Logix 5000 Controllers

Major, Minor, and I\O Faults

1756 ControlLogix, 1756 GuardLogix, 1769 CompactLogix, 1769 Compact GuardLogix, 1789 SoftLogix, 5069 CompactLogix, 5069 Compact GuardLogix, Studio 5000 Logix Emulate

Rockwell Automation Publication 1756-PM014N-EN-P - March 2022
Supersedes Publication 1756-PM014M-EN-P - September 2020

Rockwell Automation

Programming Manual

Original Instructions

Important User Information

Read this document and the documents listed in the additional resources section about installation, configuration, and operation of this equipment before you install, configure, operate, or maintain this product. Users are required to familiarize themselves with installation and wiring instructions in addition to requirements of all applicable codes, laws, and standards.

Activities including installation, adjustments, putting into service, use, assembly, disassembly, and maintenance are required to be carried out by suitably trained personnel in accordance with applicable code of practice.

If this equipment is used in a manner not specified by the manufacturer, the protection provided by the equipment may be impaired.

In no event will Rockwell Automation, Inc. be responsible or liable for indirect or consequential damages resulting from the use or application of this equipment.

The examples and diagrams in this manual are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular installation, Rockwell Automation, Inc. cannot assume responsibility or liability for actual use based on the examples and diagrams.

No patent liability is assumed by Rockwell Automation, Inc. with respect to use of information, circuits, equipment, or software described in this manual.

Reproduction of the contents of this manual, in whole or in part, without written permission of Rockwell Automation, Inc., is prohibited.

Throughout this manual, when necessary, we use notes to make you aware of safety considerations.

WARNING: Identifies information about practices or circumstances that can cause an explosion in a hazardous environment, which may lead to personal injury or death, property damage, or economic loss.

ATTENTION: Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss. Attentions help you identify a hazard, avoid a hazard, and recognize the consequence.

IMPORTANT Identifies information that is critical for successful application and understanding of the product.

Labels may also be on or inside the equipment to provide specific precautions.

SHOCK HAZARD: Labels may be on or inside the equipment, for example, a drive or motor, to alert people that dangerous voltage may be present.

BURN HAZARD: Labels may be on or inside the equipment, for example, a drive or motor, to alert people that surfaces may reach dangerous temperatures.

ARC FLASH HAZARD: Labels may be on or inside the equipment, for example, a motor control center, to alert people to potential Arc Flash. Arc Flash will cause severe injury or death. Wear proper Personal Protective Equipment (PPE). Follow ALL Regulatory requirements for safe work practices and for Personal Protective Equipment (PPE).

Rockwell Automation recognizes that some of the terms that are currently used in our industry and in this publication are not in alignment with the movement toward inclusive language in technology. We are proactively collaborating with industry peers to find alternatives to such terms and making changes to our products and content. Please excuse the use of such terms in our content while we implement these changes.

This manual includes new and updated information. Use these reference tables to locate changed information.

Grammatical and editorial style changes are not included in this summary.

Global changes

The Legal noticeshave been updated.

New or enhanced features

This table contains a list of topics changed in this version, the reason for the change, and a link to the topic that contains the changed information.

Topic Name	Reason
Minor fault codes on page 29	Replaced the Minor Fault Codes list with a link to the Logix 5000 Controller Fault Codes spreadsheet, which contains a complete list of fault codes.
Major fault codes on page 25	Replaced the Major Fault Codes list with a link to the Controller Fault Codes spreadsheet.
I/O fault codes on page 33	Replaced the I/O Fault Codes list with a link to the Controller Fault Codes spreadsheet.

Table of Contents

This manual shows how to monitor and handle major and minor controller faults. The manual also provides lists of major, minor, and I/O fault codes to use to troubleshoot the system.

This manual is one of a set of related manuals that show common procedures for programming and operating Logix 5000 controllers.

For a complete list of common procedures manuals, refer to the Logix 5000 Controllers Common Procedures Programming Manual, publication 1756-PM001.

The term Logix 5000 controller refers to any controller based on the Logix 5000 operating system.

Studio 5000 environment

The Studio 5000 Automation Engineering & Design Environment® combines engineering and design elements into a common environment. The first element is the Studio 5000 Logix Designer® application. The Logix Designer application is the rebranding of RSLogix 5000® software and will continue to be the product to program Logix 5000™ controllers for discrete, process, batch, motion, safety, and drive-based solutions.

The Studio 5000® environment is the foundation for the future of Rockwell Automation® engineering design tools and capabilities. The Studio 5000 environment is the one place for design engineers to develop all elements of their control system.

Additional resources

These documents contain additional information concerning related products from Rockwell Automation.

Resource	Description
Logix5000 Controllers General Instructions Reference Manual, publication 1756-RM003	Provides programmers with details about each available instruction for a Logix5000 controller.
Product Certifications website, http://www.ab.com	Provides declarations of conformity, certificates, and other certification details.

View or download publications at http://www.rockwellautomation.com/literature/. To order paper copies of technical documentation, contact your local Allen-Bradley distributor or Rockwell Automation sales representative.

Legal Notices

Rockwell Automation publishes legal notices, such as privacy policies, license agreements, trademark disclosures, and other terms and conditions on the Legal Notices page of the Rockwell Automation website.

End User License Agreement (EULA)

You can view the Rockwell Automation End-User License Agreement ("EULA") by opening the License.rtf file located in your product's install folder on your hard drive.

Open Source Licenses

The software included in this product contains copyrighted software that is licensed under one or more open source licenses. Copies of those licenses are included with the software. Corresponding Source code for open source packages included in this product are located at their respective web site(s).

Alternately, obtain complete Corresponding Source code by contacting Rockwell Automation via the Contact form on the Rockwell Automation website:
http://www.rockwellautomation.com/global/about-us/contact/contact.page
Please include "Open Source" as part of the request text.

A full list of all open source software used in this product and their corresponding licenses can be found in the OPENSOURCE folder. The default installed location of these licenses is `C:\Program Files (x86)\Common Files\Rockwell\Help\<Product Name>\Release Notes\OPENSOURCE\index.htm`.

Major Faults

This chapter explains major fault codes and how to work with them in the Logix Designer application.

Major Fault State

If a fault condition occurs that prevents an instruction from running, the instruction aborts and the controller reports a major fault. A major fault halts logic execution and the controller switches to faulted mode (the OK LED flashes red).

Depending on the application, you may not want all major faults to shut down the system. If you do not want all major faults to shut down the system, create a fault routine to clear the fault and let the application continue to run.

The process of resuming execution after the fault clears is known as fault recovery.

IMPORTANT Do not use fault routines to continually clear all faults on the controller. Program the fault routine to be selective in the types and number of faults cleared. It is also a good idea to log the fault occurrence to analyze it later.

IMPORTANT When an instruction generates an error due to a fault (for example, a COP with an indirect addressing programming error), the fault routine skips the instruction and does not run. This occurs with all instructions.

Example: In a system that uses recipe numbers as indirect addresses, an incorrectly typed number could produce a major fault.
To keep the entire system from shutting down in the event of this fault, program a fault routine to clear type 4, code 20, major faults.

See also

Create a routine for the controller fault handler on page 16

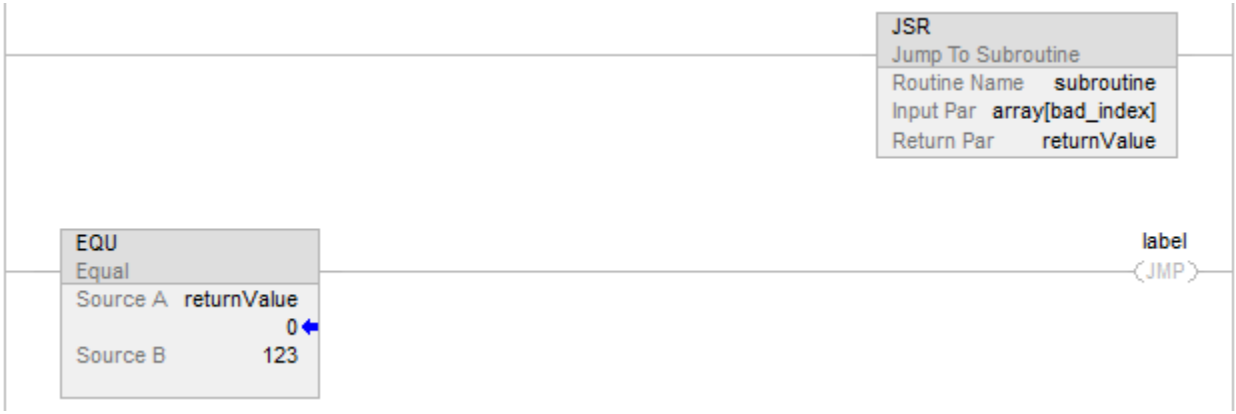
Clear a major fault during prescan on page 22

Recover from a major fault

These examples show fault routines with logic that take specific action after a major fault. If the fault clears, the faulted instruction does not run and execution resumes with the next instruction.

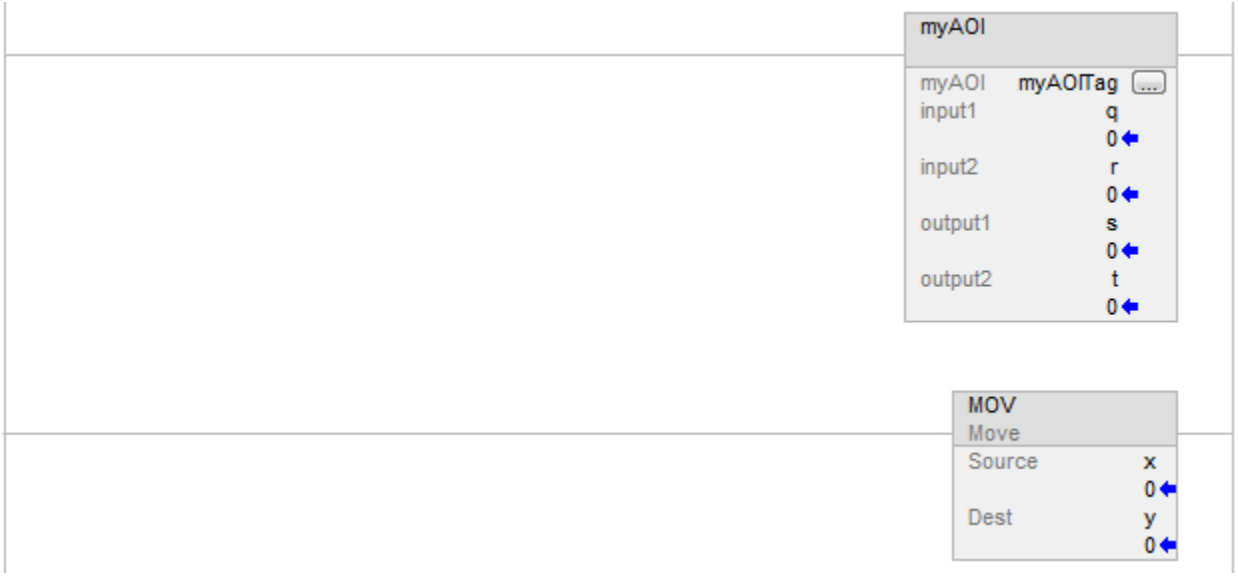
Example 1

In this example, a JSR instruction passes an input parameter containing an indirect address that is out of bounds. If the fault clears, the JSR instruction aborts (the subroutine does not run) and execution resumes with the EQU instruction.



Example 2

In this example, the logic inside an Add-On Instruction generates a fault. While the logic of an Add-On Instruction may look like a subroutine, it is not– the Add-On Instruction is an instruction. When a fault occurs inside an Add-On Instruction, the remainder of the Add-On Instruction aborts. If the fault clears, execution resumes with the MOV instruction.



Important points regarding Add-On Instructions

Keep these considerations in mind when using Add-On Instructions and major faults.

- The Add-On Instruction stops running at the instruction that caused the fault. This means that the remainder of the scan mode routine does not run.
- If the fault clears, execution resumes at the instruction following the top-level Add-On Instruction invocation. For example, assume the Add-On Instruction *myAoi* in Example 2 invokes a nested Add-On Instruction *myNested*, which invokes another nested Add-On Instruction inner. Furthermore, assume that an instruction inside of inner causes a fault. If the fault clears, execution resumes with the MOV instruction (the remainder of inner does not execute; the remainder of *myNested* does not execute; and the remainder of *myAoi* does not execute.)
- During prescan:
  - The Logic routine runs (in prescan mode).
  - The Prescan routine runs (in normal scan mode).
- During postscan:
  - The Logic routine runs (in postscan mode).
  - The Postscan routine runs (in normal scan mode).

If a fault occurs while processing the Logic routine, the Add-On Instruction aborts (the remainder of the Logic routine does not run and the pre-scan and post-scan routines do not run). If the fault clears, execution resumes at the instruction following the top-level Add-On Instruction invocation.

See also

Create a fault routine for a program on page 13

Fault handling during prescan and postscan

The behavior of each instruction varies depending on the mode in which it runs–true, false, prescan, or postscan. For details about what a specific instruction does in each mode, see the Logix 5000 Controllers General Instructions Reference Manual, publication number 1756-RM003.

- Prescan provides a system-defined initialization of the user program when the controller switches from program mode to run mode.
- Postscan provides a system-defined re-initialization of the logic invoked from an SFC action, when the action shuts down (if SFCs are configured for Automatic Reset).

If an array index is out of range during prescan, the controller could generate a major fault. There are a number of ways this could happen: the controller loses power, encounters a major fault, or the project is saved while online. Because the user program, during prescan and postscan, cannot assign values to tags, the only way to correct these issues is to manually initialize the index variables using the Logix Designer application or to write a fault handler to ignore the array faults during prescan. To reduce the need for manual

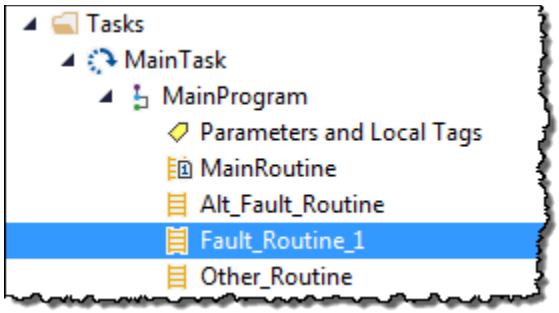
intervention, the Logix Designer application includes an internal fault handler. This handler is only used during prescan and only clears array faults (type 4, fault codes of 20 of 83).

Tip: Array faults are not ignored during postscan because the user program controls index tag values when an action is shut down.

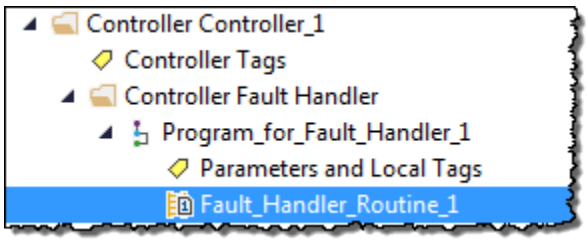
Placement of fault routines

Use a fault routine to program logic to take specific action after a fault, such as clearing the fault and continuing to run. Configure fault routines to a program, controller, or to the Power-Up Handler.

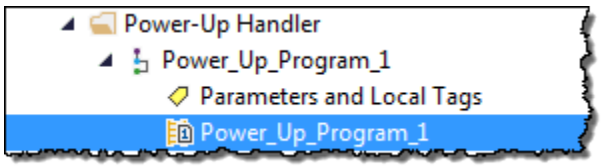
ProgramFaultRoutine



ControllerFaultRoutine



Power-UpFaultHandlerRoutine



See also

Create a fault routine for a program on page 13

Create a routine for the controller fault handler on page 16

Create a routine for the power-up handler on page 17

Choose where to place the fault routine

Where to place the fault routine depends on the type of fault. Use this table to determine where in the project to configure the fault routine.

To clear the fault when		See this section
Condition	**Fault Type**	
The execution of an instruction faults	4	Creating a Fault Routine for a Program
Communication with an I/O module fails	3	Creating a Routine for the Controller Fault Handler
Watchdog timer for a task expires	6	
A motion axis faults	11	
The controller powers up in Run or Remote Run mode	1	Creating a Routine for the Power-Up Handler

See also

Create a fault routine for a program on page 13

Create a routine for the controller fault handler on page 16

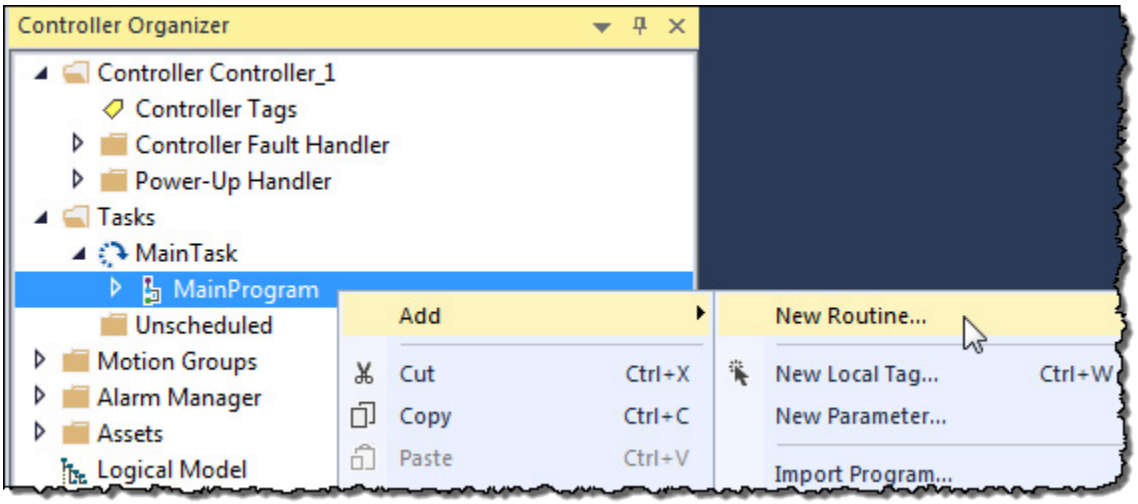
Create a routine for the power-up handler on page 17

Create a fault routine for a program

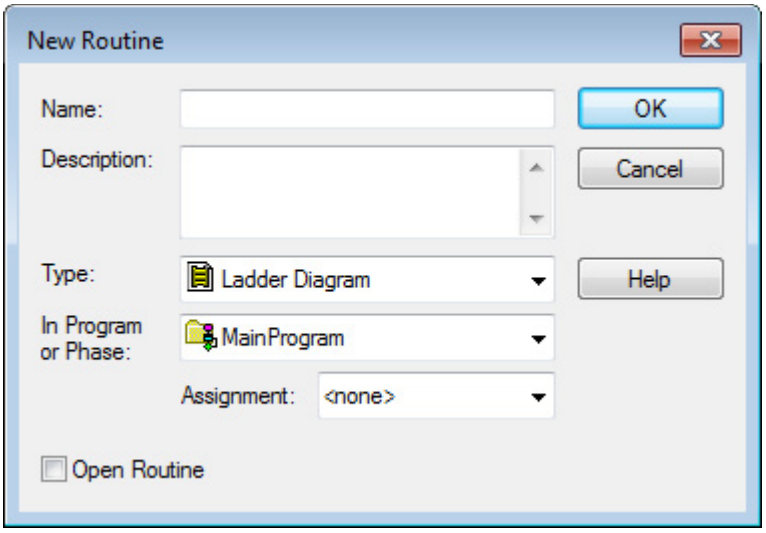
Configure any routine as the fault routine for a program. The routine executes when a program fault occurs before the controller transitions to fault mode.

To create a fault routine for a program:

1. Open the project in the Logix Designer application.
2. In the Controller Organizer, right-click **MainProgram** and select **Add>New Routine**.



3. On the **New Routine** dialog box, in **Name**, type the name of the routine.



4. (optional) In **Description**, type a description of the routine.
5. In **Type**, use the default setting, **Ladder Diagram**.
6. In **In Program or Phase**, use the default setting, **MainProgram**.

   Tip: If creating a fault routine for the Power-Up Handler or Controller Fault Handler, specify the program name of either program in **In Program or Phase**.
7. In **Assignment**, select **Fault**.
8. (optional) Select **Open Routine** to immediately open the ladder logic program.
9. Select **OK**.

See also

[Create a routine for the controller fault handler](#) on page 16

[Create a routine for the power-up handler](#) on page 17

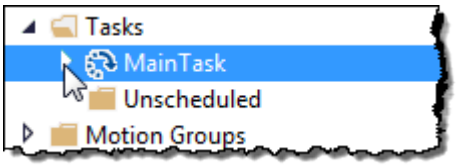
[Choose where to place the fault routine](#) on page 13

Change a fault routine assignment of a program

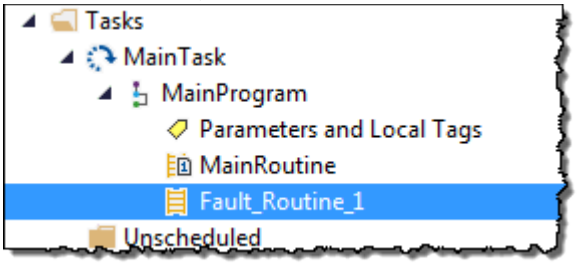
Complete these steps to change the routine assigned as the fault routine.

To change a fault routine assignment of a program:

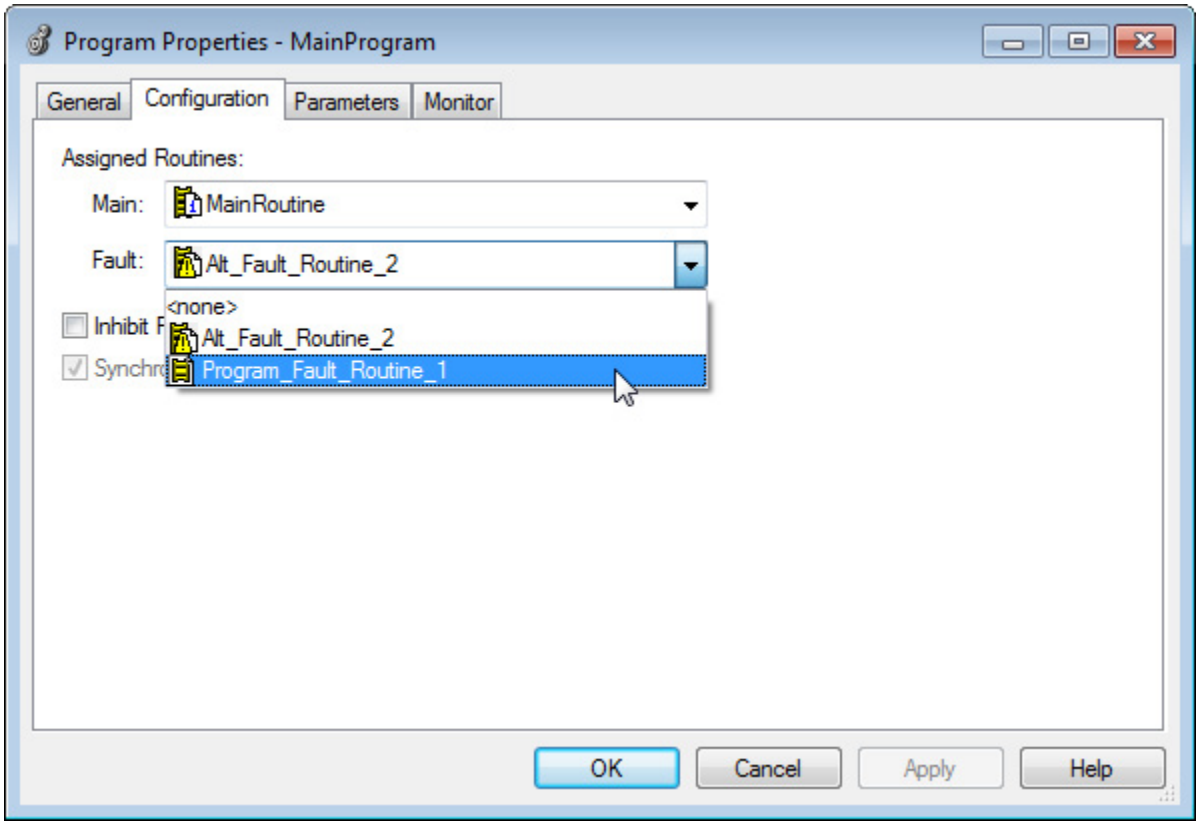
1. In the Controller Organizer, expand the **MainTask**.



If there is already a fault routine, it appears in the **MainProgram**.



2. Right-click **MainProgram** and select **Properties**.
3. On the **Program Properties - MainProgram** dialog box, select the **Configuration** tab.
4. In **Fault**, choose the routine to be the program's fault routine.



5. Select **OK**.

   The program specified in step 4 is now indicated as the fault routine in the **MainProgram**.

See also

Create a fault routine for a program on page 13

Create a routine for the controller fault handler

Use these steps to create a fault routine to operate as the controller fault handler. Program tags are automatically created during this process.

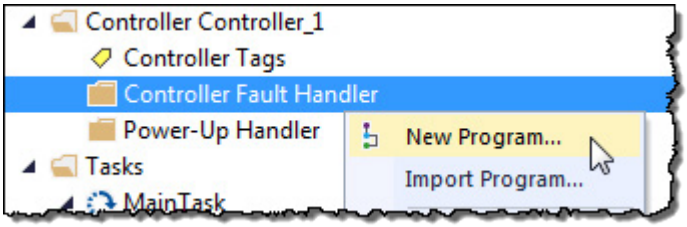
IMPORTANT When programming the fault handler, remember that any instruction that is skipped as part of the fault-handling program does not run when the main tasks and associated programs run.

For example, if the fault handler skips a JSR instruction that is causing a major fault, then that JSR instruction, including all of the programming within the subroutine, does not run.

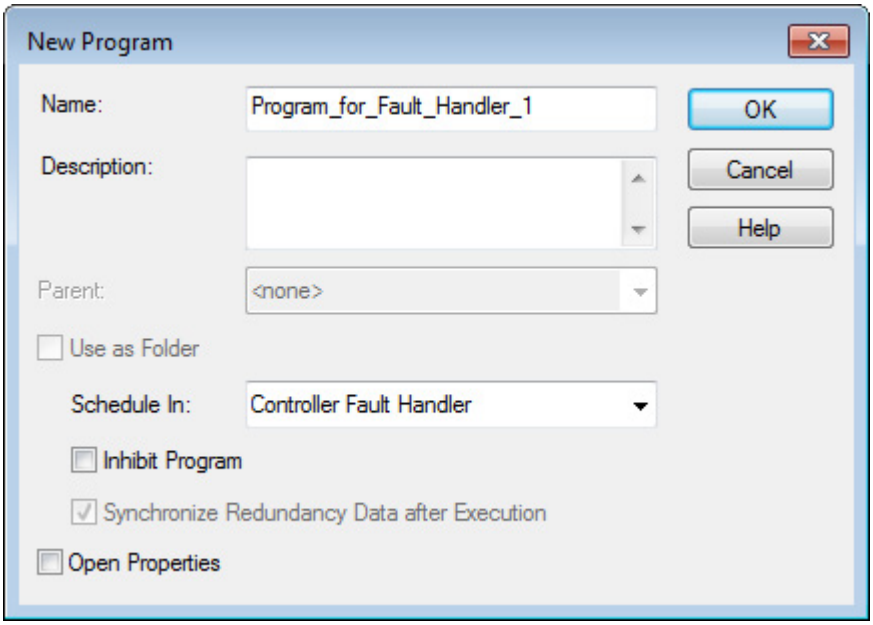
When an instruction generates an error due to a fault (for example, a COP with an indirect addressing programming error), the instruction is skipped and does not run. This occurs with all instructions.

To create a routine for the controller fault handler:

1. In the Controller Organizer, right-click **Controller Fault Handler** and select **New Program**.

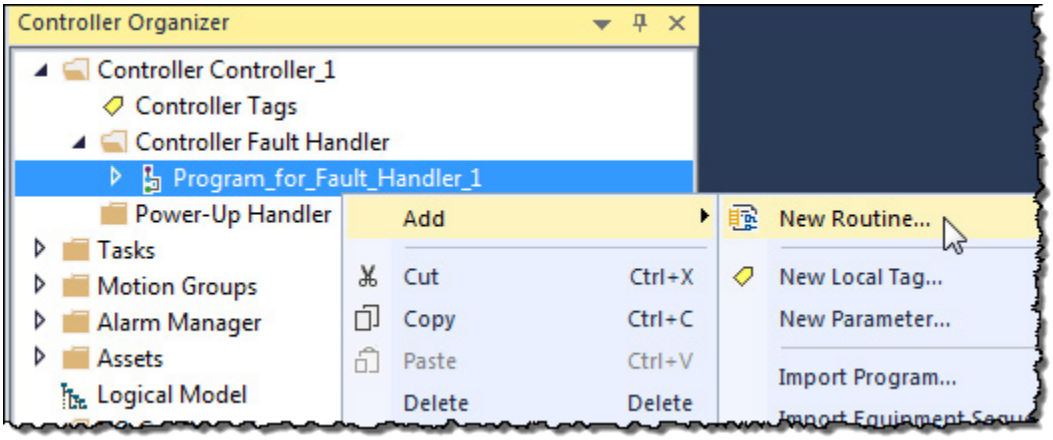


2. On the **New Program** dialog box, in **Name**, type a program name. Verify that **Schedule in** is set to **Controller Fault Handler**.



3. Select **OK**.

4. In the Controller Organizer, right-click the program created in step 2 and select **Add**>**New Routine**.

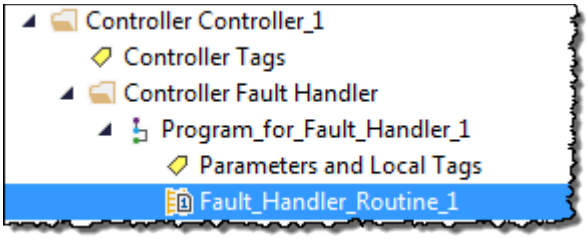


5. On the **New Routine** dialog box, in **Name**, type a name for the routine.
6. In **Type**, choose the type of routine to create. The default is Ladder Diagram.
7. In **Assignment**, use the default setting, **Main**.

   Tip: Even though **Fault** is an option in the **Assignment**, assigning the routine as a fault routine within the Controller Fault Handler is not necessary.

8. Select **OK**.

   The fault routine is created in the **Controller Fault Handler** program.



9. Double-click the fault routine to edit it.

See also

Recover from a major fault on page 9

Fault handling during prescan and postscan on page 11

Create a routine for the power-up handler

The Power-Up Handler is an optional task that executes when the controller powers up in Run or Remote Run modes.

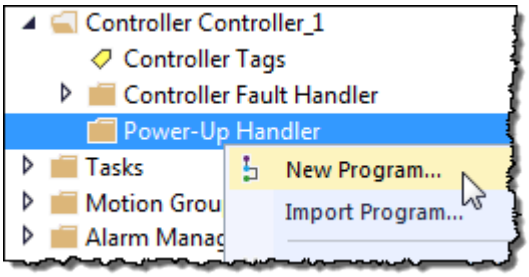
To	Do this
Prevent the controller from returning to Run or Remote mode	Leave the routine for the Power-Up Handler empty. When power restored, a major fault (type 1, code 1) occurs and the controller enters the faulted state.
Direct the controller to take specific actions, then resume normal operation when power restored	In the Power-Up Handler fault routine, complete these steps. 1. Clear the major fault (type 1, code 1). 2. Run the appropriate logic for the specific actions required.

IMPORTANT Do not use fault routines to continually clear all faults on the controller. Program the fault routine to be selective in the types and number of faults cleared.

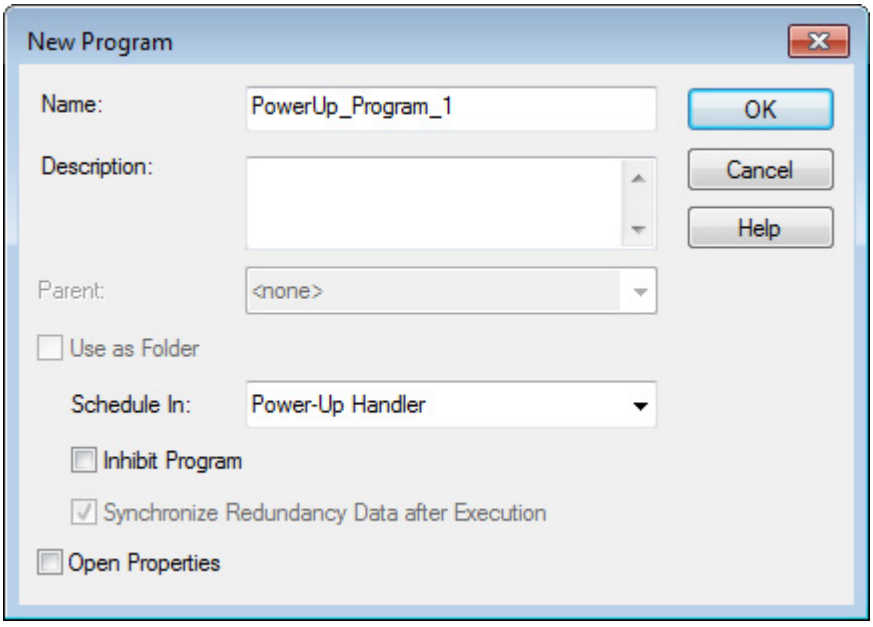
IMPORTANT When an instruction generates an error due to a fault (for example, a COP with an indirect addressing programming error), the routine skips the instruction and the instruction does not run. This occurs with all instructions.

To create a routine for the power-up handler:

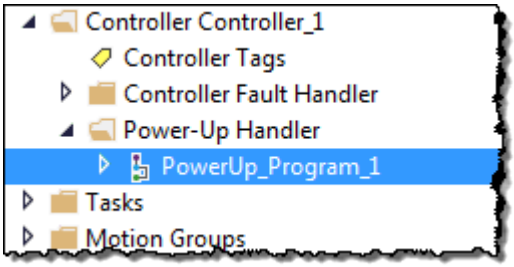
1. In the Controller Organizer, right-click **Power-Up Handler** and select **New Program**.



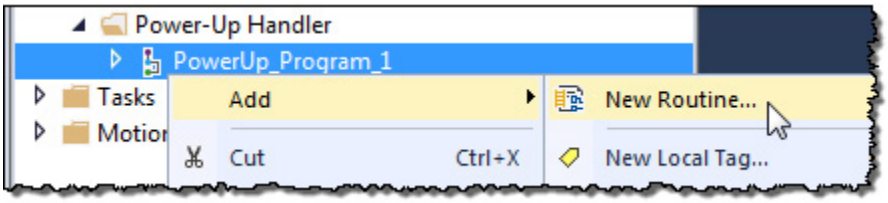
2. On the **New Program** dialog box, in **Name**, type a program name.



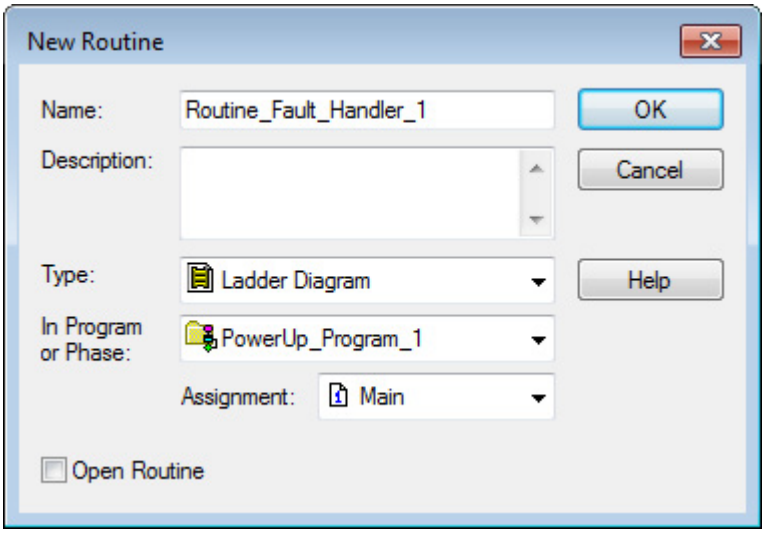
3. Select **OK**. The program is added to the Power-Up Handler.



4. Right-click the program you created in step 2 and click **Add>New Routine.**



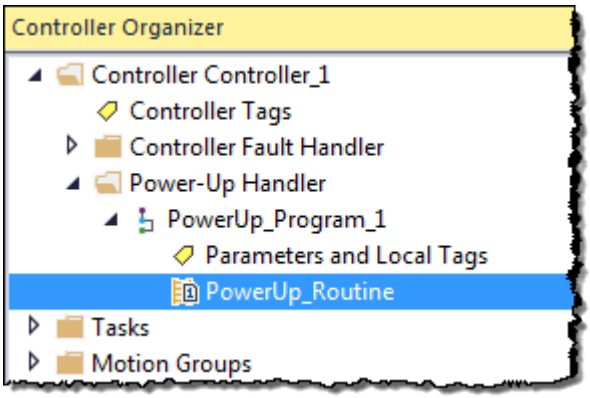
5. On the **New Routine** dialog box, in **Name**, type the name of the routine.



6. In **Assignment**, keep the default setting, **Main**.

Tip: Even though **Fault** is an option in **Assignment**, assigning the routine as a fault routine within the Power-Up Handler is not necessary.

7. Click **OK**. The fault routine is added to the **Power-Up Handler**.



8. Double-click new routine to edit.

See also

Major fault codes on page 25

Programmatically clearing a major fault

To programmatically clear a major fault that occurs during the execution of the project:

- Create a data type to store fault information
- Write a fault routine to clear the fault

IMPORTANT Do not use fault routines to continually clear all faults on the controller. Program the fault routine to be selective in the types and number of faults cleared.

IMPORTANT When an instruction generates an error due to a fault (for example, a COP with an indirect addressing programming error), the routine skips the instruction and the instruction does not run. This occurs with all instructions.

See also

Create a data type to store fault information on page 20

Write a routine to clear the fault on page 21

Create a data type to store fault information

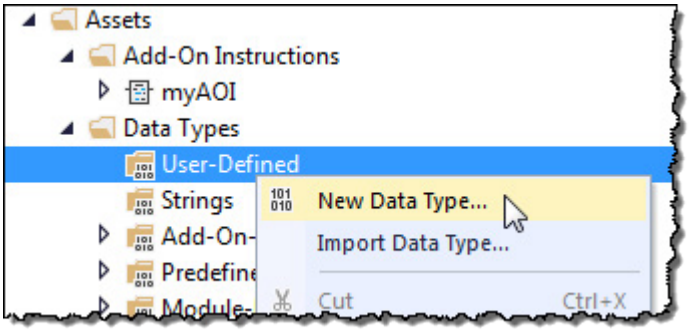
Logix 5000 controllers store system information in objects. Unlike PLC-5 or SLC 500 controllers, there is no status file.

- To access system information, use a Get System Value (GSV) or Set System Value (SSV) instruction.
- To get status information about a program, access the Program object.
- To get fault information for the program, access the MajorFaultRecord attribute of the Program object.

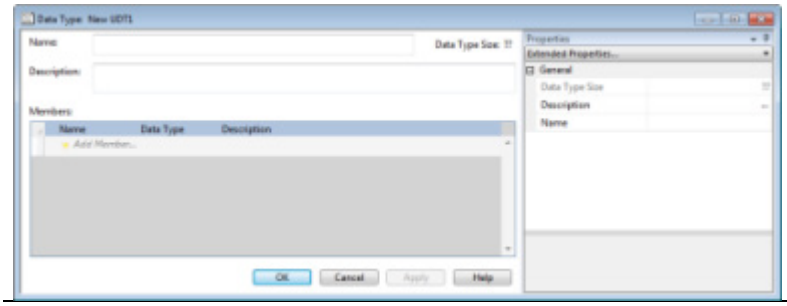
Attribute	Data Type	Instruction	Description
MajorFaultRecord	DINT[11]	GSV SSV	Records major faults for this program. Specifies the program name to determine which Program object to use, or specifies THIS to access the Program object for the program that contains the GSV or SSV instruction.

To simplify access to the MajorFaultRecord attribute, complete these steps to create a user-defined data type.

1. In the Controller Organizer, right-click **User-Defined** and select **New Data Type**.



2. On the **New Data Type** window, enter the data type information as shown in the table.



Data Type: FAULTRECORD

Name	FAULTRECORD
Description	Stores the MajorFaultRecord attribute or MinorFaultRecord attribute of the Program object.

Members

Name	Data Type	Style	Description
Time_Low	DINT	Decimal	Lower 32 bits of the fault timestamp value
Time_High	DINT	Decimal	Upper 32 bits of the fault timestamp value
Type	INT	Decimal	Fault type (program, I/O, and so forth)
Code	INT	Decimal	Unique code for the fault
Info	DINT[8]	Hex	Fault specific information

3. Select **OK**.

See also

Major fault codes on page 25

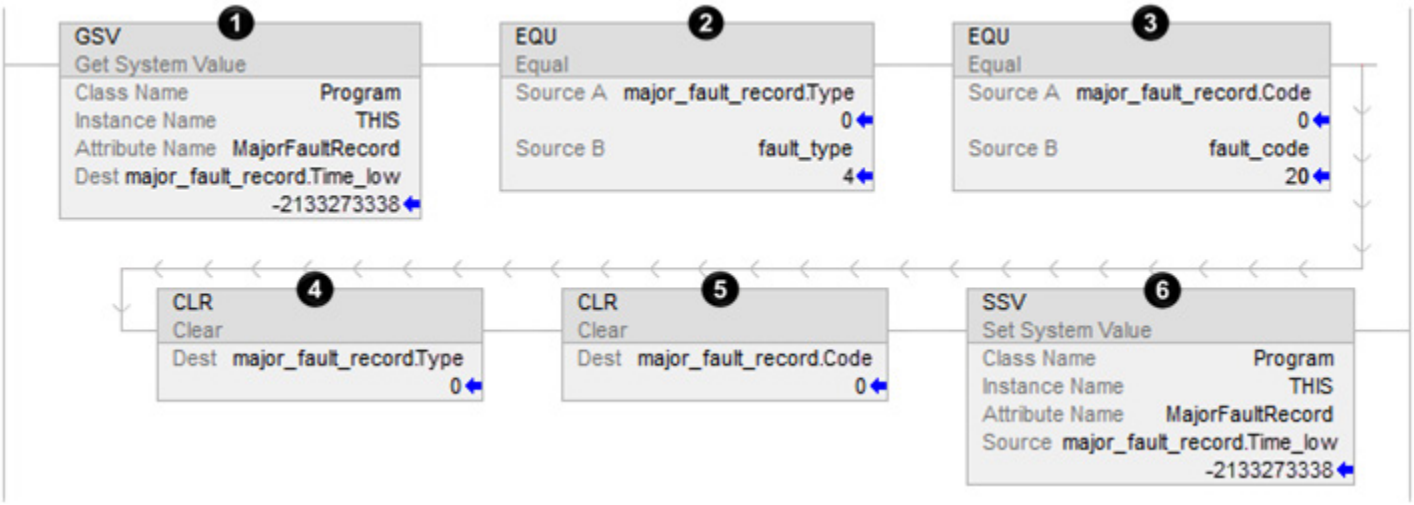
Minor fault codes on page 29

Write a routine to clear the fault

A fault routine normally contains logic to identify the program fault. Some fault routines also contain logic to clear the fault. If a fault clears, the routine continues executing at the instruction immediately after the instruction that

caused the program fault, and the controller does not enter fault mode. If a fault routine does not clear the fault, the controller invokes the Controller Fault Handler program.

Use this example to write a fault routine to clear a major fault.



Item	Reason	Description
1	Gets the fault type and code	The GSV instruction: • Accesses the MajorFaultRecord attribute of this program. This attribute stores information about the fault. • Stores the fault information in the major_fault_record (of type FAULTRECORD) tag. When the tag is based on a structure, enter the first member of the tag.
2	Checks for a specific fault.	The first EQU instruction checks for a specific type of fault, such as program, I/O. In Source B, enter the value for the type of fault that you want to clear.
3		The second EQU instruction checks for a specific fault code. In Source B, enter the value for the code that you want to clear.
4	Sets the fault code and fault type to zero	The first CLR instruction sets the value of the fault type in the major_fault_record tag to zero.
5		Add the second CLR instruction sets the value of the fault code in major_fault_record tag to zero.
6	Clears the fault	The SSV instruction writes: • The new values to the MajorFaultRecord attribute of this program. • The values contained in the major_fault_record tag. Because the Type and Code member are set to zero, the fault clears and the controller resumes execution.

See also

Create a data type to store fault information on page 20

Clear a major fault during prescan

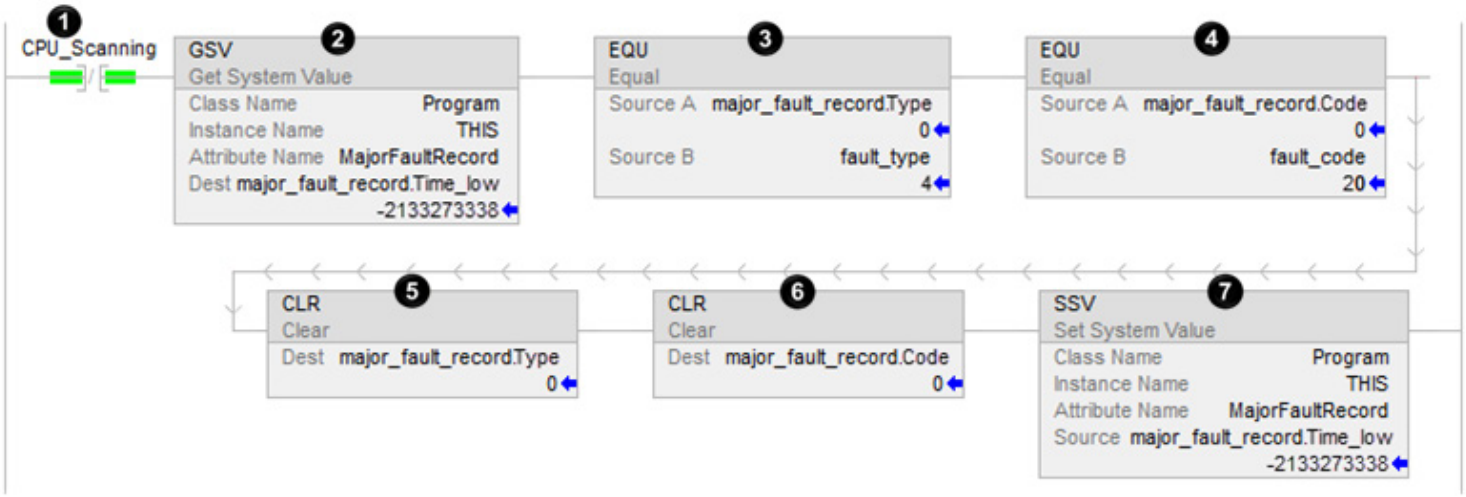
If the controller faults immediately after it switches to Run mode, examine the prescan operation for the fault. Depending on the revision of the controller, an array subscript that is beyond the range of the array (out of range) during prescan might cause a fault.

If controller is revision	Then
11.x or earlier	During prescan, an array subscript that is beyond the range of the array (out of range) produces a major fault.
12.x	See the release notes for the firmware of your controller.

If controller is revision	Then
13.0 or later	During prescan, the controller automatically clears any faults due to an array subscript that is beyond the range of the array (out of range).

This example shows a fault routine that clears a major fault that occurs during prescan.

IMPORTANT It is good programming practice to check for a specific fault before clearing that fault.



Item	Reason	Description
1	Identifies when the controller is in prescan.	The program's fault routine uses the status of this bit to determine if the fault occurred during prescan or normal scan of the logic. • During prescan, this bit is off. During prescan, the controller resets all bits referenced by OTE instructions. • When the controller begins to run the logic, the CPU_scanning bit is always on.
2	Gets the fault type and code	The GSV instruction does the following: • Accesses the program's MajorFaultRecord attribute. This attribute stores information about the fault. • Stores the fault information in the major_fault_record (of type FAULTRECORD) tag. When entering a tag that is based on a structure, enter the first member of the tag.
3	Checks for a specific fault	The first EQU instruction checks for a fault of Type 4, which means that an instruction in this program caused the fault.
4		The second EQU instruction checks for a fault of Code 20, which means that either an array subscript is too large, or a POS or LEN value of a CONTROL structure is invalid.
5		The first CLR instruction sets the value of the fault type in the major_fault_record tag to zero.
6		The second CLR instruction sets the value of the fault type in the major_fault_record tag to zero.
7	Clears the fault	The SSV instruction does the following: • Writes the new values to the program's MajorFaultRecord attribute. • Writes the values contained in the major_fault_record tag. Because the Type and Code member are set to zero, the fault clears and the logix starts running again.

See also

Fault handling during prescan and postscan on page 11

Test a fault routine

Use a JSR instruction to test a program's fault routine without creating an error (simulate a fault).

To test a fault routine:

1. Create a BOOL tag to initiate the fault.
2. In the main routine or a subroutine of the program, enter this rung, where:
   - test_fault_routine is the tag to initiate the fault.
   - Fault_Routine is the fault routine of the program.

   When test_fault_routine is on, a major fault occurs and the controller executes Fault_Routine.

See also

Create a user-defined major fault on page 24

Create a user-defined major fault

To suspend (shut down) the controller based on conditions in the application, create a user-defined major fault. With a user-defined major fault:

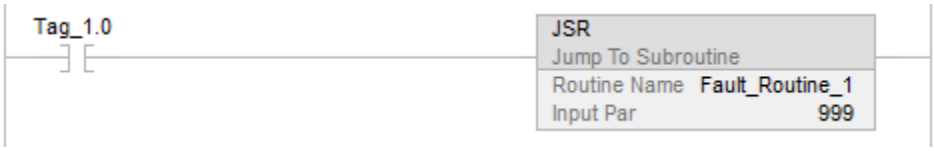
- The fault type = 4.
- Define a value for the fault code. Choose a value between 990 and 999. Logix Designer reserves these codes for user-defined faults.
- The controller handles the fault the same as other major faults:
  - The controller changes to the Program mode and stops executing the logic.
  - Sets the outputs to their configured state or value for faulted mode.

Example: When Tag_1.0 = 1, produce a major fault and generate a fault code of 999.

To create a user-defined major fault:

1. Create a fault routine for the program if one does not exist.
2. Configure the program to use the fault routine if it is not already assigned.
3. In the main routine of the program, enter this rung, where:
   - Tag_1.0 is the tag used to initiate the fault

- Fault_Routine_1 is the fault routine of the program
- 999 is the value of the fault code



4. When the major fault occurs, the controller enters faulted mode. Outputs go to the faulted state. The **Major Faults** tab in the **Controller Properties** dialog box displays code 999.

See also

Create a fault routine for a program on page 13

Change a fault routine assignment of a program on page 14

Major fault codes on page 25

Major fault codes

The type and code correspond to the type and code displayed in these locations.

- **Controller Properties** dialog box, **Major Faults** tab
- Program object, MajorFaultRecord attribute

Refer to the Logix 5000 Controller Fault Codes spreadsheet for a complete list of fault codes.

You might be asked to log in to your Rockwell Automation web account or create an account if you do not have one. You do not need a support contract to access the article.

See also

Minor fault codes on page 9

I/O fault codes on page 33

Minor Faults

This chapter explains minor fault codes and how to work with them in the Logix Designer application.

Identify minor faults

Use this table to understand how to use ladder logic to monitor information about common minor faults.

To check for a	Do this
Task overlap	• Enter a GSV instruction that gets the FaultLog object, MinorFaultBits attribute. • Monitor bit 6.
Load from nonvolatile memory	• Enter a GSV instruction that gets the FaultLog object, MinorFaultBits attribute. • Monitor bit 7.
Serial port fault	• Enter a GSV instruction that gets the FaultLog object, MinorFaultBits attribute. • Monitor bit 9.
Low battery, energy storage status or uninterruptable power supply (UPS) fault	• Enter a GSV instruction that gets the FaultLog object, MinorFaultBits attribute. • Monitor bit 10.
Instruction-related fault	1. Create a user-defined data type that stores the fault information. Name the data type FaultRecord and assign the following members. **Name** / **Data Type** / **Style** TimeLow / DINT / Decimal TimeHigh / DINT / Decimal Type / INT / Decimal Code / INT / Decimal Info / DINT[8] / Hex 1. Create a tag that stores the values of the MinorFaultRecord attribute. 2. From the Data Type menu in step 1 of this instruction, choose the data type. 3. Monitor S:MINOR. 4. Use a GSV instruction to get the values of the MinorFaultRecord attribute if S:MINOR is on. 5. Reset S:MINOR if you want to detect a minor fault that is cause by another instruction. S:MINOR remains set until the end of the scan.

See also

Minor fault codes on page 29

Minor fault examples

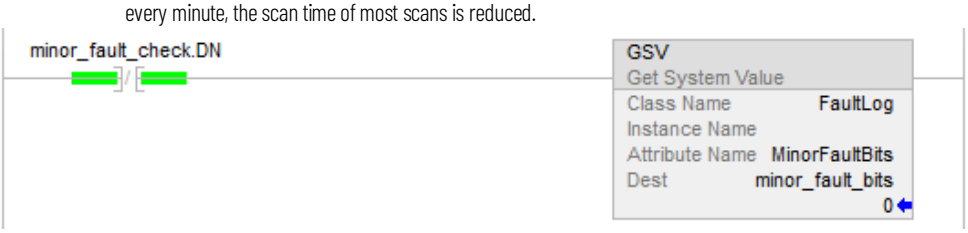
Use these examples to check for minor faults.

Checks for a low battery warning

Example: Checks for a minor fault.

Minor_fault_check times for 1 minute (60000 ms) and then automatically restarts itself.

Every minute, minor_fault_check.DN turns on for one scan. When this occurs, the GSV instruction gets the value of the FaultLog object, MinorFaultBits attribute, and stores it in the minor_fault_bits tag. Because the GSV instruction only runs once every minute, the scan time of most scans is reduced.

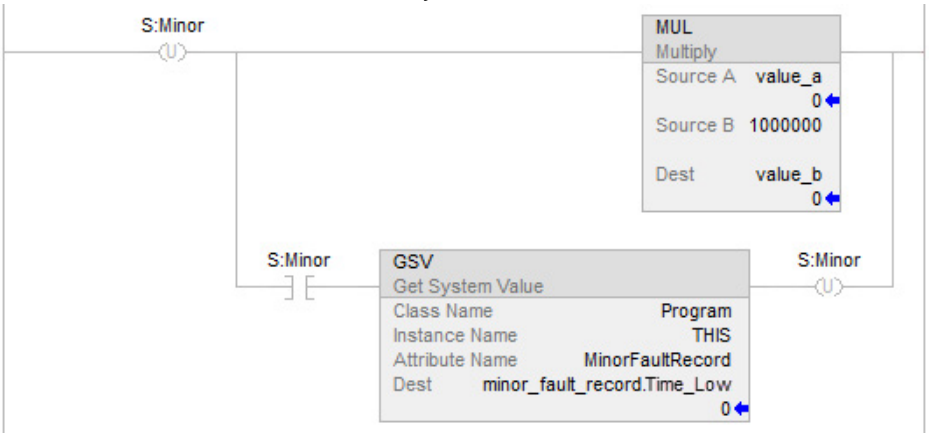


If minor_fault_bits.10 is on, depending on the controller, the battery is low or the ESM or UPS needs to be replaced or is missing.

Checks for a minor fault that is caused by a specific instruction

Example: Check for a minor fault that is caused by an instruction.

- Multiply value_a by 1000000 and check for a minor fault, such as a math overflow.
- To make sure that a previous instruction did not produce the fault, the rung first clears S:MINOR.
- The rung then executes the multiply instruction.
- If the instruction produces a minor fault, the controller sets S:MINOR.
- If S:MINOR is set, the GSV instruction gets information about the fault and resets S:MINOR.



See also

Create a data type to store fault information on page 20

Minor fault codes

Minor faults get recorded in these locations.

- **Controller Properties** dialog box, **Minor Faults** tab
- Program object, MinorFaultRecord attribute

Refer to the Logix 5000 Controller Fault Codes spreadsheet for a complete list of fault codes.

You might be asked to log in to your Rockwell Automation web account or create an account if you do not have one. You do not need a support contract to access the article.

See also

Major fault codes on page 25

I/O fault codes on page 33

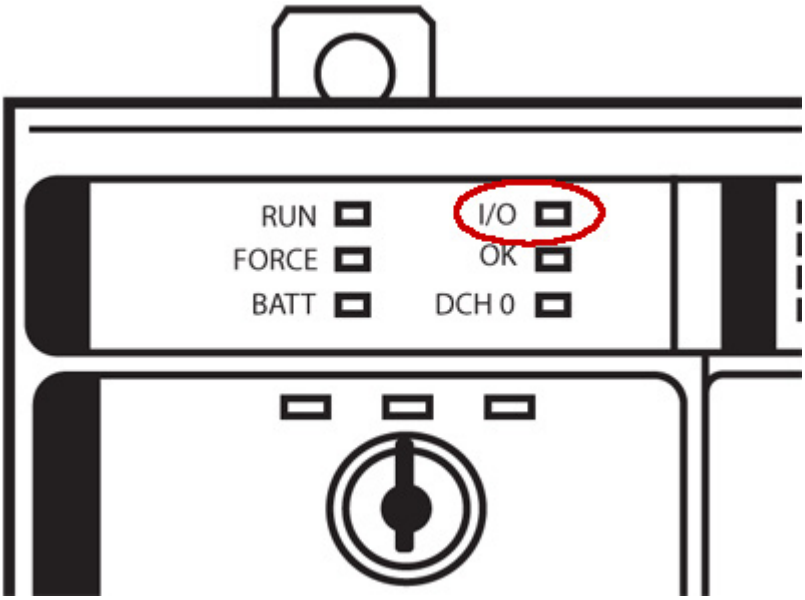
I/O Fault Codes

This chapter explains I/O fault codes and how to work with them in the Logix Designer application.

Indications of I/O faults

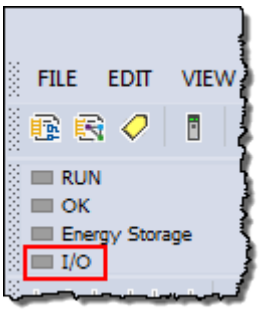
The indication of I/O faults displays in various ways depending on the controller.

- The I/O indicator of the controller (shown in examples below) flashes green or red.

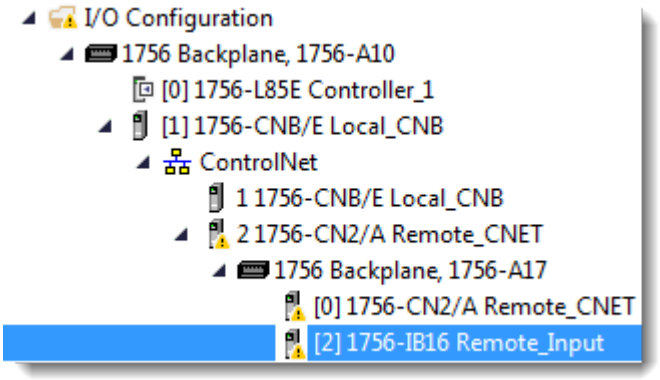


- The controller status display indicates I/O fault messages.

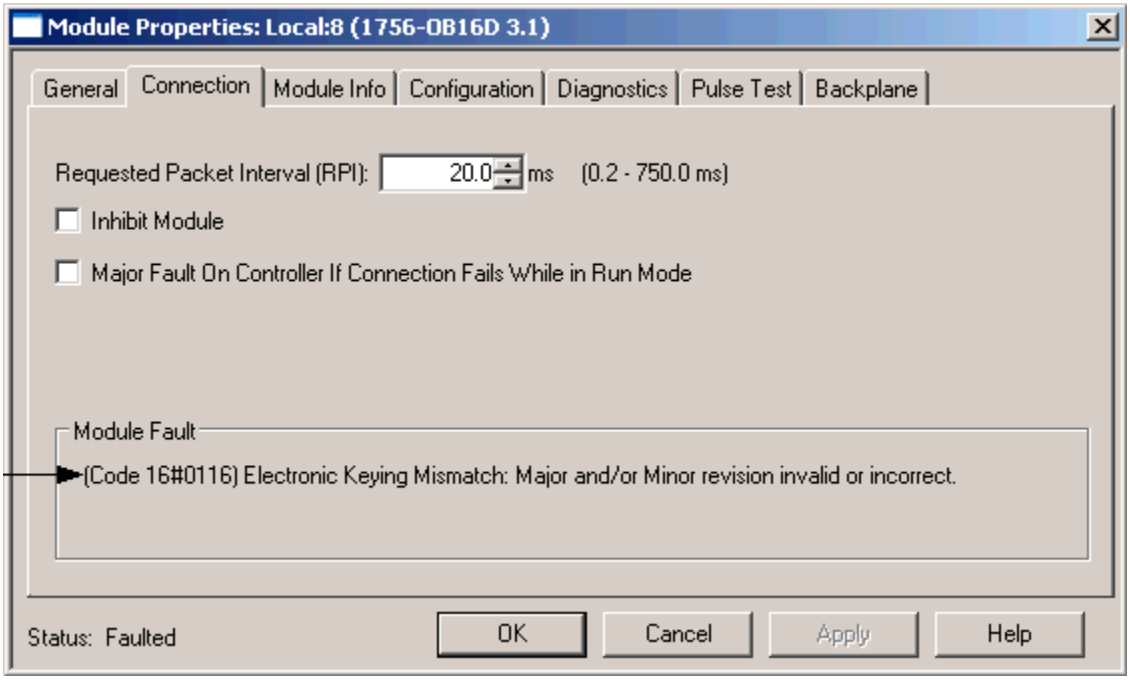
- The I/O status indicator and messages show in the controller status area of the Logix Designer application. The indicator flashes green or red and the corresponding status message indicates an error.



- A yellow warning symbol appears on the module in the I/O Configuration tree of the Logix Designer application.



- A module fault code and description appear in the **Connection** tab of the **Module Properties** dialog box.



I/O Fault Codes

Depending where the fault code displays, the code format contains either the full Hexadecimal number (for example, 16#000A) or the last characters of the code (for example, #000A).

Refer to the Logix 5000 Controller Fault Codes spreadsheet for a complete list of fault codes.

You might be asked to log in to your Rockwell Automation web account or create an account if you do not have one. You do not need a support contract to access the article.

See also

Major fault codes on page 25

Minor fault codes on page 29

Index

T

U

W

Rockwell Automation support

Use these resources to access support information.

Technical Support Center	Find help with how-to videos, FAQs, chat, user forums, and product notification updates.	rok.auto/support
Knowledgebase	Access Knowledgebase articles.	rok.auto/knowledgebase
Local Technical Support Phone Numbers	Locate the telephone number for your country.	rok.auto/phonesupport
Literature Library	Find installation instructions, manuals, brochures, and technical data publications.	rok.auto/literature
Product Compatibility and Download Center (PCDC)	Get help determining how products interact, check features and capabilities, and find associated firmware.	rok.auto/pcdc

Documentation feedback

Your comments help us serve your documentation needs better. If you have any suggestions on how to improve our content, complete the form at rok.auto/docfeedback.

Waste Electrical and Electronic Equipment (WEEE)

At the end of life, this equipment should be collected separately from any unsorted municipal waste.

Rockwell Automation maintains current product environmental information on its website at rok.auto/pec.

Allen-Bradley, expanding human possibility, Logix, Rockwell Automation, and Rockwell Software are trademarks of Rockwell Automation, Inc.

EtherNet/IP is a trademark of ODVA, Inc.

Trademarks not belonging to Rockwell Automation are property of their respective companies.

Rockwell Otomayson Ticaret A.Ş. Kar Plaza İş Merkezi E Blok Kat:6 34752, İçerenkÖy, İstanbul, Tel: +90 (216) 5698400 EEE YÖnetmeliğine Uygundur

Connect with us.

AMERICAS: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444
EUROPE/MIDDLE EAST/AFRICA: Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640
ASIA PACIFIC: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Rockwell Automation Publication 1756-PM014N-EN-P - March 2022
Supersedes Publication 1756-PM014M-EN-P - September 2020

Copyright © 2022 Rockwell Automation Technologies, Inc. All Rights Reserved. Printed in the U.S.A.