	Required if config_type is basic, test_pulse_external, external_readback, or dual_and_test_pulse_external	Time string	—
fail_safe_on_over_current	Required if config_type is basic, test_pulse_internal, test_pulse_external, dual, dual_and_test_pulse_internal, dual_and_test_pulse_external, or external_readback	Boolean string	Can not be true if auto_recover_on_over_current is true. If both failsafe_on_over_current and auto_recover_on_over_current are false, the module will not attempt to recover from the fault.
fail_safe_on_open_circuit	Required if config_type is basic, test_pulse_internal, test_pulse_external, dual, dual_and_test_pulse_internal, dual_and_test_pulse_external, or external_readback	Boolean string	
fail_safe_on_discrepancy	Required if config_type is dual, dual_and_test_pulse_internal, or dual_and_test_pulse_external	Boolean string	—
fail_safe_on_test_pulse	Required if config_type is test_pulse_internal, test_pulse_external, dual_and_test_pulse_internal, or dual_and_test_pulse_external	Boolean string	—

Table 25. Digital Output Object Parameters (Continued)

Term	Required?	Type	Notes
fail_safe_on_readback	Required if config_type is basic, external_readback, dual	Boolean string	—
auto_recover_on_over_current	Required if failsafe_on_over_current is false	Boolean string	Can not be true if failsafe_on_over_current is true. If both failsafe_on_over_current and auto_recover_on_over_current are false, the module will not attempt to recover from the fault.
over_current_auto_recover_time	Required if auto_recover_on_over_current is true	Time string	Must have a value between 3 s and 16384 s.

Table 26. Analog Input Object Parameters

Term	Required?	Type	Notes
alias_name	Yes	String	—
config_type	Yes	String	Must be single1oo1, dual1oo2, triple2oo3, reserved, or unused
threshold_lowlow	Required if config_type is analog_input	Numeric	Threshold expressed in nA
threshold_low	Required if config_type is analog_input	Numeric	Threshold expressed in nA
threshold_high	Required if config_type is analog_input	Numeric	Threshold expressed in nA

Table 26. Analog Input Object Parameters (Continued)

Term	Required?	Type	Notes
threshold_highhigh	Required if config_type is analog_input	Numeric	Threshold expressed in nA
discrepancy_time	Required if config_type is dual1oo2 or triple2oo3	Time string	Must have a value between 1000 µs and 1800000000 µs.
discrepancy_current	Required if config_type is dual1oo2 or triple2oo3	Numeric	The difference between channels required to trigger a discrepancy fault in nA. Value must be greater than 0.
hysteresis	Required if config_type is single1oo1, dual1oo2, or triple2oo3	Numeric	The value by which the analog input must surpass a threshold (above or below) for the channel to be considered in the next region in nA.
fail_safe_on_over_current	Required if config_type is single1oo1, dual1oo2, or triple2oo3	Boolean string	—
fail_safe_on_discrepancy	Required if config_type is dual1oo2 or triple2oo3	Boolean string	—

Table 27. Variable Object Parameters

Term	Required?	Type	Notes
alias_name	Yes	Identifier string	—
config_type	Yes	String	Must be internal or unused

Table 28. State Machine Object Parameters

Term	Required?	Type	Notes
`name`	Yes	String	—
`actions`	Yes, but may be empty	Array	Action objects
`transitions`	Yes, but may be empty	Array	Transition objects
`substates`	Yes, but may be empty	Array	State tree node objects
`initial_state`	Optional	Identifier string	—

Table 29. Action Object Parameters

Term	Required?	Type	Notes
`signal`	Yes	Identifier string	—
`value`	Yes	Output drive string	—

Table 30. Transition Object Parameters

Term	Required?	Type	Notes
`expression`	Yes	Expression string	—
`next_state`	Yes	Path string	—

Semantic Definitions

JSON statements must adhere to the following criteria.

- Every signal, state, and state machine must have a unique name.
- Every top-level state machine must have an initial state, and that initial state must be resolvable.
- The `next_state` parameter of every transition must refer to a resolvable state.
- No top-level state machine may have more than 32 states.
- The `signal` parameter of every `action` object must have the name of a digital output or writable variable.
- The DI and DO channels are dependent. Refer to the following table for the configuration required for each channel pair.

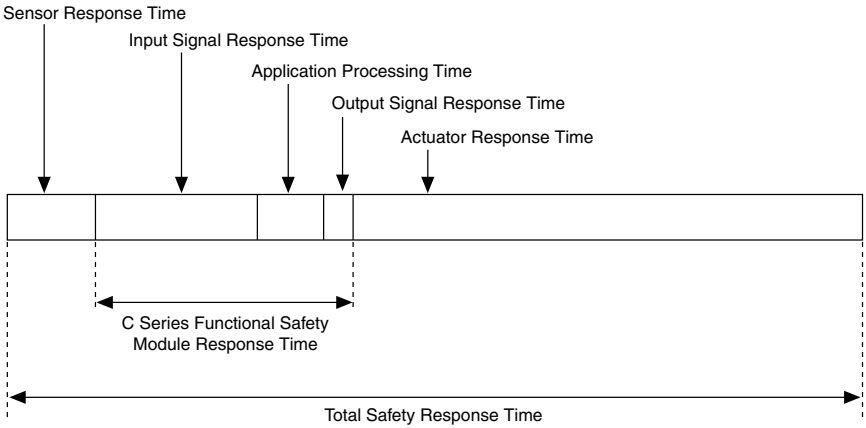
Table 31. Digital Channel Pair Configurations

DIn	DIn+1	DOn	DOn+1
Unused or Basic	Unused or Basic	Unused or Basic	Unused or Basic
Unused or Basic	Reserved	Unused or Basic	External Readback or Test Pulse
Reserved	Unused or Basic	External Readback or Test Pulse	Unused or Basic
Reserved	Reserved	External Readback or Test Pulse	External Readback or Test Pulse
Test Pulse	Unused or Basic	Reserved	Unused or Basic
Test Pulse	Reserved	Reserved	External Readback or Test Pulse
Test Pulse	Test Pulse	Reserved	Reserved
Dual	Reserved	Unused or Basic	Unused or Basic
Dual Test Pulse	Reserved	Reserved	Reserved
Reserved	Reserved	Dual Test Pulse	Reserved
Unused or Basic	Unused or Basic	Dual	Reserved
Dual	Reserved	Dual	Reserved

Safety System Response Time

The total safety response time of a C Series Functional Safety module is the delay from the sensor detecting an unsafe condition to the actuator implementing a control to return the system to a safe state.

Figure 35. Safety Response Time



Note You must verify the system safety response time before deploying the safety system.

Related Information

Safety Response Time Specifications on page 69

Calculating Safety System Response Times

Use the following equation to calculate the total safety response time for your C Series Functional Safety system. Optionally, you can use the times in the *C Series Functional Safety Response Time Specifications* section.

Total safety response time = Sensor response time + Input signal response time + Application processing time + Output signal response time + Actuator response time

1. Calculate the sensor response time. Refer to the *Sensor Response Time* section.
2. Calculate the input response time. Refer to the *Input Signal Response Time* section.
3. Calculate the diagnostic response time, if applicable. Refer to the *Diagnostic Response Time Specifications* section.
4. Determine the application processing time. This is a guaranteed minimum of 60 µs for each state in the defined execution path of the User Program. Refer to the *Application Processing Time* section.
5. Add the output signal response time from the *Response Time Specifications* section.
6. Calculate the actuator response time. Refer to the *Actuator Response Time* section.

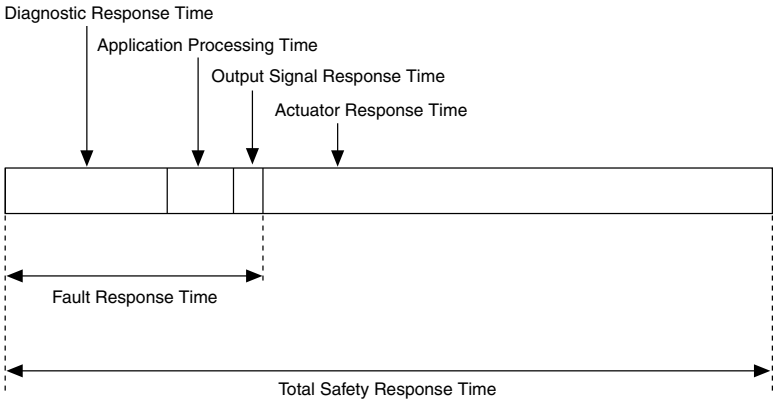
Fault Response Time

Fault diagnostics run parallel to and independent of the application processes. Faults used as transition conditions in the state machine diagram require the time necessary for fault processing and application processing to set an output response.

- User-configurable diagnostics used as transition conditions: *Fault response time = Diagnostic response time + Application processing time + Output signal response time*

Fault response time will vary depending on the execution path defined in the User Program. If the fault triggers a state that sets an output, only one iteration of the application processing time will elapse. Otherwise, the application processing time will increase by the number of states and associated transitions through which the User Program cycles to fully react to the fault.

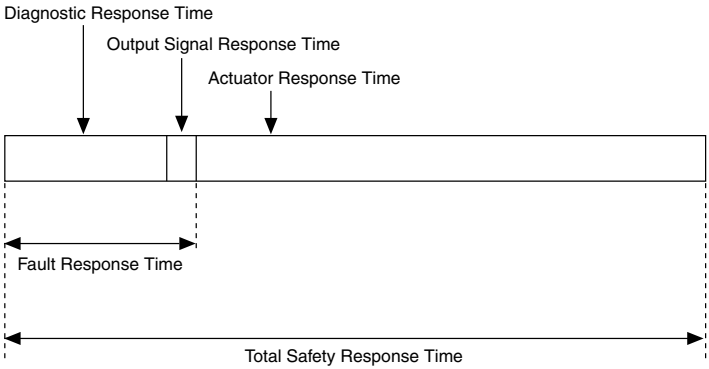
Figure 36. Response Time for Faults Used in Transitions



Faults configured to trigger Fail-safe Mode and automatic self-diagnostics do not require the application processing time. After the fault response time elapses, the module transitions into Fail-safe Mode.

- Automatic self-diagnostics: *Fault response time = Diagnostic response time + Output signal response time*
- User-configurable diagnostics configured to trigger Fail-safe Mode: *Fault response time = Diagnostic response time + Output signal response time*

Figure 37. Response Time to Trigger Fail-Safe Mode and Automatic Self-Diagnostics



Refer to the *Diagnostic Response Time Specifications* to determine the diagnostic response time for your application.

Related Information

Setting Faults to Trigger Fail-Safe Mode on page 33

Sensor Response Time

Sensor Response Time is the total time it takes for sensors to detect an unsafe situation and propagate the input signal to the C Series Functional Safety module. Sensor response times are dependent on the peripheral devices, the safety application, resistors, the length of the cabling, and the total load on the cabling. Refer to the third-party device specifications to calculate sensor response times.

Note Make sure you include debounce time for the sensors used in your system. Refer to the third-party device specifications for debounce time.

Digital Input Signal Response Time

Input signal response time describes the total time it takes to charge or discharge line load capacitance, propagate the signal to the User Program, and pass the signal through the debounce filter.

Use the following equations to calculate input signal response times.

(1 to 0) Input signal response time:

Input signal response time = *Input discharge time* + *Debounce filter time* + *Debounce constant* + 15 µs

(0 to 1) Single Input:

Input signal response time = *Debounce filter time* + *Debounce constant* + 18 µs

(0 to 1) Single Input with test pulse:

Input signal response time = (2 × *Debounce filter time*) + (3 × *Debounce constant*) + *Test pulse width* + 39 µs

(0 to 1) Dual Input:

Input signal response time = *Debounce filter time* + *Debounce constant* + *Discrepancy time* + *Discrepancy constant* + 18 µs

(0 to 1) Dual Input with test pulse:

Input signal response time = (2 × *Debounce filter time*) + (3 × *Debounce constant*) + *Discrepancy time* + *Discrepancy constant* + *Test pulse width* + 39 µs

Debounce filter time, discrepancy time, and test pulse widths are user-configurable parameters in the Functional Safety Editor.

Use the following table to determine the debounce constant based on debounce filter time.

Table 32. Debounce Constant

Debounce Filter Time	Debounce Constant
0 µs ≤ *debounce filter time* ≤ 16,383 µs	2 µs
16,500 µs ≤ *debounce filter time* ≤ 2,047.875 ms	126 µs
2,125 ms < *debounce filter time* ≤ 5 s	126 ms

Calculate the input discharge time using the following equation or use recommended input discharge times based on configuration, external load, cable length, and cable capacitance.

$$t = -R \times \left(C + 600\,pF\right) \times \ln\left(\frac{R \times 0.8\,mA + 5.7\,V}{R \times 0.8\,mA + 30\,V}\right)$$

Table 33. Recommended Input Discharge Times

Configuration	External Load[2]	Cable Length/ Capacitance	Input Discharge Time
• Single output • Dual output • Single output with internal test pulse • Dual output with internal test pulse	High Impedance	≤10 m and ≤1.8 nF	1,000 µs
	High Impedance	≤50 m and ≤9 nF	10,000 µs
	High Impedance	>50 m	100 ms
	≤3 kΩ	≤10 m and ≤1.8 nF	40 µs
	≤3 kΩ	≤50 m and ≤9 nF	1,000 µs
	≤3 kΩ	>50 m	10,000 µs
• Single output with external test pulse • Dual output with external test pulse • Single output with external readback • Single input with test pulse • Dual input with test pulse	>3 kΩ	≤50 m and ≤9 nF	1,000 µs
	>3 kΩ	>50 m	10,000 µs
	≤3 kΩ	≤50 m and ≤9 nF	40 µs
	≤3 kΩ	>50 m	10,000 µs

Analog Input Signal Response Time

Input signal response time describes the total time it takes to read the analog signal, convert the signal to digital, and propagate the signal to the User Program.

Single input	2 ms
Dual input (1oo2)	2 ms
Triple input (2oo3)	2 ms

Diagnostic Response Times

Use these specifications and equations to determine diagnostic response times. Diagnostic response times allow you to calculate fault response times for your C Series Functional Safety system.

[2] When the output load on the DO channel is a DI channel on the same module, load is >3 kΩ.

Digital Diagnostics

Open Circuit and Overcurrent Diagnostic Response Time

Open circuit	145 µs
Overcurrent	145 µs

To calculate open circuit diagnostic response times when using test pulses, use the following equation:

Open circuit diagnostic response time (with test pulses) = Test pulse width + Test pulse constant + 295 µs

- Test pulse width is a user-configurable parameter in the Functional Safety Editor.
- Use the following table to determine the test pulse constant based on test pulse width.

To calculate overcurrent diagnostic response times when using test pulses, use the following equation:

Overcurrent diagnostic response time (with test pulses) = Test pulse width + Test pulse constant + 295 µs

- Test pulse width is a user-configurable parameter in the Functional Safety Editor.
- Use the following table to determine the test pulse constant based on test pulse width.

Table 34. Test Pulse Constant

Test Pulse Width	Test Pulse Constant
≤16,384 µs	1 µs
≥16,500 µs	125 µs

Test Pulse Diagnostic Response Time

Use the following equation to calculate test pulse diagnostic response times.

Test pulse diagnostic response time = Test pulse period

- Test pulse period is a user-configurable parameter in the Functional Safety Editor.
- For more information on determining the test pulse period, refer to the *Test Pulses* section.

Discrepancy Diagnostic Response Time

Use the following equations to calculate discrepancy diagnostic response times.

Discrepancy diagnostic response time (0 to 1)

Discrepancy diagnostic response time = Discrepancy time + Discrepancy constant + Debounce filter time + Debounce constant + 18 µs

Discrepancy diagnostic response time (0 to 1) with test pulses configured:

Discrepancy diagnostic response time = (2 × *Debounce filter time*) + (3 × *Debounce constant*) + *Discrepancy time* + *Discrepancy constant* + *Test pulse width* + 39 µs

Discrepancy diagnostic response time (1 to 0) with or without test pulses configured

Discrepancy diagnostic response time = *Input discharge time* + *Discrepancy time* + *Discrepancy constant* + *Debounce filter time* + *Debounce constant* + 15 µs

- Discrepancy time, debounce filter time, and test pulse width are user-configurable parameters in the Functional Safety Editor.
- For more information about determining the debounce filter time when using test pulses, refer to the *Filter Times for Test Pulses* section.
- For more information on configuring discrepancy times, refer to the *Discrepancy Diagnostics for Digital Inputs* section.
- Use the following table to determine the discrepancy constant based on discrepancy time.

Table 35. Discrepancy Constant

Discrepancy Time	Discrepancy Constant
100 µs ≤ *discrepancy time* ≤ 16,383 µs	2 µs
16,500 µs ≤ *discrepancy time* ≤ 2,047.875 ms	126 µs
2,125 ms ≤ *discrepancy time* ≤ 30 s	126 ms

Note You cannot set discrepancy time to 0 µs when using dual input or dual input with test pulse.

- Use the following table to determine the debounce constant based on debounce filter time.

Table 36. Debounce Constant

Debounce Filter Time	Debounce Constant
0 µs < *debounce filter time* ≤ 16,383 µs	2 µs
16,500 µs ≤ *debounce filter time* ≤ 2,047.875 ms	126 µs
2,125 ms ≤ *debounce filter time* ≤ 5 s	126 ms

Note You cannot set debounce filter time to 0 µs when using dual input with test pulse.

Readback Diagnostic Response Time

For single and dual output configurations, use the following times.

Readback diagnostic response times (per output load)	
Very light	150 µs
Light	1,100 µs
Medium	11,025 µs
Heavy	110,150 µs

For single outputs configured with external readback, use the following equations to calculate readback diagnostic response times based on output line load.

Table 37. Readback Diagnostic Response Time Equations

Output Line Load	Readback Diagnostic Response Time Equation
Very light	*Debounce filter time* + *Debounce constant* + *Readback delay* + *Readback constant* + 61 µs
Light	*Debounce filter time* + *Debounce constant* + *Readback delay* + *Readback constant* + 1,021 µs
Medium	*Debounce filter time* + *Debounce constant* + *Readback delay* + *Readback constant* + 10,021 µs
Heavy	*Debounce filter time* + *Debounce constant* + *Readback delay* + *Readback constant* + 100,145 µs

- Debounce filter time, readback delay, and output line load are user-configurable parameters in the Functional Safety Editor.
- For more information on configuring readback delay refer to the *Readback Diagnostics* section.
- Use the following table to determine the debounce constant based on the debounce filter time.

Table 38. Debounce Constant

Debounce Filter Time	Debounce Constant
0 µs < *debounce filter time* ≤ 16,383 µs	2 µs
16,500 µs ≤ *debounce filter time* ≤ 2,047.875 ms	126 µs
2,125 ms ≤ *debounce filter time* ≤ 5 s	126 ms

- Use the following table to determine the readback constant based on the readback delay.

Table 39. Readback Constant

Readback Delay	Readback Constant
0 µs ≤ *readback delay* ≤ 16,383 µs	1 µs
16,500 µs ≤ *readback delay* ≤ 2,047.875 ms	125 µs
2,125 ms ≤ *readback delay* ≤ 5 s	125 ms

Analog Diagnostics (NI 9351 Only)

Overcurrent Diagnostic Response Time

Overcurrent fault 2 ms

Discrepancy Diagnostic Response Time

To calculate discrepancy diagnostic response times, use the following equation.

Discrepancy response time = *discrepancy time* + *discrepancy constant* + 2 ms

- Discrepancy time is a user-configurable parameter in the Functional Safety Editor.
- Use the following table to determine the discrepancy constant based on discrepancy time.

Table 40. Discrepancy Constant

Discrepancy Time	Discrepancy Constant
1000 µs ≤ *discrepancy time* ≤ 16,383 µs	2 µs
16,500 µs ≤ *discrepancy time* ≤ 2,047.875 ms	126 µs
2,125 ms ≤ *discrepancy time* ≤ 1800 s	126 ms

Application Processing Time

Application processing time represents how long the User Program needs to read an input and respond by setting an output. The application processing time consists of one or more application processing loops.

One iteration of the application processing loop takes 30 µs. At the beginning of every loop, the User Program samples inputs and sets outputs based on the current state. When the loop runs, the User Program evaluates transition conditions from the current state against the sampled inputs. If the inputs trigger a transition, the User Program updates to a new state. At the beginning of the next loop, the User Program sets outputs based on the new state.

If an input value changes after the beginning of the loop, that input will not be sampled until the beginning of the next loop. The total application response will require two iterations of the loop and take 60 µs.

Total application processing time increases for each state and associated transition in the defined execution path. Transition timers increase processing time by the number of loops required to resolve the timing logic as true.

Output Signal Response Time

Output signal response time describes the total time it takes to send the signal from the User Program to the output channel. For a maximum output signal response time, refer to the *Safety Response Time Specifications* section.

Power Down Response Time

Every DO channel implements redundant FETs to ensure the output channel can de-energize in the case of a single fault. If multiple internal faults occur and both DO FETs on a channel short, the module will go into Power Down Mode and de-energize all outputs. For the maximum power down response time for output self-diagnostics, refer to the *Safety Response Time Specifications* section.

Actuator Response Time

Actuator response time is the total time it takes for the output to propagate from the C Series Functional Safety module to the actuator and for the actuator to implement a control. Actuator response times are dependent on the peripheral devices, the safety application, resistors, the length of the cabling, and the total load on the cabling.

Tip If you calculated the discharge time based on output line load, you can use that time when calculating the actuator response time. The discharge time represents the amount of time it takes the signal to propagate from the C Series Functional Safety module to the actuator.

Safety Response Time Specifications

Use these specifications to calculate safety response times for your C Series Functional Safety system.

Note These specifications list *minimum guaranteed* times unless otherwise noted.

Total C Series Functional Safety Module Response Time

Digital Configurations	
Single input and dual input	122 µs
Single input with test pulse and dual input with test pulse	230 µs
Analog Configurations	
Single input (1oo1)	2.065 ms
Dual input (1oo2)	2.065 ms
Triple input (2oo3)	2.065 ms

Application processing time	60 µs
Output signal response time[3]	5 µs, maximum

Digital Input Signal Response Time

Input signal response time (1 to 0)	
Single or dual input	57 µs
Single or dual input with test pulses	165 µs

Note Input signal response time (1 to 0) assumes a 3 kΩ external pull-down, 50 m cable length with capacitance <180 pF/m, and the debounce filter and test pulse width set to the minimum values for the given configuration.

Input signal response time (0 to 1)	
Single input	20 µs
Dual input	122 µs
Single input with test pulses	346 µs
Dual input with test pulses	649 µs

Note Input signal response time (0 to 1) assumes the debounce filter, test pulse width, and discrepancy time are set to minimum values for the given configuration.

Input discharge time (1 to 0)	40 µs

Note Input discharge time represents a 30 V to low state voltage threshold crossing. The maximum input discharge time assumes a 3 kΩ external pull-down, 50 m cable length with a capacitance < 180 pF/m.

Analog Input Signal Response Time

Single input	2 ms
Dual input (1oo2)	2 ms
Triple input (2oo3)	2 ms

Automatic Self-Diagnostics Test Interval

Temperature fault	1.01 s
Analog input self-diagnostic faults	2 ms

[3] Includes delay time from a change in safety processing logic output to the digital output turning off. This does not include the discharge time for the output load.

Other automatic self-diagnostic faults	160 µs
Power down response time[4]	435 µs

User-Configurable Diagnostic Response Time

Digital Configurations	
Overcurrent	145 µs
Open circuit	145 µs
Test pulse	340 µs and configured with very light load
Internal readback	150 µs and configured with very light load
External readback	149 µs
Dual input discrepancy	159 µs
Analog Configurations	
Overcurrent fault	2 ms
Discrepancy fault	3.002 ms
Discrepancy warning	3.002 ms

Note User-configurable diagnostics assume the minimum values for test pulse width, test pulse period, discrepancy time, read back delay, and output line load.

Related Information

Safety System Response Time on page 59

Diagnostics

Diagnostics allow users to test the system periodically to detect and prevent possible unsafe failures. The C Series Functional Safety modules feature user-configurable diagnostics and automatic self-diagnostics.

Related Information

Setting Faults to Trigger Fail-Safe Mode on page 33
Transition Statements on page 45

[4] Includes delay time from a change in safety processing logic output until power down activates. This does not include the discharge time for the internal 40 µF bulk capacitance or the output load.

Fault Detection

Digital Input Configurations

Table 41. Single Input

Fault	Input Signal Status	
	Low	High
Short to ground	Not detected, safe state	Not detected, safe state
Short to V_{sup}	Not detected	Not detected
Open circuit	Not detected, safe state	Not detected, safe state

Table 42. Single Input with Test Pulse

Fault	Switch Condition	
	Open	Closed
Short to ground (test pulse output channel)	Overcurrent fault	Overcurrent fault
Short to V_{sup} (test pulse output channel)	Not detected, safe state	Test pulse fault
Short to ground (input channel)	Not detected, safe state	Overcurrent fault
Short to V_{sup} (input channel)	Test pulse fault	Test pulse fault
Open circuit	Not detected, safe state	Not detected, safe state

Table 43. Equivalent Dual Input

Fault	Input Signal Status	
	Low	High
Short to ground (one channel)	Not detected, safe state	Discrepancy fault
Short to ground (both channels)	Not detected, safe state	Not detected, safe state
Short to V_{sup} (one channel)	Discrepancy fault	Not detected
Short to V_{sup} (both channels)	Not detected	Not detected
Channels shorted together	Not detected	Not detected
Open circuit (one channel)	Not detected, safe state	Discrepancy fault
Open circuit (both channels)	Not detected, safe state	Not detected, safe state

Table 44. Complimentary Dual Input

Fault	Input Signal Status	
	Low	High
Short to ground (one channel)	Not detected	Discrepancy fault
Short to ground (both channels)	Discrepancy fault	Discrepancy fault
Short to V_{sup} (one channel)	Discrepancy fault	Not detected
Short to V_{sup} (both channels)	Discrepancy fault	Discrepancy fault
Channels shorted together	Discrepancy fault	Discrepancy fault
Open circuit (one channel)	Not detected	Discrepancy fault
Open circuit (both channels)	Discrepancy fault	Discrepancy fault

Table 45. Equivalent Dual Input with Test Pulse

Fault	Switch Condition	
	Open	Closed
Short to ground (one or both test pulse output channels)	Overcurrent fault	Overcurrent fault
Short to V_{sup} (one or both test pulse output channels)	Test pulse fault	Test pulse fault
Test pulse output channels shorted together	Test pulse fault	Test pulse fault
Short to ground (one or both input channels)	Not detected, safe state	Overcurrent fault
Short to V_{sup} (one or both input channels)	Test pulse fault	Test pulse fault
Input channels shorted together	Test pulse fault	Test pulse fault
Open circuit (one channel)	Not detected, safe state	Discrepancy fault
Open circuit (both channels)	Not detected, safe state	Not detected, safe state

Table 46. Complimentary Dual Input with Test Pulse

Fault	Switch Condition	
	Open	Closed
Short to ground (one or both test pulse output channels)	Overcurrent fault	Overcurrent fault
Short to V_{sup} (one or both test pulse output channels)	Test pulse fault	Test pulse fault
Test pulse output channels shorted together	Test pulse fault	Test pulse fault
Short to ground (one input channel)	Not detected	Overcurrent fault
Short to ground (both input channels)	Discrepancy fault	Discrepancy fault
Short to V_{sup} (one or both input channels)	Test pulse fault	Test pulse fault
Input channels shorted together	Test pulse fault	Test pulse fault
Open circuit (one channel)	Not detected	Discrepancy fault
Open circuit (both channels)	Discrepancy fault	Discrepancy fault

Related Information

Digital Configurations on page 20

Digital Output Configurations

Table 47. Single Output

Fault	Output Signal Value	
	True	False
Short to ground	Overcurrent fault	Not detected, safe state
Short to V_{sup}	Not detected	Readback fault
Open circuit	Open circuit fault	Not detected, safe state

Table 48. Single Output with Internal Test Pulse

Fault	Output Signal Value	
	True	False
Short to ground	Overcurrent fault	Not detected, safe state
Short to V_{sup}	Test pulse fault	Readback fault
Open circuit	Open circuit fault	Not detected, safe state

Table 49. Single Output with External Test Pulse

Fault	Output Signal Value	
	True	False
Short to ground (output channel)	Overcurrent fault	Not detected, safe state
Short to V_{sup} (output channel)	Test pulse fault	Readback fault
Short to ground (input channel)	Overcurrent fault	Not detected, safe state
Short to V_{sup} (input channel)	Test pulse fault	Readback fault
Open circuit	Open circuit fault	Not detected, safe state

Table 50. Single Output with External Readback

Fault	Output Signal Value	
	True	False
Short to ground (output channel)	Overcurrent fault	Not detected, safe state
Short to V_{sup} (output channel)	Not detected	Readback fault
Short to ground (input channel)	Overcurrent fault	Not detected, safe state
Short to V_{sup} (input channel)	Not detected	Readback fault
Open circuit	Open circuit fault	Not detected, safe state

Table 51. Dual Output

Fault	Output Signal Value	
	True	False
Short to ground (one or both channels)	Overcurrent fault	Not detected, safe state
Short to V_{sup} (one or both channels)	Not detected	Readback fault
Open circuit (one or both channels)	Open circuit fault	Not detected, safe state

Table 52. Dual Output with Internal Test Pulse

Fault	Output Signal Value	
	True	False
Short to ground (one or both channels)	Overcurrent fault	Not detected, safe state
Short to V_{sup} (one or both channels)	Test pulse fault	Readback fault
Open circuit (one or both channels)	Open circuit fault	Not detected, safe state

Table 53. Dual Output with External Test Pulse

Fault	Output Signal Value	
	True	False
Short to ground (output channel)	Overcurrent fault	Not detected, safe state
Short to V_{sup} (output channel)	Test pulse fault	Readback fault
Short to ground (input channel)	Overcurrent fault	Not detected, safe state
Short to V_{sup} (input channel)	Test pulse fault	Readback fault
Open circuit	Open circuit fault	Not detected, safe state

Related Information

Digital Configurations on page 20

Analog Input Configurations (NI 9351 Only)

Note Discrepancy fault and discrepancy warning are dependent on the measured current. For a discrepancy fault or warning to occur, the faulted channel must have a current delta with respect to the other channel(s) greater than the configured discrepancy current.

Table 54. Single Input

Condition	Resulting Current Region	Diagnostic Fault
Short to ground	Low low	—
Current exceeds specifications	High high	Overcurrent fault
Open circuit	Low low	—
Incorrect sensor output	Current region depends on the current input from the faulted sensor.	—

Table 55. Dual Input (1oo2)

Condition	Resulting Current Region	Diagnostic Fault(s)
Short to ground (one channel)	Low low	Discrepancy fault
Short to ground (both channels)	Low low	—
Current exceeds specifications (one channel)	High high unless the other channel is low low.	Overcurrent fault and discrepancy fault
Current exceeds specifications (both channels)	High high	Overcurrent fault
Open circuit (one channel)	Low low	Discrepancy fault
Open circuit (both channels)	Low low	—
Incorrect sensor output (one channel or both channels)	Current region depends on module FPGA voting outcome.	Discrepancy fault

Table 56. Triple Input (2oo3)

Condition	Resulting Current Region	Diagnostic Fault(s)
Short to ground (one channel)	Current region depends on module FPGA voting outcome. Module FPGA assigns low low current region to faulted channel.	Discrepancy warning
Short to ground (two channels)	Low low	Discrepancy warning
Short to ground (all channels)	Low low	—

Table 56. Triple Input (2oo3) (Continued)

Condition	Resulting Current Region	Diagnostic Fault(s)
Current exceeds specifications (one channel)	Current region depends on module FPGA voting outcome. Module FPGA assigns high high current region to faulted channel.	Discrepancy warning
Current exceeds specifications (two channels)	High high	Overcurrent fault and discrepancy warning
Current exceeds specifications (all channels)	High high	Overcurrent fault
Open circuit (one channel)	Current region depends on module FPGA voting outcome. Module FPGA assigns low low current region to faulted channel.	Discrepancy warning
Open circuit (two channels)	Low low	Discrepancy warning
Open circuit (all channels)	Low low	—
Incorrect sensor output (one channel)	Current region depends on module FPGA voting outcome.	Discrepancy warning
Incorrect sensor output (two channels or all channels)	Current region depends on module FPGA voting outcome.	Discrepancy fault or discrepancy warning

Related Information

Analog Configurations (NI 9351 Only) on page 21
Current Threshold Diagnostics on page 90
Thresholds on page 30
Discrepancy Current on page 33
Discrepancy Faults for Analog Input Configurations on page 86
Discrepancy Warning on page 88

User-Configurable Digital Diagnostics

User-configurable diagnostics allow your User Program to monitor and detect various faults related to your sensors, cabling, and final elements. User-configurable diagnostics populate the Fault section of the I/O Configuration table in the Functional Safety Editor based on the channel configuration you select. Refer to the following table for the fault options that populate for a selected configuration.

Table 57. Fault Configuration

Signal Type	Configuration	Fault Options
Digital Inputs	Single input	No fault options
	Single input with test pulse	Overcurrent fault, test pulse fault
	Dual input	Discrepancy fault
	Dual input with test pulse	Discrepancy fault, overcurrent fault, test pulse fault
Digital Outputs	Single output	Readback fault, overcurrent fault, open circuit fault
	Single output with external readback	Readback fault, overcurrent fault, open circuit fault
	Single output with internal test pulse	Overcurrent fault, open circuit fault, test pulse fault
	Single output with external test pulse	Overcurrent fault, open circuit fault, test pulse fault
	Dual output	Readback fault, overcurrent fault, open circuit fault
	Dual output with internal test pulse	Overcurrent fault, open circuit fault, test pulse fault
	Dual output with external test pulse	Overcurrent fault, open circuit fault, test pulse fault

Related Information

Digital Configurations on page 20

Test Pulses

You can configure internal or external test pulses on all digital signals. Test pulses verify that a line can be de-energized or driven to a safe state.

- For outputs with test pulses, DOn is available to use in the state machine diagram. The User Program will return test pulse faults on DOn.TestPulseFault.
- For inputs with test pulses, DOn is reserved and is not available to use in state machine diagram. The User Program will return test pulse faults on DIn.TestPulseFault.

Note Test pulses always pulse low.

Note Dual input and dual output configurations with test pulses are staggered to monitor for channel to channel shorts.

Note The User Program runs both readback diagnostics and test pulse diagnostics when you configure test pulses. Readback faults and test pulse faults report as a single event in the User Program. If you configure a channel with DIn.TestPulseFault or DOn.TestPulseFault, a test pulse fault or a readback fault would trigger a response. Scan Interface reports readback faults and test pulse faults as unique events. Scan Interface will report whether a readback fault or a test pulse fault triggered that response.

Note Make sure test pulse width does not cause the final element or safety actuator to switch off.

You can use test pulse faults in two ways:

- You can check the **Module failsafe** box for DIn.TestPulseFault or DOn.TestPulseFault. If a test pulse fault occurs, the module will go into Fail-safe Mode.
- You can use DIn.TestPulseFault or DOn.TestPulseFault in the transition condition on the State Machine Diagram. If a test pulse fault occurs, the User Program will transition to a user-configured state.

Configuring Test Pulses

Test pulses are defined by four configurable parameters in the I/O Configuration table. You can modify these parameters to meet the application requirements for your system.

- Output line loading
- Test pulse width
- Test pulse period
- Debounce filter

You can constrain the test pulse width and test pulse period to reduce inaccuracy and uncertainty. You must configure the test pulse width and test pulse period based on your output line load. Refer to the *Test Pulse Parameters* section for suggested parameter values.

Follow these steps to configure test pulses.

1. Set output line loading based on configuration. Refer to the *Output Line Load Recommendations* table in the *Output Line Load* section.
2. Set the minimum test pulse width based on output line loading. Setting the smallest possible test pulse width minimizes the input signal response time. Refer to the *Minimum test pulse parameters* table in the *Test pulse parameters* section.
3. Set the test pulse period to greater than or equal to four times the test pulse width.

   $Test\ pulse\ period \geq 4 \times Test\ pulse\ width$
4. Set the debounce filter based on the output loading and the test pulse parameters. Refer to the *Filter Times for Test Pulses* section.

Test Pulse Parameters

Use these parameter values to configure minimum test pulse widths and periods based on output loading.

Table 58. Minimum Test Pulse Parameters

Output Loading	Test Pulse Width	Test Pulse Period
Very Light	85 µs	340 µs
Light	1.045 ms	4.18 ms
Medium	10.045 ms	40.25 ms
Heavy	100.125 ms	400.5 ms

Filter Times for Test Pulses

Use the following equations to calculate the minimum debounce filter times for test pulses based on output loading.

Table 59. Minimum Debounce Filter Times

Output Loading	Test Pulse Width ≤ 16,384 µs	Test Pulse Width ≥ 16,500 µs
Very Light	*Debounce filter time* ≥ *test pulse width* + 23 µs	*Debounce filter time* ≥ *test pulse width* + 147 µs
Light		
Medium		
Heavy	*Debounce filter time* ≥ *test pulse width* + 271 µs	

You must confine the maximum debounce filter time for the following configurations to ensure that the debounce filter will not filter out the remaining high time of the input period. Use the following equations to calculate the maximum debounce filter times for test pulses based on output loading.

Table 60. Maximum Debounce Filter Times

Output Loading	Test Pulse Width ≤ 16,384 µs	Test Pulse Width ≥ 16,500 µs
Very Light	*Debounce filter time* < *test pulse period* - *Test pulse width* - 23 µs	*Debounce filter time* < *test pulse period* - *test pulse width* - 147 µs
Light		
Medium		
Heavy	*Debounce filter time* < *test pulse period* - *test pulse width* - 271 µs	

Readback Diagnostics

- Internal readback diagnostics are available on both **Single output** and **Dual output** configurations. When you select one of these configurations, DOn.ReadbackFault will populate the **Faults** table.
- Internal readback monitors the output signal and compares it to the configured output value in the User Program. If the output does not match the configured value, the User Program will return a fault on DOn.Readbackfault.
- External readback diagnostics are available for single outputs. Select the **Single output with external readback** in the configuration column.
- External readback monitors digital output signals for expected values. You can configure the readback delay to accommodate the time it takes for the signal to propagate. If the output signal does not match the expected value, the User Program will return a fault on DOn.Readbackfault.
  - You can set a maximum time for output values to propagate to the input channel.
  - A readback fault occurs when the propagation exceeds the defined maximum time limit.
  - To turn off readback delay, set value to 0.
  - Toggling outputs faster than the readback diagnostic response time will result in loss of the readback diagnostic.
  - If you set a digital output signal to Flash, you must configure the flash period larger than two times the readback diagnostic response time. *Flash period* = 2 × *Readback diagnostic response time*
- You can monitor external readback diagnostics anywhere on the output circuit, before or after the actuator.

You can use readback faults in two ways:

- You can check the **Module failsafe** box for DOn.ReadbackFault. If a readback fault occurs, the module will go into Fail-safe Mode.
- You can use DOn.ReadbackFault in the transition condition on the State Machine Diagram. If a readback fault occurs, the User Program will transition to a user-configured state.

Overcurrent Diagnostics (Digital)

You can enable overcurrent diagnostics for:

- Digital outputs in any configuration
- Digital inputs configured with test pulses

When the current on the configured channel exceeds module specifications, the module will respond based on how you configured the overcurrent fault. The pull-down menu in the overcurrent recovery column of the Faults table allows you to choose from three different responses for an overcurrent fault.

- **Failsafe**—The module goes into Fail-safe Mode.
- **Auto recover**—The channel de-energizes and the User Program returns a fault on DOn.OvercurrentFault or DIn.OvercurrentFault. You can use the fault as a transition condition on the state machine diagram. If an overcurrent fault occurs, the User Program

will transition to a user-configured state. After the **Recovery time** elapses, the fault will clear, allowing the user to energize the output again. If the current remains in an overcurrent state, the channel will de-energize again. The de-energize and auto recover cycle will continue until the module no longer reads an overcurrent condition.

- **No recover**—The channel de-energizes and the User Program returns a fault on DOn.OvercurrentFault or DIn.OvercurrentFault. You can use the fault as a transition condition on the state machine diagram. If an overcurrent fault occurs, the User Program will transition to a user-configured state. The digital output channel on which the fault occurred will de-energize. You must cycle V_{sup} power to the module to re-energize the channel.

Related Information

Overcurrent Recovery (Digital Configurations) on page 34

Open Circuit Diagnostics

You can enable open circuit diagnostics on any digital output channel. If the output channel is energized and the current falls below open circuit detection specifications, the User Program will return a fault on DOn.OpenCircuitFault.

You can use open circuit faults in two ways:

- You can check the **Module failsafe** box for DOn.OpenCircuitFault. If an open circuit fault occurs, the module will go into Fail-safe Mode.
- You can use DOn.OpenCircuitFault in the transition condition on the State Machine Diagram. If an open circuit fault occurs, the User Program will transition to a user-configured state.

Discrepancy Diagnostics for Digital Inputs

You can configure discrepancy diagnostics on consecutive even and odd digital input channels (DIn and DIn+1). Select the dual input configuration on the even-numbered channel (DIn) in the I/O Configuration table. The discrepancy diagnostic monitors both channels. If a discrepancy occurs, the User Program returns a fault on DIn.DiscrepancyFault.

If you configure the channels as equivalent inputs, both channels must be in the same state after the discrepancy time elapses. If you configure the channels as complementary inputs, the two channels must be in opposite states before the discrepancy time elapses.

The User Program returns discrepancy faults on the even-numbered channel (DIn). You can use discrepancy faults in two ways:

- You can check the **Module failsafe** box for DIn.DiscrepancyFault. If a discrepancy fault occurs, the module will go into Fail-safe Mode.
- You can use DIn.DiscrepancyFault in the transition condition on the State Machine Diagram. If a discrepancy fault occurs, the User Program will transition to a user-configured state.

Safe State

C Series Functional Safety modules are designed for use in De-Energize to Trip functions. A de-energized (low) state represents the safe state for a given channel. Refer to the following table for channel safe states and user program input values based on configuration and true value.

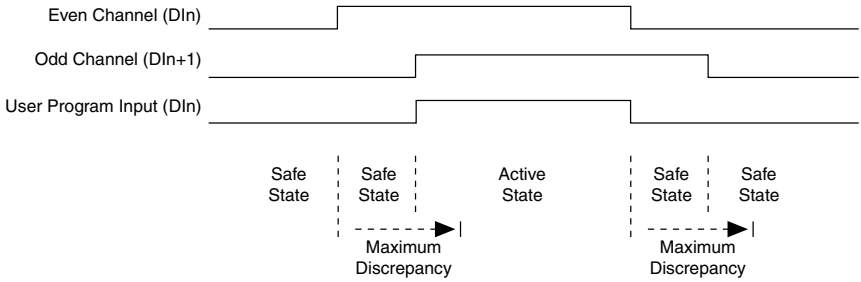
Table 61. Channel Safe States for Discrepancy Configurations

Discrepancy Fault Configuration	True Value	DI Even (DIn) Safe State	DI Odd (DIn+1) Safe State	User Program Input Value (DIn)
Equivalent	Active high	Off	Off	False
	Active low	Off	Off	True
Complementary	Active high	Off	On	False
	Active low	Off	On	True

Equivalent Inputs with Active High True Value

If you select active high on equivalent inputs, low is the safe state for both the even-numbered channel (DIn) and the odd-numbered channel (DIn+1). If both inputs go high, the User Program returns a fault on DIn.DiscrepancyFault.

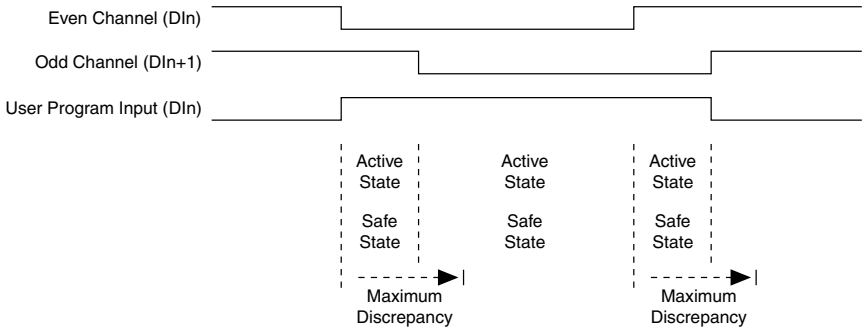
Figure 38. Equivalent Inputs with Active High True Value



Equivalent Inputs with Active Low True Value

If you select active low on equivalent inputs, high is the safe state for both the even-numbered channel (DIn) and the odd-numbered channel (DIn+1). If either input goes low, the User Program returns a fault on DIn.DiscrepancyFault.

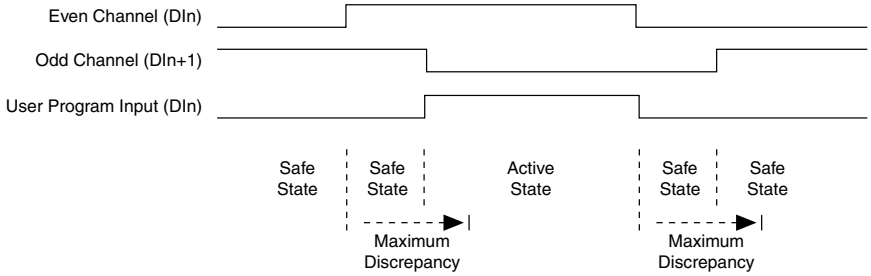
Figure 39. Equivalent Inputs with Active Low True Value



Complementary Inputs with Active High True Value

If you select active high on complementary inputs, low is the safe state for the even-numbered channel (DIn) and high is the safe state for the odd-numbered channel (DIn+1). If both inputs go into an active state, the User Program returns a fault on DIn.DiscrepancyFault.

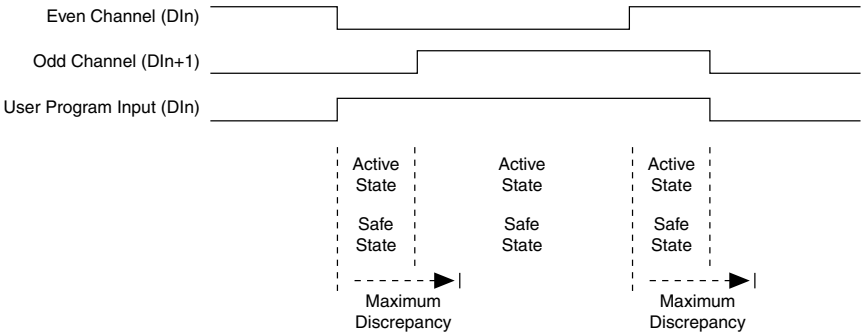
Figure 40. Complementary Inputs with Active High True Value



Complementary Inputs with Active Low True Value

If you select active low on complementary inputs, high is the safe state for the even-numbered channel (DIn) and the low is the safe state for the odd-numbered channel (DIn+1). If either input goes into an active state, the User Program returns a fault on DIn.DiscrepancyFault.

Figure 41. Complementary Inputs with Active Low True Value



Note If one input channel toggles twice while the other input channel remains the same, the User Program returns a fault on DIn.DiscrepancyFault. This diagnostic is monitored post-debounce filter.

User-Configurable Analog Diagnostics (NI 9351 Only)

User-configurable diagnostics allow your User Program to monitor and detect various faults related to your sensors, cabling, and final elements. User-configurable diagnostics populate the Fault section of the I/O Configuration table in the Functional Safety Editor based on the channel configuration you select. Refer to the following table for the fault options that populate for a selected configuration.

Table 62. Analog Fault Configuration

Configuration	Fault Options
Single input	Overcurrent fault
Dual input (1oo2)	Overcurrent fault, discrepancy fault
Triple input (2oo3)	Overcurrent fault, discrepancy fault, discrepancy warning

Related Information

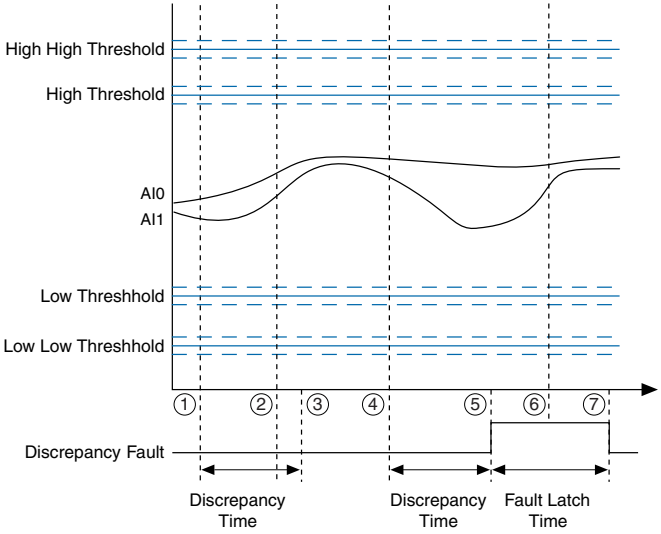
Analog Configurations (NI 9351 Only) on page 21

Discrepancy Faults for Analog Input Configurations

The module FPGA returns a Discrepancy Fault to the User Program in the following situations:

- In a dual input (1oo2) configuration the two channels read currents that differ by more than the Discrepancy Current for a time greater than the Discrepancy Time.
- In a triple input (2oo3) configuration all three channels read currents that differ from each other by more than the Discrepancy Current for a time greater than the Discrepancy Time.

Figure 42. Discrepancy Fault Signal Diagram



1. Measured currents exceed discrepancy parameter limits.
2. Measured currents return to defined parameter limits.
3. Discrepancy time expires. Since measured currents are within defined parameter limits, the module FPGA does not generate a discrepancy fault.
4. Measured currents exceed discrepancy parameter limits.
5. Discrepancy time expires. Since measured currents still exceed parameter limits, the module FPGA asserts a discrepancy fault and the fault latch timer starts.
6. Measured currents return to defined parameter limits.
7. Fault latch time expires. Since measured currents are within defined parameter limits, the module FPGA no longer asserts a discrepancy fault.

Tip To ensure that the module detects and responds to discrepancy faults, set the discrepancy current parameter as low as the system will allow.

Note You must set the discrepancy current greater than 0 mA.

You can use discrepancy faults in two ways:

- You check the **Module failsafe** box for AIn.DiscrepancyFault. If a discrepancy fault occurs, the module will go into Fail-safe Mode.
- You can use AIn.DiscrepancyFault in the transition condition on the state machine diagram. If a discrepancy fault occurs, the User Program will transition to a user-configured state.

Related Information

Discrepancy Time (Analog Configurations) on page 32
Discrepancy Current on page 33
Analog Input Configurations (NI 9351 Only) on page 76

Discrepancy Warning

Discrepancy Warnings are only used in Triple Input (2oo3) configurations. The module FPGA returns a Discrepancy Warning to the User Program when one of the three input channels read a current discrepancy that exceeds the defined parameters.

Tip To ensure that the module detects and responds to discrepancy warnings, set the discrepancy current parameter as low as the system will allow.

Note You must set the discrepancy current greater than 0 mA.

You can use AIn.DiscrepancyWarning as a transition condition on the state machine diagram. If a discrepancy warning fault occurs, the User Program will transition to a user-configured state.

Related Information

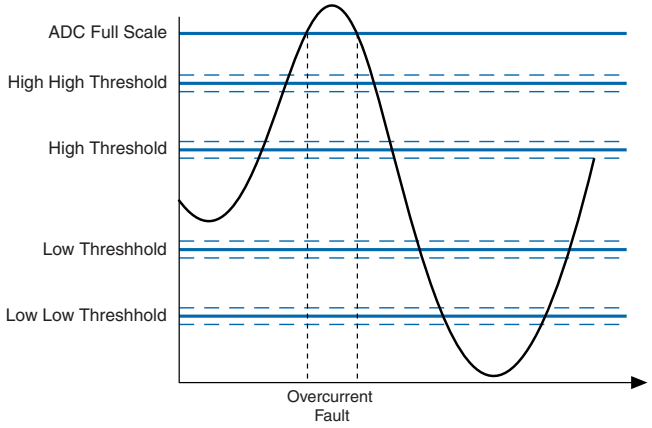
Discrepancy Time (Analog Configurations) on page 32
Discrepancy Current on page 33
Analog Input Configurations (NI 9351 Only) on page 76

Overcurrent Diagnostics (Analog)

Each analog input configuration has overcurrent detection. Overcurrent diagnostics in dual input and triple input configurations use the same voting strategy as threshold diagnostics. For more information on voting strategies, refer to the *Voting Strategies in Dual Input (1oo2) and Triple Input (2oo3) Configurations* section.

An overcurrent condition occurs any time the input current exceeds the largest current the channel ADC can digitize, shown as ADC Full Scale in the following figure.

Figure 43. Overcurrent Signal Diagram



- In dual input (1oo2) configurations, an overcurrent condition on one or two channels will generate an overcurrent fault.
- In triple input (2oo3) configurations, an overcurrent condition on two or three channels will generate an overcurrent fault.

The User Program will read overcurrent faults based on configuration.

Table 63. Overcurrent Faults for Analog Configurations

Analog Configuration	Available Overcurrent Faults
Single input (1oo1)	AIn.OvercurrentFault, where n is the number of the channel configured for single input
Dual input (1oo2)	AI0.OvercurrentFault or AI2.OvercurrentFault (the even-numbered channel of dual input channel pair)
Triple input (2oo3)	AI0.OvercurrentFault only

Scan Interface will read overcurrent faults for each individual channel, regardless of configuration. In a triple input (2oo3) configuration, a single channel in an overcurrent condition will not generate an overcurrent fault in the User Program, but will report an overcurrent fault for that channel on the Check Channel Status method in LabVIEW.

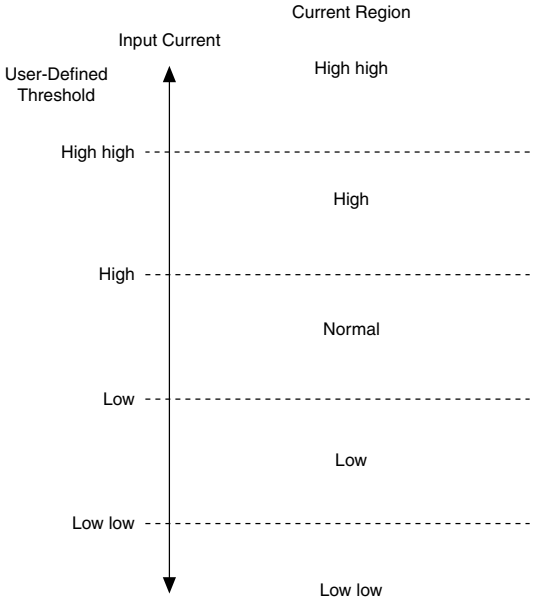
You can use overcurrent faults in two ways:

- You check the **Module failsafe** box for AIn.OvercurrentFault. If an overcurrent fault occurs, the module will go into Fail-safe Mode.
- You can use AIn.OvercurrentFault in the transition condition on the state machine diagram. If an overcurrent fault occurs, the User Program will transition to a user-configured state.

Current Threshold Diagnostics

The C Series Functional Safety module uses five current regions to describe the state of the current input to the channel. You define these regions by setting four current thresholds in the Functional Safety Editor.

Figure 44. Current Regions

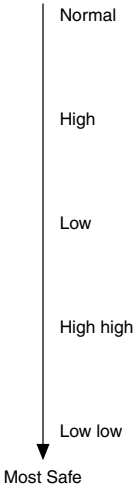


The module FPGA converts current inputs into digital signals to return the current region defined by the configured thresholds. The User Program only reads one current region for each analog configuration. If all the channels in a dual input or triple input configuration read the same current region, the module FPGA returns that current region. If one or more of the channel readings comes back in different current regions than the module FPGA implements a voting strategy to determine which current region to return to the User Program.

C Series Functional Safety modules employ the De-Energize to Trip principle, so the voting strategy defines the Low low as the most safe current region. The second most safe current region is high high because high high could be a result of a fault such as short to AI Vsup or overcurrent.

Refer to the following image for the evaluation of most safe current regions that the module FPGA uses to resolve voting situations.

Figure 45. Current Region Voting Strategy



Tip To ensure the module FPGA returns a low low region for a channel, set the low low threshold high enough to filter out inaccuracy and noise.

Related Information

Thresholds on page 30

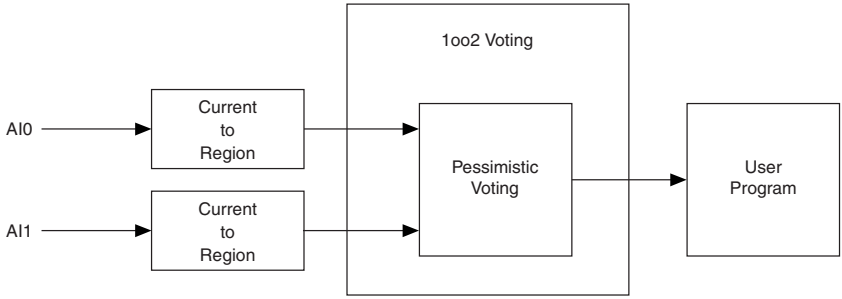
Hysteresis on page 32

Analog Input Configurations (NI 9351 Only) on page 76

Voting in Dual Input (1oo2) Safety Architectures

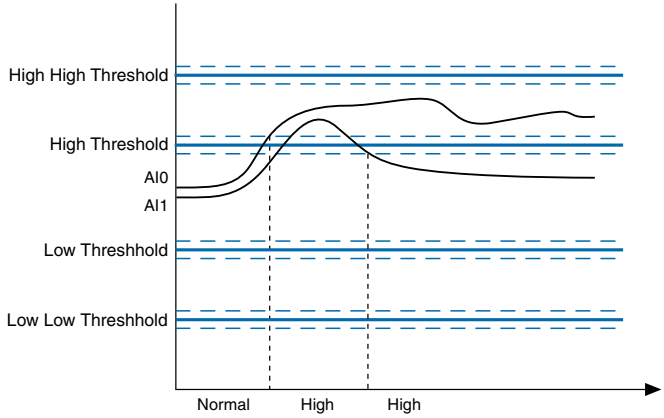
The module FPGA employs a pessimistic voting strategy in a dual input (1oo2) architecture. When the module reads different currents on the two configured channels, the module FPGA will return the safest region of the two.

Figure 46. Dual Input (1oo2) Voting



1. The C Series Functional Safety module reads the current on the two input channels.
2. The module FPGA assigns a current region to each channel based on the configured thresholds.
3. If the channels are in the same current region, the voting selects that region. If not, the voting selects the most safe of the two current regions.
4. The User Program uses the single current region selected and returned by the voting strategy.

Figure 47. Dual Input (1oo2) Signal Diagram



The following table shows the voting results for each combination of current regions in a dual input configuration.

Table 64. Dual Input (1oo2) Voting

AIn Current Region	AIn+1 Current Region	Region Returned to the User Program
High high	High high	High high
High high	High	High high

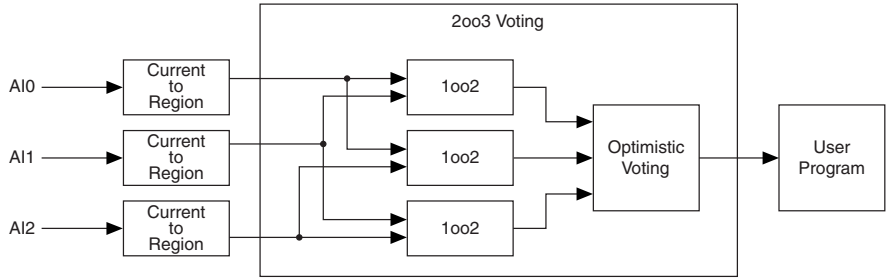
Table 64. Dual Input (1oo2) Voting (Continued)

AIn Current Region	AIn+1 Current Region	Region Returned to the User Program
High high	Normal	High high
High high	Low	High high
High high	Low low	Low low
High	High high	High high
High	High	High
High	Normal	High
High	Low	Low
High	Low low	Low low
Normal	High high	High high
Normal	High	High
Normal	Normal	Normal
Normal	Low	Low
Normal	Low low	Low low
Low	High high	High high
Low	High	Low
Low	Normal	Low
Low	Low	Low
Low	Low low	Low low
Low low	High high	Low low
Low low	High	Low low
Low low	Normal	Low low
Low low	Low	Low low
Low low	Low low	Low low

Voting in Triple Input (2oo3) Safety Architectures

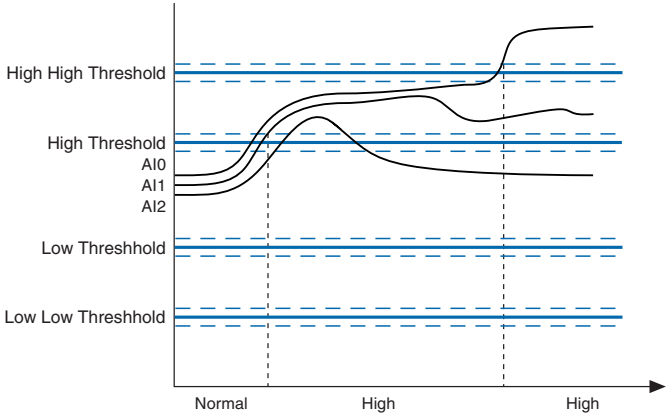
In a 2oo3 architecture, the module FPGA employs a two step voting process before returning a current region to the User Program.

Figure 48. Triple Input (2oo3) Voting



1. The C Series Functional Safety module reads the current on three input channels.
2. The module FPGA assigns a current region to each channel based on the configured thresholds.
3. The module FPGA employs a pessimistic voting strategy to compare the current regions between each pair of channels: 0 and 1, 1 and 2, 0 and 2.
4. The voting strategy selects the most safe current region of each channel pair to produce three reference regions.
5. The module FPGA employs an optimistic voting strategy to compare the three reference regions. The voting strategy selects the least safe reference region of the three.
6. The User Program reads the single current region selected and returned by the module FPGA.

Figure 49. Triple Input (2oo3) Signal Diagram



The following table shows the current region returned to the User Program for each unique combination of inputs.

Table 65. Triple Input (2oo3) Voting

Current Region Combinations			Region Returned to the User Program
High high	High high	High high	High high
High high	High high	High	High high
High high	High high	Normal	High high
High high	High high	Low	High high
High high	High high	Low low	High high
High high	High	High	High
High high	High	Normal	High
High high	High	Low	Low
High high	High	Low low	High high
High high	Normal	Normal	Normal
High high	Normal	Low	Low
High high	Normal	Low low	High high
High high	Low	Low	Low
High high	Low	Low low	High high
High high	Low low	Low	High high
High high	Low low	Low low	Low low
High	High	High	High
High	High	Normal	High
High	High	Low	High
High	High	Low low	High
High	Normal	Normal	Normal
High	Normal	Low	High
High	Normal	Low low	High
High	Low	Low	Low
High	Low	Low low	Low
High	Low low	Low low	Low low
Normal	Normal	Normal	Normal

Table 65. Triple Input (2oo3) Voting (Continued)

Current Region Combinations			Region Returned to the User Program
Normal	Normal	Low	Normal
Normal	Normal	Low low	Normal
Normal	Low	Low	Low
Normal	Low	Low low	Low
Normal	Low low	Low low	Low low
Low	Low	Low	Low
Low	Low	Low low	Low
Low	Low low	Low low	Low low
Low low	Low low	Low low	Low low

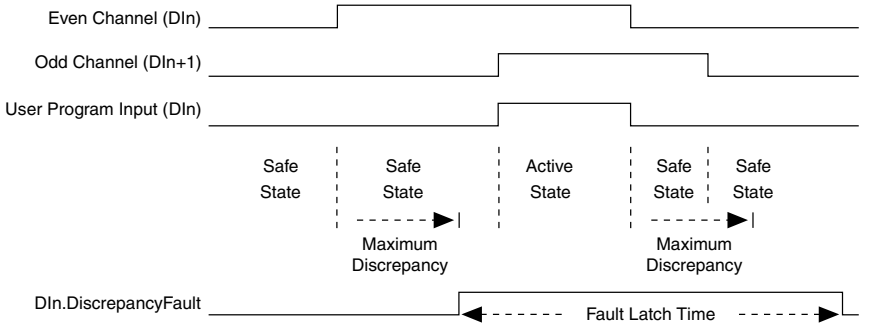
Fault Latching

C Series Functional Safety modules latch detected faults. Fault latch time is a programmable parameter on the **Module and Diagram** tab of the configuration pane.

The NI 9350 has a single fault latch timer which resets for any fault.

The NI 9351 has two fault latch timers. One for digital faults and one for analog faults. The fault latch time parameter applies to both fault timers. Digital faults will not reset the analog fault time and analog faults will not reset the digital fault time.

Figure 50. Fault Latch Time on Discrepancy Faults



Latched faults persist until the configured latch time has elapsed. Detected faults will remain latched when the operational mode changes from Verification Mode to Operational Mode or from Operational Mode to Verification Mode. When the User Program restarts, a latched fault

from the previous mode can trigger Fail-safe Mode or transition the User Program to a new state.

Tip To monitor faults through Scan Interface, set the fault latch time larger than the Scan Interface scan rate.

Automatic Self-Diagnostics

C Series Functional Safety modules run a range of automatic self-diagnostics that continuously check for module hardware and power supply faults. Automatic self-diagnostic faults send the module into Fail-safe Mode. The Check Module Status method in LabVIEW will return an Internal Error or Temperature Fault, depending on the fault.

Some of the automatic self-diagnostics implemented by all C Series Functional Safety modules include:

- DI reference compares
- Redundant DO FETs
- Watchdog timers custom-designed for high accuracy
- Internal temperature monitoring
- Power supply voltage monitoring on external supply and all internal rails
- Redundant ADC, signal conditioning and muxes on AI channels
- Range checks on AI channels
- High accuracy AI reference compares

Note Errors detected by automatic self-diagnostics on analog configurations will assign a low low current region to all affected channels and generate a flashing pattern on the Internal Fault LED.

Tip Every DO channel implements redundant FETs and meets SIL3 capability. To create additional redundancy on DO channels, only connect loads above 20 mA. If both FETs fail in the closed position, the open circuit diagnostic will return a fault.

LED Diagnostics

V_{sup}/Status LED

The V_{sup}/Status LED is green and indicates the status of the C Series Functional Safety Module and the User Program.

Table 66. V_{sup}/Status LED Flash Behavior

LED Color	LED Pattern	Module Status	User Program Status
Green	On	Operational Mode	Running
	Flashing	Operational Mode	Not running
		Verification Mode	Running
		Verification Mode	Not running
		—	No User Program downloaded
		User Program Download Mode	—
		Fail-safe Mode	Not running
		Upgrading firmware	—
	Off	Powered off	—

Internal Fault LED

The Internal Fault LED is red and indicates the module is in Fail-safe Mode. This includes Fail-safe Mode triggered by automatic self-diagnostic faults or user-configurable diagnostic faults.

Table 67. Internal Fault LED Flash Behavior

LED Color	LED Pattern	Indication
Red	Flashes once then pauses	A user-configured fault has triggered Fail-safe Mode.
	Flashes twice then pauses	A temperature fault has been detected.
	Flashes three times then pauses	A power fault has been detected.
	Flashes more than three times then pauses	An internal fault has been detected. **Record the number of flashes and contact National Instruments.**
	Off	No Internal Fault has been detected.

I/O Fault LED

The I/O Fault LED is red and indicates faults on I/O channels. The I/O Fault LED displays both a fault code and a channel code. You will first see a series of short flashes that indicate the fault type, then a series of long flashes that indicate the channel number.

Long flashes indicate channel number so that *n + 1 flashes = channel n*.

For example, a test pulse or readback fault on DO2 will cause the I/O Fault LED to display the following flash pattern: Two short flashes + pause + three long flashes

Table 68. I/O Fault LED Fault Code

LED Pattern	Indication
Flashes once then pauses	An overcurrent fault has been detected on a digital input or digital output channel.
Flashes twice then pauses	A test pulse or readback fault has been detected.
Flashes three times then pauses	A discrepancy fault has been detected on a digital dual input configuration.
Flashes four times then pauses	An overcurrent fault has been detected on an analog input channel.
Flashes five times then pauses	A discrepancy fault has been detected on an analog dual input or triple input configuration.
Off	No I/O Fault has been detected.

UserLED0

- UserLED0 behavior is user-configurable.
- Enable or disable the LED and set the flash period in the I/O Configuration.
- Set the output value of UserLED0 in the state machine diagram.

Finding Resources

Developing a Functional Safety System

- **Tutorials**—For tutorials on developing state machines, go to *ni.com/info* and enter Info Code `safetytutorial`.
- **Whitepapers**—For whitepapers with detailed information about creating a Functional Safety system, go to *ni.com/info* and enter Info Code `safetypapers`.
- **Videos**—For videos about using C Series Functional Safety modules, go to *ni.com/info* and enter Info Code `safetyvideo`.

Using the Hardware

- **Hardware documentation**—For getting started guides, datasheets, user manuals, and other hardware documentation for CompactRIO controllers and the C Series Functional Safety modules, visit *ni.com/manuals*.

Using LabVIEW

- *LabVIEW Help*—Use the LabVIEW Help to access information about LabVIEW programming concepts, step-by-step instructions for using LabVIEW, and reference

information about LabVIEW VIs, functions, palettes, menus, tools, properties, methods, events, dialog boxes, and so on. The LabVIEW Help also lists the LabVIEW documentation resources available from National Instruments. Access the LabVIEW Help by selecting **Help»Search the LabVIEW Help**.

- *Getting Started with LabVIEW*—Use this document as a tutorial to familiarize yourself with the LabVIEW graphical programming environment and the basic LabVIEW features you use to build data acquisition and instrument control applications. Access the Getting Started with LabVIEW PDF by selecting **Start»All Programs»National Instruments»LabVIEW»LabVIEW Manuals»LV_Getting_Started.pdf**.
- *Getting Started with the LabVIEW Real-Time Module*—Use this document to learn how to develop a real-time project and VIs, from setting up RT targets to building, debugging, and deploying real-time applications. Access the Getting Started with the LabVIEW Real-Time Module PDF by selecting **Start»All Programs»National Instruments»LabVIEW»LabVIEW Manuals»RT_Getting_Started.pdf**.
- *CompactRIO Reference and Procedures (Scan Interface)*—Use this help file to learn about using the CompactRIO system in Scan Interface programming mode. To access this help file from LabVIEW, select **Help»Search the LabVIEW Help**, then expand Real-Time Module on the Contents tab and select CompactRIO Reference and Procedures (Scan Interface).
- **Example VIs**—To search for an example Functional Safety VI, select **Help»Find Examples...** in LabVIEW.

Finding Known Issues

- **Functional Safety Editor**—For a list of known issues for the Functional Safety Editor, go to *ni.com/info* and enter Info Code `safetyissues`.
- **C Series Functional Safety Module Firmware**—For a list of known issues for the firmware that installs on the C Series Functional Safety module, refer to the *Known Issues for Firmware Versions* section.
- **NI-RIO Device Drivers**—For a list of known issues and bug fixes for all NI-RIO Device Drivers versions, go to *ni.com/info* and enter Info Code `rioknownissues`.
- **NI-RIO Device Drivers May 2017**—For a list of known issues for NI-RIO Device Drivers May 2017 release, go to *ni.com/info* and enter Info Code `rio2017knownissues`.

Related Information

C Series Functional Safety Systems on page 3

Updating Safety Software and Firmware

NI provides application updates, new features, and bug-fixes for software through the NI Update Service.

C Series Functional Safety modules ship with installed firmware that runs the User Program. Updates to this firmware will be distributed by NI as appropriate.

For information about receiving and managing updates, refer to the *NI Update Service Help*.

Note Visit *ni.com/critical-updates* for information about available critical and security updates for NI software.

C Series Functional Safety Firmware

Firmware Release Versions

The following sections detail the C Series Functional Safety Firmware releases and associated known issues. You can read the firmware version through Scan Interface. To determine the firmware version, add a Property Node to your monitoring VI in LabVIEW.

Caution You must verify and formally document that your safety application is not affected by the known issues documented below.

Table 69. C Series Functional Safety Firmware Releases

Module	Version	Date Released	Notes
NI 9350	0x17081412	September 2017	This version shipped with the initial release of the NI 9350.
	0x18010811	March 2018	This version implements a fix for undetected faults when driving heavy loads.
NI 9351	0x18031516	May 2018	This version shipped with the initial release of the NI 9351.

Table 70. C Series Functional Safety Firmware Known Issues List

Known Issue	Version Affected		
	0x17081412	0x18010811	0x18031516
Open circuit faults on digital outputs	✓	✓	
Passthrough value behavior	✓	✓	
Overcurrent on dual digital inputs with test pulses	✓	✓	✓
Undetected faults when driving heavy loads	✓		

Known Issues for Firmware Versions

Open Circuit Faults on Digital Outputs

An open circuit fault will persist for the duration of the fault latch time when the User Program restarts as a result of changing operation modes.

Passthrough Value Behavior

A channel driven true by passthrough from the monitoring VI in LabVIEW remains high after the module has been removed from the CompactRIO controller or if communication to the controller is lost.

Overcurrent on Dual Digital Inputs with Test Pulses

In a dual input with test pulse configuration, an overcurrent condition on the odd-numbered channel causes both channels to de-energize. An overcurrent condition on an even-numbered channel only causes the even-numbered channel to de-energize.

Undetected Faults When Driving Heavy Loads

In the following digital output configurations, the module will not generate test pulses, detect shorts to Vsup, or detect stuck high conditions.

- Single output with heavy output loading when the corresponding DI channel is unused or the corresponding DI channel is used and has a debounce filter of 0 µs to 16,383 µs or 2,125 ms to 5 s.
- Dual output with heavy output loading when the corresponding DI channel is unused or the corresponding DI channel is used and has a debounce filter of 0 µs to 16,383 µs or 2,125 ms to 5 s.
- Single output with external readback with heavy output loading and a debounce filter of 85 µs to 16,383 µs or 2,125 ms to 5 s.
- Single output with internal test pulse with heavy output loading when the corresponding DI channel is unused or the corresponding DI channel is used and has a debounce filter of 0 µs to 16,383 µs.
- Dual output with internal test pulse with heavy output loading when the corresponding DI channel is unused or the corresponding DI channel is used and has a debounce filter of 0 µs to 16,383 µs.

To avoid this condition, use one of the following methods:

- Use a debounce time between 16.5 ms and 2.047s.
- Verify that your application requires heavy output line loading. Refer to the *Output Line Load* section.
  - If your application does not require heavy output line loading, use a maximum of medium output line loading.
  - If your application requires heavy output line loading, add a pulldown resistor to the output channel to reduce the output line load.

Worldwide Support and Services

The NI website is your complete resource for technical support. At *ni.com/support*, you have access to everything from troubleshooting and application development self-help resources to email and phone assistance from NI Application Engineers.

Visit *ni.com/services* for information about the services NI offers.

Visit *ni.com/register* to register your NI product. Product registration facilitates technical support and ensures that you receive important information updates from NI.

NI corporate headquarters is located at 11500 North Mopac Expressway, Austin, Texas, 78759-3504. NI also has offices located around the world. For support in the United States, create your service request at *ni.com/support* or dial 1 866 ASK MYNI (275 6964). For support outside the United States, visit the *Worldwide Offices* section of *ni.com/niglobal* to access the branch office websites, which provide up-to-date contact information.

377937C-01 May 15, 2018