Logix5000 Controllers I/O and Tag Data

Programming Manual

Catalog Numbers 1756 ControlLogix, 1769 CompactLogix, 1789 SoftLogix, 1794 FlexLogix, PowerFlex 700S with DriveLogix

ALLEN-BRADLEY • ROCKWELL SOFTWARE **Rockwell Automation**

Important User Information

Solid state equipment has operational characteristics differing from those of electromechanical equipment. Safety Guidelines for the Application, Installation and Maintenance of Solid State Controls (publication SGI-1.1 available from your local Rockwell Automation sales office or online at http://literature.rockwellautomation.com) describes some important differences between solid state equipment and hard-wired electromechanical devices. Because of this difference, and also because of the wide variety of uses for solid state equipment, all persons responsible for applying this equipment must satisfy themselves that each intended application of this equipment is acceptable.

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

Purpose of this Manual

This manual shows how to access I/O and tag data in Logix5000 controllers. This manual is one of a set of related manuals that show common procedures for programming and operating Logix5000 controllers. For a complete list of common procedures manuals, refer to the Logix 5000 Controllers Common Procedures Programming Manual, publication 1756-PM001.

The term Logix5000 controller refers to any controller that is based on the Logix5000 operating system, such as:

- CompactLogix controllers
- ControlLogix controllers
- DriveLogix controllers
- FlexLogix controllers
- SoftLogix5800 controllers

How to Use this Manual

Some text is formatted differently from the rest of the text.

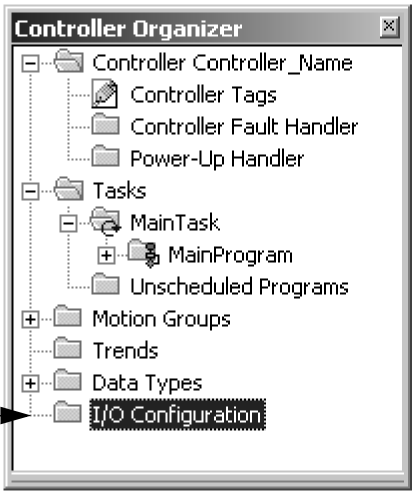
Text that is	Identifies	For example	Means
Italic	the actual name of an item that you see on your screen or in an example	Right-click *User-Defined* …	Right-click the item that is named User-Defined.
courier	information that you must supply based on your application (a variable)	Right-click *name_of_program* …	You must identify the specific program in your application. Typically, it is a name or variable that you have defined.
enclosed in brackets	a keyboard key	Press [Enter].	Press the Enter key.

Notes:

Communicating with I/O

Introduction

To communicate with an I/O module in your system, you add the module to the I/O Configuration folder of the controller.



When you add the module, you also define a specific configuration for the module. While the configuration options vary from module to module, there are some common options that you typically configure:

- Requested Packet Interval
- Communication Format
- Electronic Keying

Requested Packet Interval

The Logix5000 controller uses connections to transmit I/O data.

Term	Definition
Connection	A communication link between two devices, such as between a controller and an I/O module, PanelView terminal, or another controller. Connections are allocations of resources that provide more reliable communications between devices than unconnected messages. The number of connections that a single controller can have is limited. You indirectly determine the number of connections the controller uses by configuring the controller to communicate with other devices in the system. The following types of communication use connections: • I/O modules • produced and consumed tags • certain types of Message (MSG) instructions (not all types use a connection)
Requested packet interval (RPI)	The RPI specifies the period at which data updates over a connection. For example, an input module sends data to a controller at the RPI that you assign to the module. • Typically, you configure an RPI in milliseconds (ms). The range is 0.2 ms (200 microseconds) to 750 ms. • If a ControlNet network connects the devices, the RPI reserves a slot in the stream of data flowing across the ControlNet network. The timing of this slot may not coincide with the exact value of the RPI, but the control system guarantees that the data transfers at least as often as the RPI.

In Logix5000 controllers, I/O values update at a period that you configure via the I/O configuration folder of the project. The values update asynchronous to the execution of logic. At the specified interval, the controller updates a value independently from the execution of logic.

ATTENTION

Take care to ensure that data memory contains the appropriate values throughout a task's execution. You can duplicate or buffer data at the beginning of the scan to provide reference values for your logic.

- Programs within a task access input and output data directly from controller-scoped memory.
- Logic within any task can modify controller-scoped data.
- Data and I/O values are asynchronous and can change during the course of a task's execution.
- An input value referenced at the beginning of a task's execution can be different when referenced later.
- To prevent an input value from changing during a scan, copy the value to another tag and use the data from there (buffer the values).

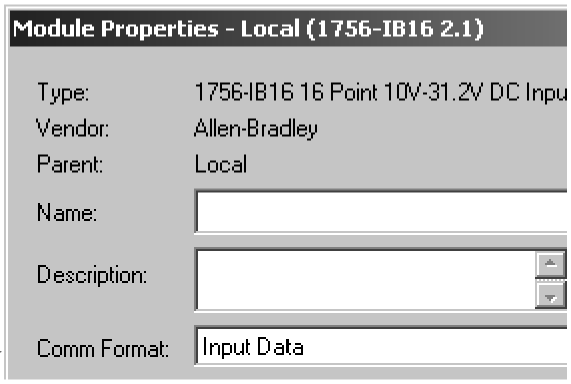
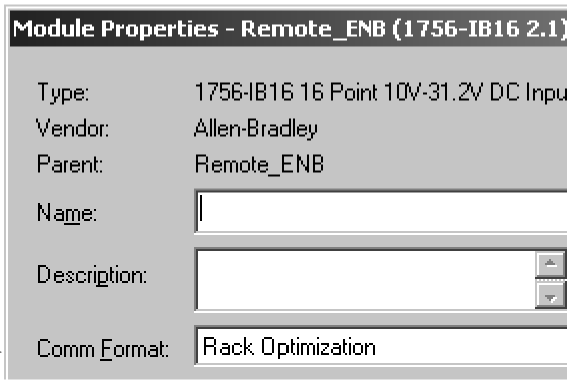
Communication Format

The communication format that you choose determines the data structure for the tags that are associated with the module. Many I/O modules support different formats. Each format uses a different data structure. The communication format that you choose also determines:

- Direct or Rack-Optimized Connection
- Ownership

Direct or Rack-Optimized Connection

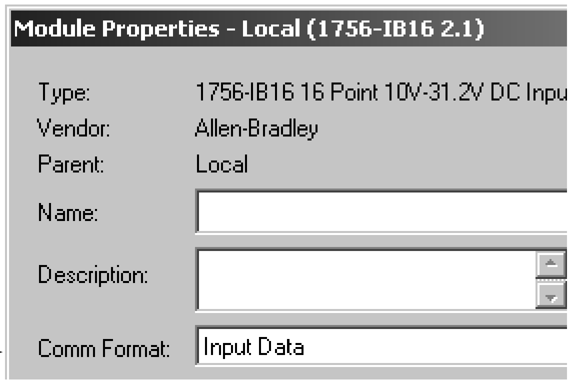
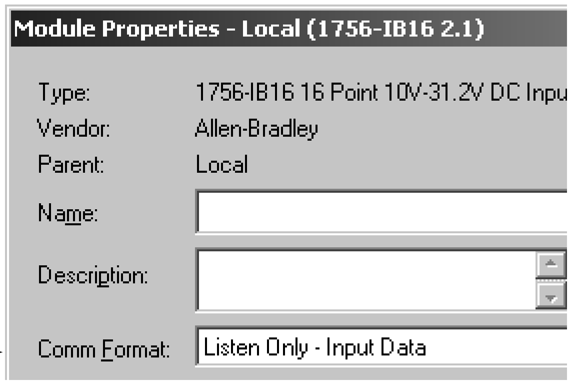
The Logix5000 controller uses connections to transmit I/O data. These connections can be direct connections or rack-optimized connections.

Term	Definition
Direct connection	A direct connection is a real-time, data transfer link between the controller and an I/O module. The controller maintains and monitors the connection with the I/O module. Any break in the connection, such as a module fault or the removal of a module while under power, sets fault bits in the data area associated with the module.  Module Properties - Local (1756-IB16 2.1) Type: 1756-IB16 16 Point 10V-31.2V DC Inpu Vendor: Allen-Bradley Parent: Local Name: Description: Comm Format: Input Data. A direct connection is any connection that does not use the Rack Optimization Come Format.
Rack-optimized connection	For digital I/O modules, you can select rack-optimized communication. A rack-optimized connection consolidates connection usage between the controller and all the digital I/O modules in the chassis (or DIN rail). Rather than having individual, direct connections for each I/O module, there is one connection for the entire chassis (or DIN rail).  Module Properties - Remote_ENB (1756-IB16 2.1) Type: 1756-IB16 16 Point 10V-31.2V DC Inpu Vendor: Allen-Bradley Parent: Remote_ENB Name: Description: Comm Format: Rack Optimization. Rack-optimized connection

Ownership

In a Logix5000 system, modules multicast data. This means that multiple devices can receive the same data at the same time from a single device.

When you choose a communication format, you have to choose whether to establish an owner or listen-only relationship with the module.

Owner controller	The controller that creates the primary configuration and communication connection to a module. The owner controller writes configuration data and can establish a connection to the module. An owner connection is any connection that does not include Listen-Only in its Comm Format.  Module Properties - Local (1756-IB16 2.1) Type: 1756-IB16 16 Point 10V-31.2V DC Inpu Vendor: Allen-Bradley Parent: Local Name: Description: Comm Format: Input Data
Listen-only connection	An I/O connection where another controller owns/provides the configuration data for the I/O module. A controller using a listen-only connection only monitors the module. It does not write configuration data and can only maintain a connection to the I/O module when the owner controller is actively controlling the I/O module. Listen-only connection  Module Properties - Local (1756-IB16 2.1) Type: 1756-IB16 16 Point 10V-31.2V DC Inpu Vendor: Allen-Bradley Parent: Local Name: Description: Comm Format: Listen Only - Input Data

Use the following table to choose the type of ownership for a module:

If the module is an	And another controller	And you want to	Then use this type of connection
Input module	Does not own the module	$\longrightarrow$	Owner (not listen-only)
	Owns the module	Maintain communication with the module if it loses communication with the other controller	Owner (not listen-only) Use the same configuration as the other owner controller.
		Stop communication with the module if it loses communication with the other controller	Listen-only
Output module	Does not own the module	$\longrightarrow$	Owner (i.e., not listen-only)
	Owns the module	$\longrightarrow$	Listen-only

There is a noted difference in controlling input modules versus controlling output modules.

Controlling	This ownership	Description
Input modules	Owner	An input module is configured by a controller that establishes a connection as an owner. This configuring controller is the first controller to establish an owner connection. Once an input module has been configured (and owned by a controller), other controllers can establish owner connections to that module. This lets additional owners to continue to receive multicast data if the original owner controller breaks its connection to the module. All other additional owners must have the identical configuration data and identical communications format that the original owner controller has, otherwise the connection attempt is rejected.
	Listen-only	Once an input module has been configured (and owned by a controller), other controllers can establish a listen-only connection to that module. These controllers can receive multicast data while another controller owns the module. If all owner controllers break their connections to the input module, all controllers with listen-only connections no longer receive multicast data.
Output modules	Owner	An output module is configured by a controller that establishes a connection as an owner. Only one owner connection is allowed for an output module. If another controller attempts to establish an owner connection, the connection attempt is rejected.
	Listen-only	Once an output module has been configured (and owned by one controller), other controllers can establish listen-only connections to that module. These controllers can receive multicast data while another controller owns the module. If the owner controller breaks its connection to the output module, all controllers with listen-only connections no longer receive multicast data.

Electronic Keying

ATTENTION

⚠

Be careful when you disable electronic keying. If used incorrectly, this option can lead to personal injury or death, property damage, or economic loss.

When you configure a module, you specify the slot number for the module. However, it is possible to place a different module in that slot, either on purpose or accidently.

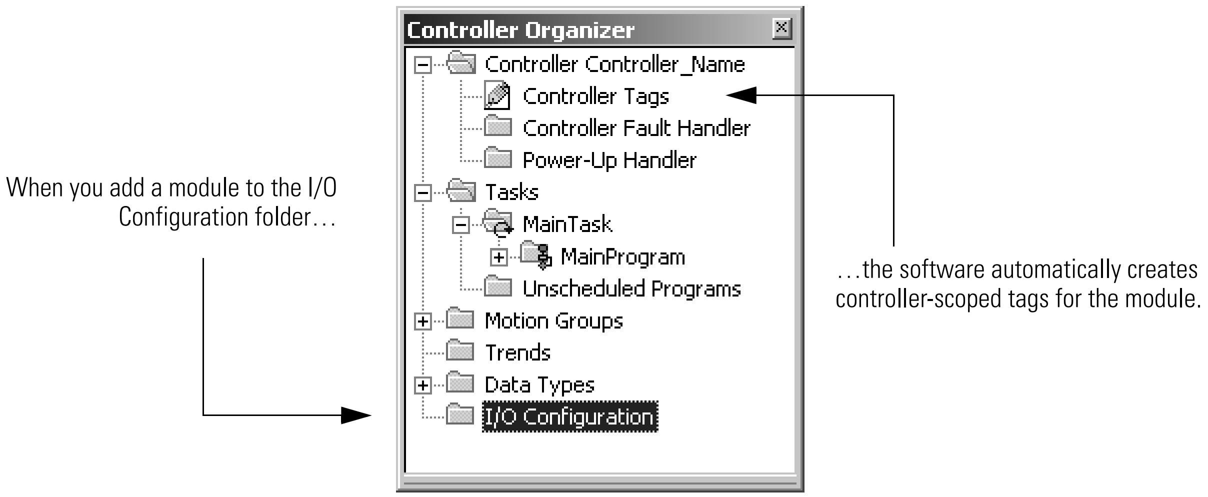
Electronic keying lets you protect your system against the accidental placement of the wrong module in a slot. The keying option you choose determines how closely any module in a slot must match the configuration for that slot.

If	Then Select
All information must match: • type • catalog number • vendor • major and minor revision number	Exact Match
All information except the minor revision number	Compatible Module
No information must match	Disable Keying

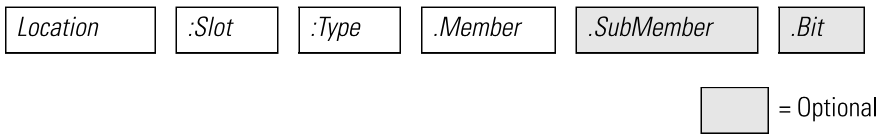
Address I/O Data

I/O information is presented as a set of tags.

- Each tag uses a structure of data. The structure depends on the specific features of the I/O module.
- The name of the tags is based on the location of the I/O module in the system.



An I/O address follows this format:



Where	Is
Location	Network location LOCAL = same chassis or DIN rail as the controller *ADAPTER_NAME* = identifies remote communication adapter or bridge module
Slot	Slot number of I/O module in its chassis or DIN rail

Type	Type of data
	I = input
	O = output
	C = configuration
	S = status

Member	Specific data from the I/O module; depends on what type of data the module can store. • For a digital module, a Data member usually stores the input or output bit values. • For an analog module, a Channel member (CH#) usually stores the data for a channel.
SubMember	Specific data related to a Member.
Bit	Specific point on a digital I/O module; depends on the size of the I/O module (0-31 for a 32-point module)

Buffering I/O

Buffering is a technique in which logic does not directly reference or manipulate the tags of real I/O devices. Instead, the logic uses a copy of the I/O data. Buffer I/O in the following situations:

- To prevent an input or output value from changing during the execution of a program. (I/O updates asynchronous to the execution of logic.)
- To copy an input or output tag to a member of a structure or element of an array.

Buffer I/O

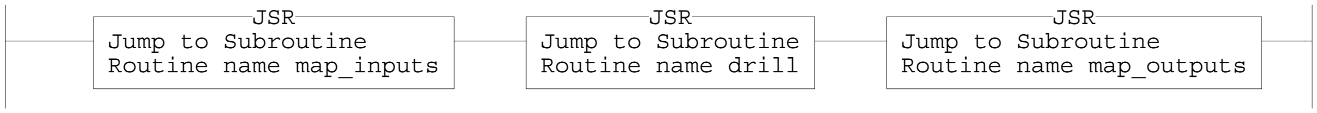
To buffer I/O, perform these actions.

1. On the rung before the logic for the function (s), copy or move the data from the required input tags to their corresponding buffer tags.
2. In the logic of the function (s), reference the buffer tags.
3. On the rung after the function (s), copy the data from the buffer tags to the corresponding output tags.

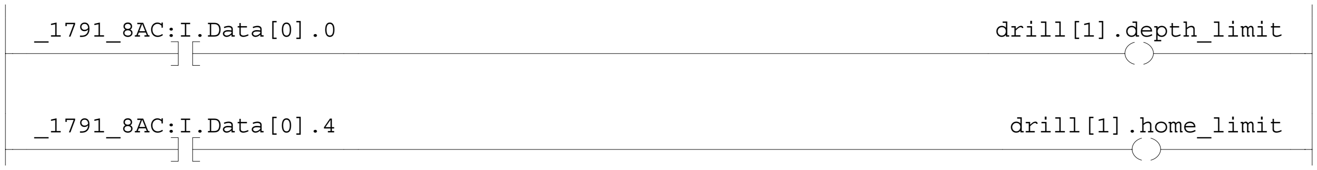
This example copies inputs and outputs to the tags of a structure for a drill machine.

EXAMPLE Buffer I/O

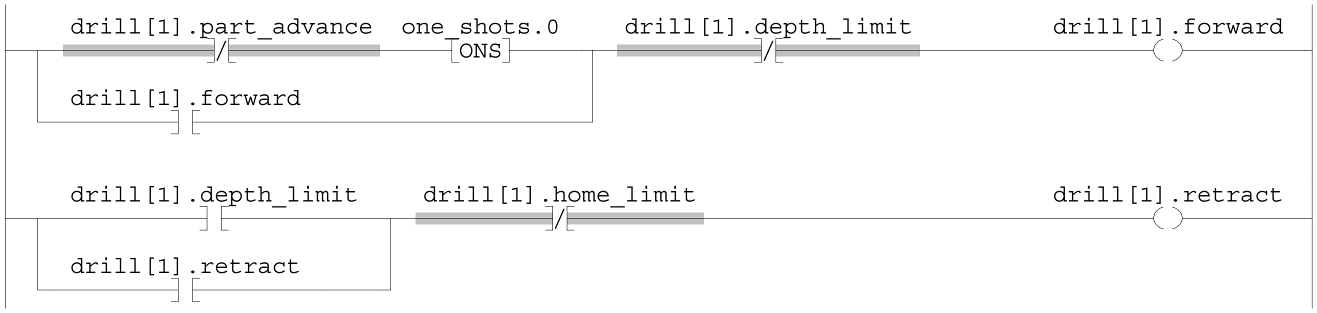
The main routine of the program executes the following subroutines in this sequence.



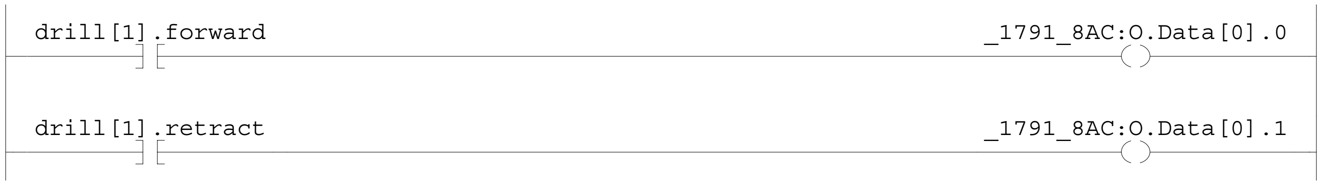
The map_inputs routine copies the values of input devices to their corresponding tags that are used in the drill routine.



The drill routine executes the logic for the drill machine.



The map_outputs routine copies the values of output tags in the drill routine to their corresponding output devices.



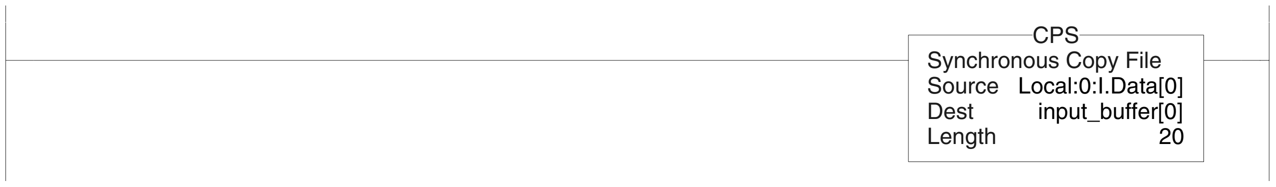
42369

This example uses the CPS instruction to copy an array of data that represent the input devices of a DeviceNet network.

EXAMPLE Buffer I/O

Local:0:I.Data stores the input data for the DeviceNet network that is connected to the 1756-DNB module in slot 0. To synchronize the inputs with the application, the CPS instruction copies the input data to input_buffer.

- While the CPS instruction copies the data, no I/O updates can change the data.
- As the application executes, it uses for its inputs the input data in input_buffer.



42578

Organizing Tags

Introduction

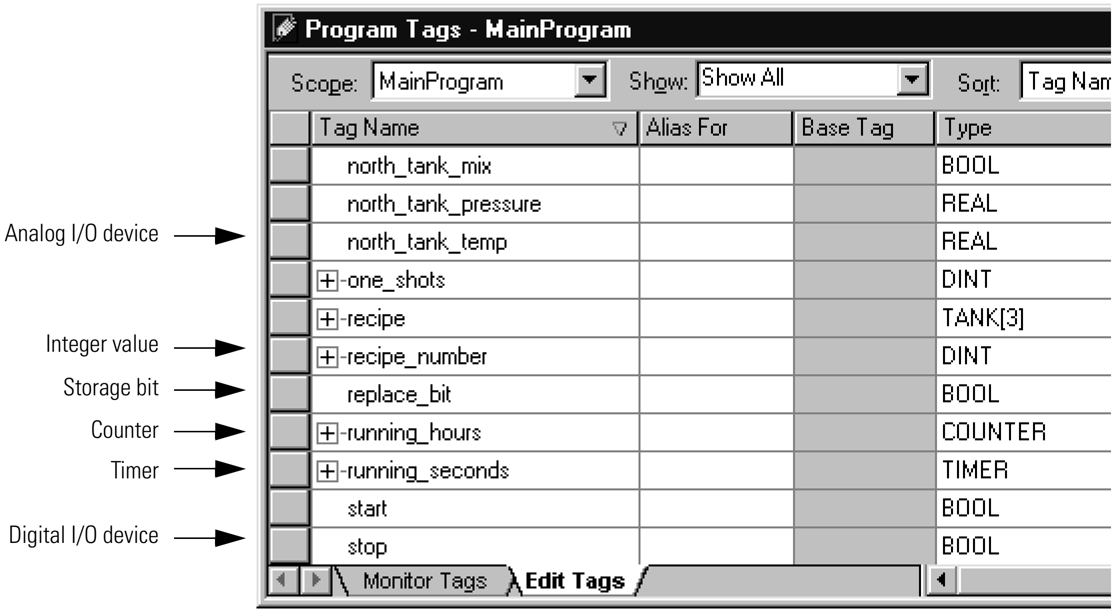
With a Logix5000 controller, you use a tag (alphanumeric name) to address data (variables).

Term	Definition
Tag	A text-based name for an area of the controller's memory where data is stored. • Tags are the basic mechanism for allocating memory, referencing data from logic, and monitoring data. • The minimum memory allocation for a tag is four bytes. • When you create a tag that stores data that requires less than four bytes, the controller allocates four bytes, but the data only fills the part it needs.

The controller uses the tag name internally and doesn't need to cross-reference a physical address.

- In conventional programmable controllers, a physical address identifies each item of data.
  - Addresses follow a fixed, numeric format that depend on the type of data, such as N7:8, F8:3.
  - Symbols are required to make logic easier to interpret.
- In Logix5000 controllers, there is no fixed, numeric format. The tag name itself identifies the data. This lets you:
  - organize your data to mirror your machinery
  - document (through tag names) your application as you develop it

EXAMPLE Tags



Scope: MainProgram Show: Show All Sort: Tag Nam

Tag Name	Alias For	Base Tag	Type
north_tank_mix			BOOL
north_tank_pressure			REAL
north_tank_temp			REAL
+-one_shots			DINT
+-recipe			TANK[3]
+-recipe_number			DINT
replace_bit			BOOL
+-running_hours			COUNTER
+-running_seconds			TIMER
start			BOOL
stop			BOOL

Tag Type

The tag type defines how the tag operates within your project.

If you want the tag to	Then choose this type
Store a value or values for use by logic within the project	Base
Represent another tag.	Alias
Send data to another controller	Produced
Receive data from another controller	Consumed

If you plan to use produced or consumed tags, you must follow additional guidelines as you organize your tags.

Data Type

Term	Definition
Data type	The data type defines the type of data that a tag stores, such as a bit, integer, floating-point value, string, etc.
Structure	A data type that is a combination of other data types. • A structure is formatted to create a unique data type that matches a specific need. • Within a structure, each individual data type is called a member. • Like tags, members have a name and data type. • A Logix5000 controller contains a set of predefined structures (data types) for use with specific instructions such as timers, counters, Function Blocks, etc. • You can create your own structures, called a user-defined data type.

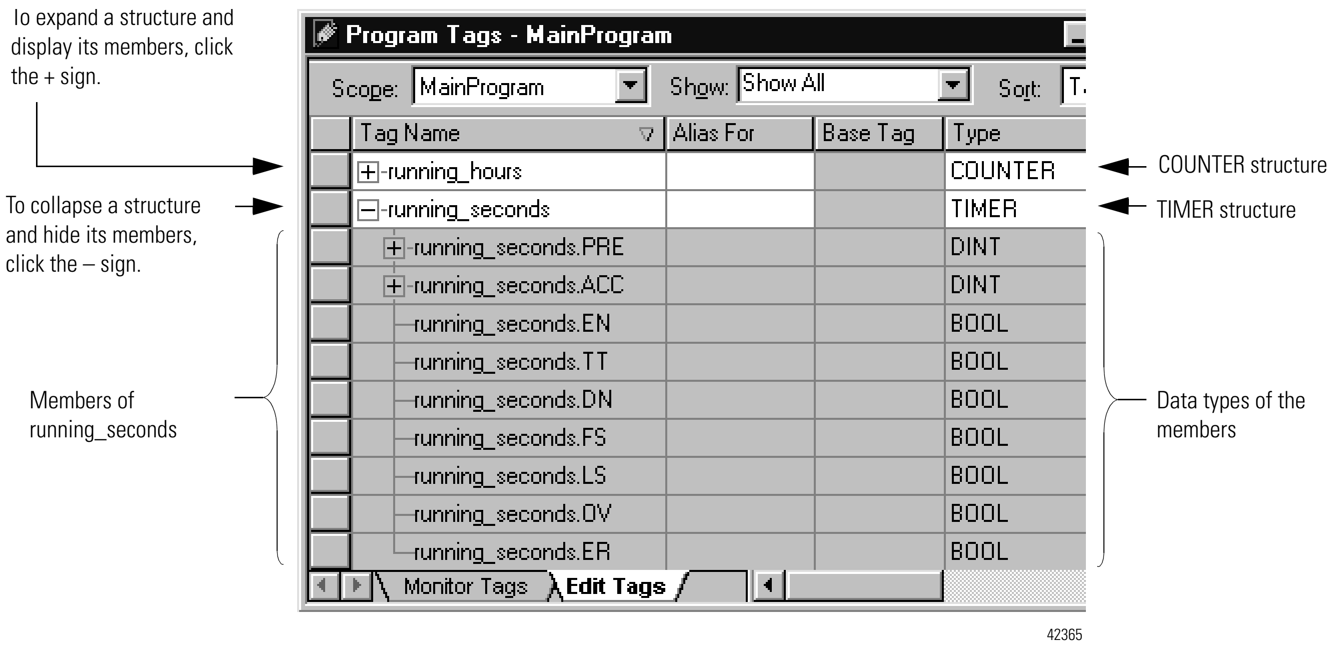
The following table outlines the most common data types and when to use each.

For	Select
Analog device in floating-point mode	REAL
Analog device in integer mode (for very fast sample rates)	INT
ASCII characters	String
Bit	BOOL
Counter	COUNTER
Digital I/O point	BOOL
Floating-point number	REAL
Integer (whole number)	DINT
Sequencer	CONTROL
Timer	TIMER

The minimum memory allocation for a tag is 4 bytes. When you create a tag that stores data that requires less than four bytes, the controller allocates 4 bytes, but the data only fills the part it needs.

Data type	**Bits**			
	31 16	**15 8**	**7 1**	**0**
BOOL	Not used			0 or 1
SINT	Not used		-128 to +127	
INT	Not used	-32,768 to +32767		
DINT	-2,147,483,648 to +2,147,483,647			
REAL	$-3.40282347E^{38}$ to $-1.17549435E^{-38}$ (negative values) 0 $1.17549435E^{-38}$ to $3.40282347E^{38}$ (positive values)			

The COUNTER and TIMER data types are examples of commonly used structures.


Io expand a structure and display its members, click the + sign.
To collapse a structure and hide its members, click the – sign.
Members of running_seconds
COUNTER structure
TIMER structure
Data types of the members
Program Tags - MainProgram
Monitor Tags
Edit Tags
42365

Scope: MainProgram Show: Show All Sort: T

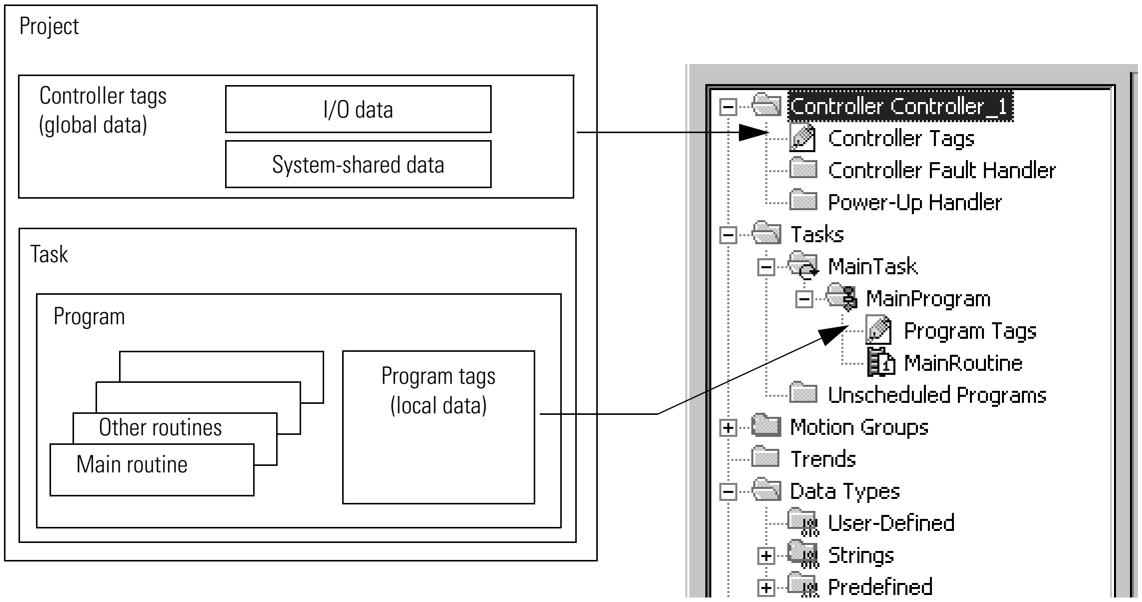
Tag Name	Alias For	Base Tag	Type
+-running_hours			COUNTER
–-running_seconds			TIMER
+-running_seconds.PRE			DINT
+-running_seconds.ACC			DINT
running_seconds.EN			BOOL
running_seconds.TT			BOOL
running_seconds.DN			BOOL
running_seconds.FS			BOOL
running_seconds.LS			BOOL
running_seconds.OV			BOOL
running_seconds.ER			BOOL

To copy data to a structure, use the COP instruction.

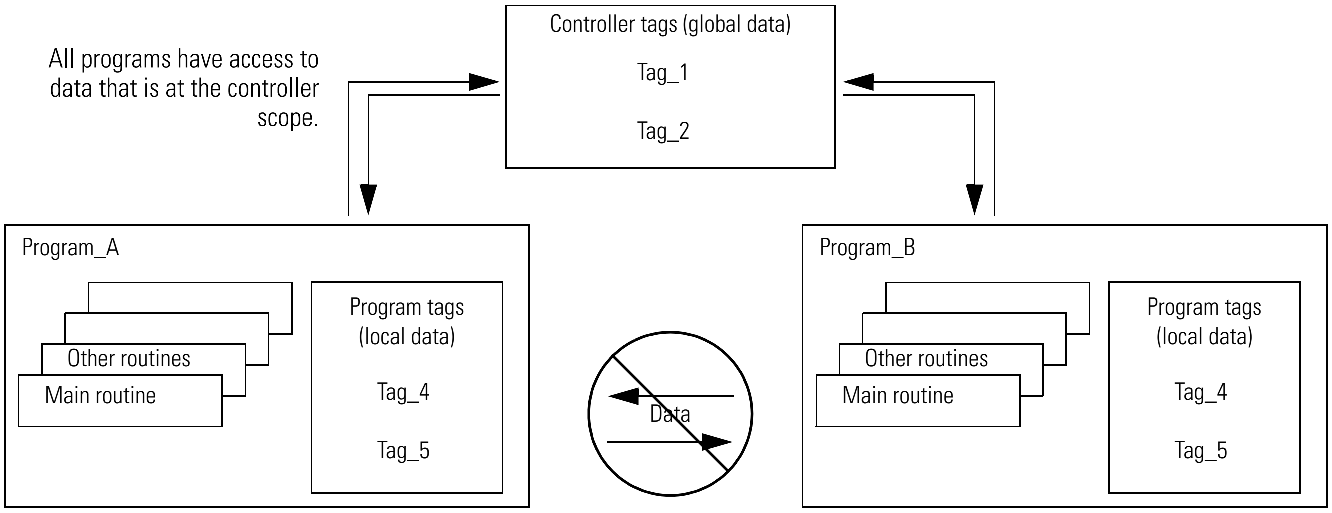
Refer to the Logix5000 Controllers General Instruction Set Reference Manual, publication 1756-RM003.

Scope

When you create a tag, you define it as either a controller tag (global data) or a program tag for a specific program (local data).



A Logix5000 controller lets you divide your application into multiple programs, each with its own data. There is no need to manage conflicting tag names between programs. This makes it easier to re-use both code and tag names in multiple programs.


All programs have access to data that is at the controller scope.
Controller tags (global data)
Tag_1
Tag_2
Program_A
Other routines
Main routine
Program tags (local data)
Tag_4
Tag_5
Data
Program_B
Other routines
Main routine
Program tags (local data)
Tag_4
Tag_5

Data at the program scope is isolated from other programs:

- Routines cannot access data that is at the program scope of another program.
- You can re-use the tag name of a program-scoped tag in multiple programs.

  For example, both Program_A and Program_B can have a program tag named Tag_4.

Avoid using the same name for a both controller tag and a program tag. Within a program, you cannot reference a controller tag if a tag of the same name exists as a program tag for that program.

Certain tags must be controller scope (controller tag).

If you want to use the tag	Then assign this scope
In more than one program in the project	Controller scope (controller tags)
In a Message (MSG) instruction	
To produce or consume data	
To communicate with a PanelView terminal	
None of the above	Program scope (program tags)

Guidelines for Tags

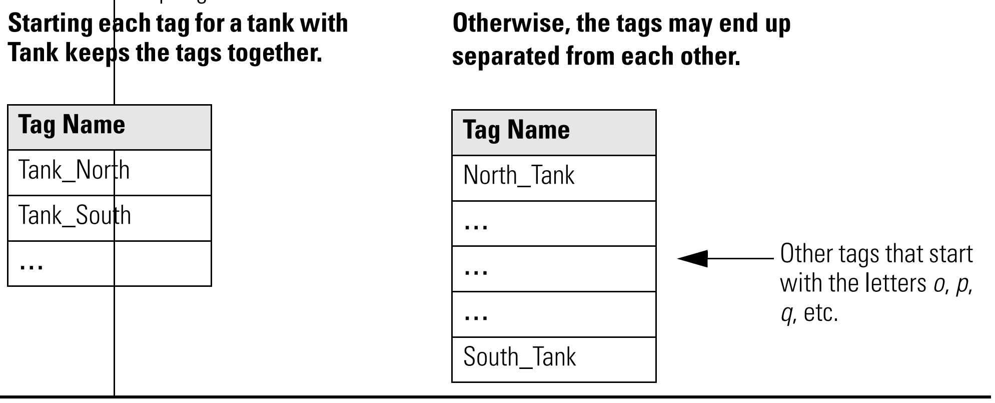
Use the following guidelines to create tags for a Logix5000 project:

Guideline	Details
❑ 1. Create user-defined data types.	User-defined data types (structures) let you organize data to match your machine or process. A user-defined data type provides these advantages: • One tag contains all the data related to a specific aspect of your system. This keeps related data together and easy to locate, regardless of its data type. • Each individual piece of data (member) gets a descriptive name. This automatically creates an initial level of documentation for your logic. • You can use the data type to create multiple tags with the same data lay-out. For example, use a user-defined data type to store all the parameters for a tank, including temperatures, pressures, valve positions, and preset values. Then create a tag for each of your tanks based on that data type.
❑ 2. Use arrays to quickly create a group of similar tags.	An array creates multiple instances of a data type under a common tag name. • Arrays let you organize a block of tags that use the same data type and perform a similar function. • You organize the data in 1, 2, or 3 dimensions to match what the data represents. For example, use a 2 dimension array to organize the data for a tank farm. Each element of the array represents a single tank. The location of the element within the array represents the geographic location of the tank. Important: Minimize the use of BOOL arrays. Many array instructions do not operate on BOOL arrays. This makes it more difficult to initialize and clear an array of BOOL data. • Typically, use a BOOL array for the bit-level objects of a PanelView screen. • Otherwise, use the individual bits of a DINT tag or an array of DINTs.

Guideline	**Details**	
❑ 3. Take advantage of program-scoped tags.	If you want multiple tags with the same name, define each tag at the program scope (program tags) for a different program. This lets you re-use both logic and tag names in multiple programs. Avoid using the same name for both a controller tag and a program tag. Within a program, you cannot reference a controller tag if a tag of the same name exists as a program tag for that program. Certain tags must be controller scope (controller tag).	
	If you want to use the tag	**Then assign this scope**
	In more than one program in the project	Controller scope (controller tags)
	In a Message (MSG) instruction	
	To produce or consume data	
	To communicate with a PanelView terminal	
	None of the above	Program scope (program tags)
❑ 4. For integers, use the DINT data type.	To increase the efficiency of your logic, minimize the use of SINT or INT data types. Whenever possible, use the DINT data type for integers. • A Logix5000 controller typically compares or manipulates values as 32-bit values (DINTs or REALs). • The controller typically converts a SINT or INT value to a DINT or REAL value before it uses the value. • If the destination is a SINT or INT tag, the controller typically converts the value back to a SINT or INT value. • The conversion to or from SINTs or INTs occurs automatically with no extra programming. But it takes extra execution time and memory.	

Guideline	Details
❑ 5. Limit a tag name to 40 characters.	Here are the rules for a tag name: • only alphabetic characters (A-Z or a-z), numeric characters (0-9), and underscores (_) • must start with an alphabetic character or an underscore • no more than 40 characters • no consecutive or trailing underscore characters (_) • not case sensitive
❑ 6. Use mixed case.	Although tags are not case sensitive (upper case *A* is the same as lower case *a*), mixed case is easier to read.
❑ 7. Consider the alphabetical order of tags.	RSLogix 5000 software displays tags of the same scope in alphabetical order. To make it easier to monitor related tags, use similar starting characters for tags that you want to keep together.

These tags are easier to read	Than these tags
Tank_1	TANK_1
Tank1	TANK1
	tank_1
	tank1



Starting each tag for a tank with Tank keeps the tags together.

Tag Name
Tank_North
Tank_South
…

Otherwise, the tags may end up separated from each other.

Tag Name
North_Tank
…
…
…
South_Tank

Other tags that start with the letters *o*, *p*, *q*, etc.

Create a Tag

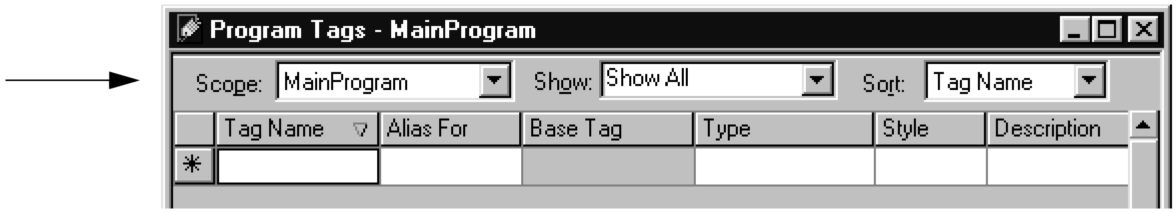
IMPORTANT RSLogix 5000 software automatically creates tags when you:

- add an element to a sequential function chart (SFC).
- add a Function Block instruction to a Function Block diagram.

The Tags window lets you create and edit tags using a spreadsheet-style view of the tags.

1. From the Logic menu, select Edit Tags.

2. Select a scope for the tag.



42350

If you will use the tag:	Then select:
In more than one program within the project	*name_of_controller(controller)*
As a producer or consumer	
In a message	
In only one program within the project	Program that will use the tag

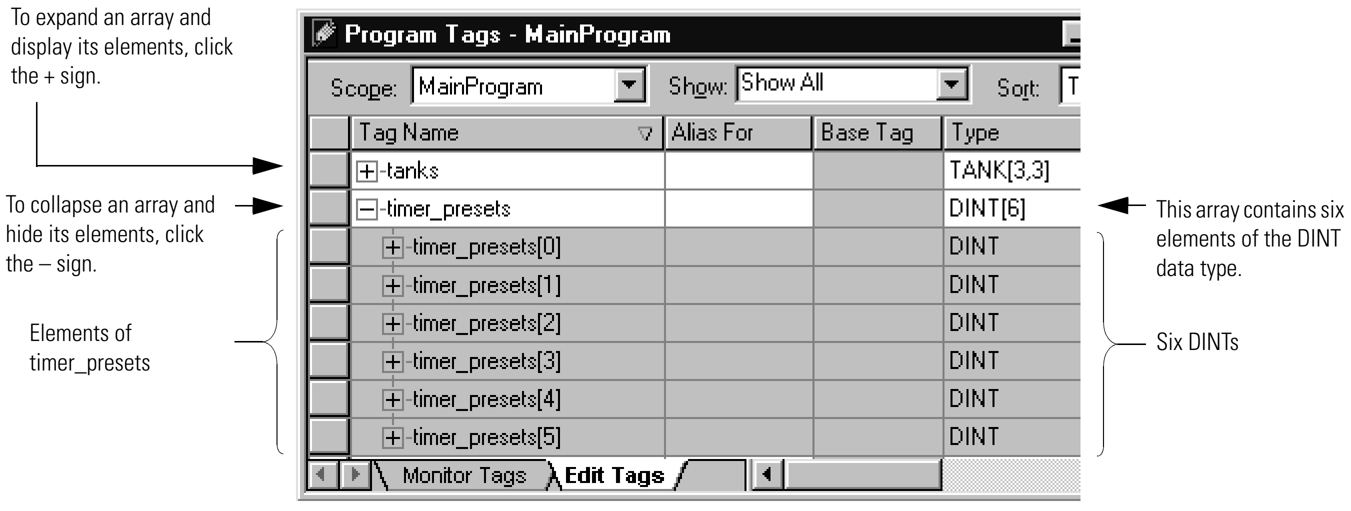
3. Type a name, data type, and description (optional) for the tag.

Create an Array

Logix5000 controllers also let you use arrays to organize data.

Term	Definition
Array	A tag that contains a block of multiple pieces of data. • An array is similar to a file. • Within an array, each individual piece of data is called an element. • Each element uses the same data type. • An array tag occupies a contiguous block of memory in the controller, each element in sequence. • You can use array and sequencer instructions to manipulate or index through the elements of an array • You organize the data into a block of 1, 2, or 3 dimensions.

A subscript (s) identifies each individual element within the array. A subscript starts at 0 and extends to the number of elements minus 1 (zero based).



To expand an array and display its elements, click the + sign.

To collapse an array and hide its elements, click the – sign.

Elements of timer_presets

This array contains six elements of the DINT data type.

Six DINTs

Program Tags - MainProgram

Scope: MainProgram Show: Show All Sort: T

Tag Name	Alias For	Base Tag	Type
⊞-tanks			TANK[3,3]
⊟-timer_presets			DINT[6]
⊞-timer_presets[0]			DINT
⊞-timer_presets[1]			DINT
⊞-timer_presets[2]			DINT
⊞-timer_presets[3]			DINT
⊞-timer_presets[4]			DINT
⊞-timer_presets[5]			DINT

42367

The following example compares a structure to an array:

This is a tag that uses the Timer structure (data type).

Tag Name	Data Type
⊟ Timer_1	TIMER
⊞ Timer_1.PRE	DINT
⊞ Timer_1.ACC	DINT
Timer_1.EN	BOOL
Timer_1.TT	BOOL
Timer_1.DN	BOOL

This is a tag that uses an array of the Timer data type.

Tag Name	Data Type
⊟ Timers	TIMER[3]
⊞ Timer[0]	TIMER
⊞ Timer[1]	TIMER
⊞ Timer[2]	TIMER

EXAMPLE

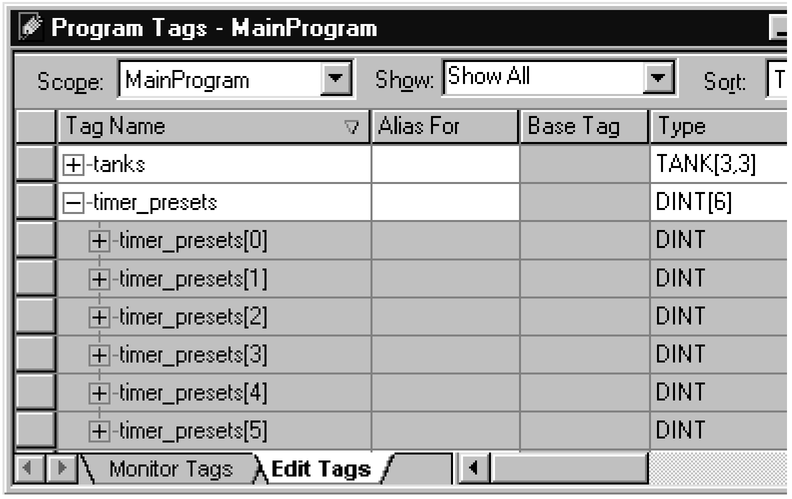
Single dimension array

In this example, a single timer instruction times the duration of several steps. Each step requires a different preset value. Because all the values are the same data type (DINTs) an array is used.

To expand an array and display its elements, click the + sign.

To collapse an array and hide its elements, click the – sign.

Elements of timer_presets



Tag Name	Alias For	Base Tag	Type
⊞-tanks			TANK[3,3]
⊟-timer_presets			DINT[6]
⊞-timer_presets[0]			DINT
⊞-timer_presets[1]			DINT
⊞-timer_presets[2]			DINT
⊞-timer_presets[3]			DINT
⊞-timer_presets[4]			DINT
⊞-timer_presets[5]			DINT

This array contains six elements of the DINT data type.

Six DINTs

42367

EXAMPLE

Two dimension array

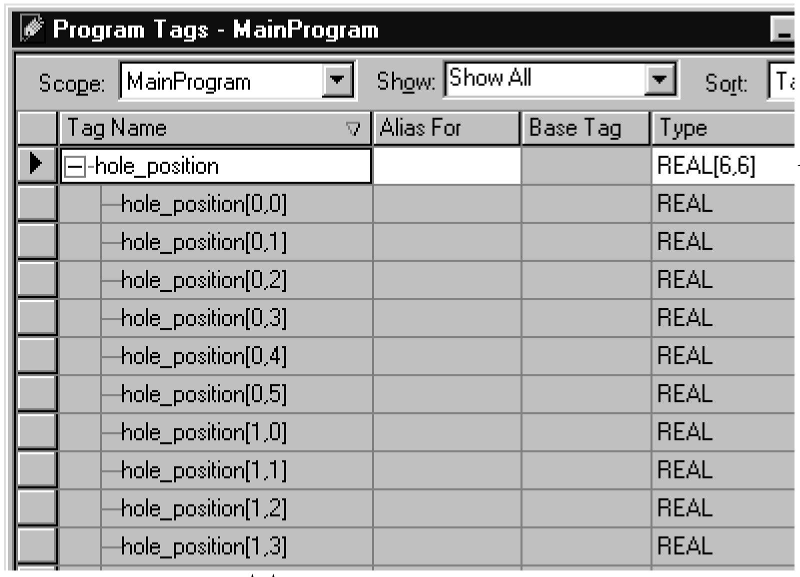
A drill machine can drill one to five holes in a book. The machine requires a value for the position of each hole from the leading edge of the book. To organize the values into configurations, a two dimension array is used. The first subscript indicates the hole to which the value corresponds and the second subscript indications how many holes will be drilled (one to five).

Subscript of second dimension

Subscript of first dimension

	0	1	2	3	4	5	Description
0							
1		1.5	2.5	1.25	1.25	1.25	Position of first hole from leading edge of book
2			8.0	5.5	3.5	3.5	Position of second hole from leading edge of book
3				9.75	7.5	5.5	Position of third hole from leading edge of book
4					9.75	7.5	Position of fourth hole from leading edge of book
5						9.75	Position of fifth hole from leading edge of book

In the Tags window, the elements are in the order depicted below.



Tag Name	Alias For	Base Tag	Type
hole_position			REAL[6,6]
hole_position[0,0]			REAL
hole_position[0,1]			REAL
hole_position[0,2]			REAL
hole_position[0,3]			REAL
hole_position[0,4]			REAL
hole_position[0,5]			REAL
hole_position[1,0]			REAL
hole_position[1,1]			REAL
hole_position[1,2]			REAL
hole_position[1,3]			REAL

This array contains a two-dimensional grid of elements, six elements by six elements.

42367

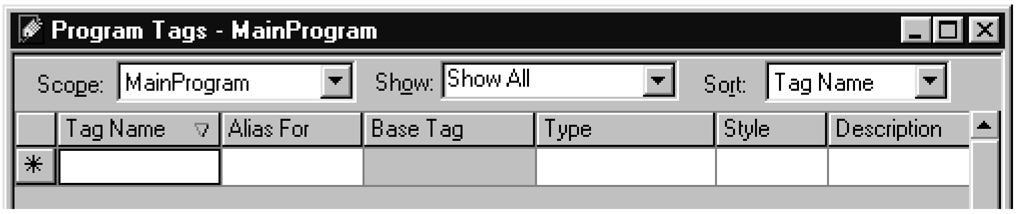
The right-most dimension increments to its maximum value then starts over.

When the right-most dimension starts over, the dimension to the left increments by one.

Create an Array

To create an array, you create a tag and assign dimensions to the data type:

1. From the Logic menu, select Edit Tags.



42350

2. Type a name for the tag and select a scope for the tag:

If you will use the tag	Then select
In more than one program within the project	*name_of_controller(controller)*
As a producer or consumer	
In a message	
In only one program within the project	Program that will use the tag

3. Assign the array dimensions:

If the tag is	Then type	Where
One dimension array	*data_type[x]*	*data_type* is the type of data that the tag stores. *x* is the number of **elements** in the first dimension. *y* is the number of elements in the second dimension. *z* is the number of elements in the third dimension.
Two dimension array	*data_type[x,y]*	
Three dimension array	*data_type[x,y,z]*	

Create a User-Defined Data Type

User-defined data types (structures) let you organize your data to match your machine or process.

EXAMPLE

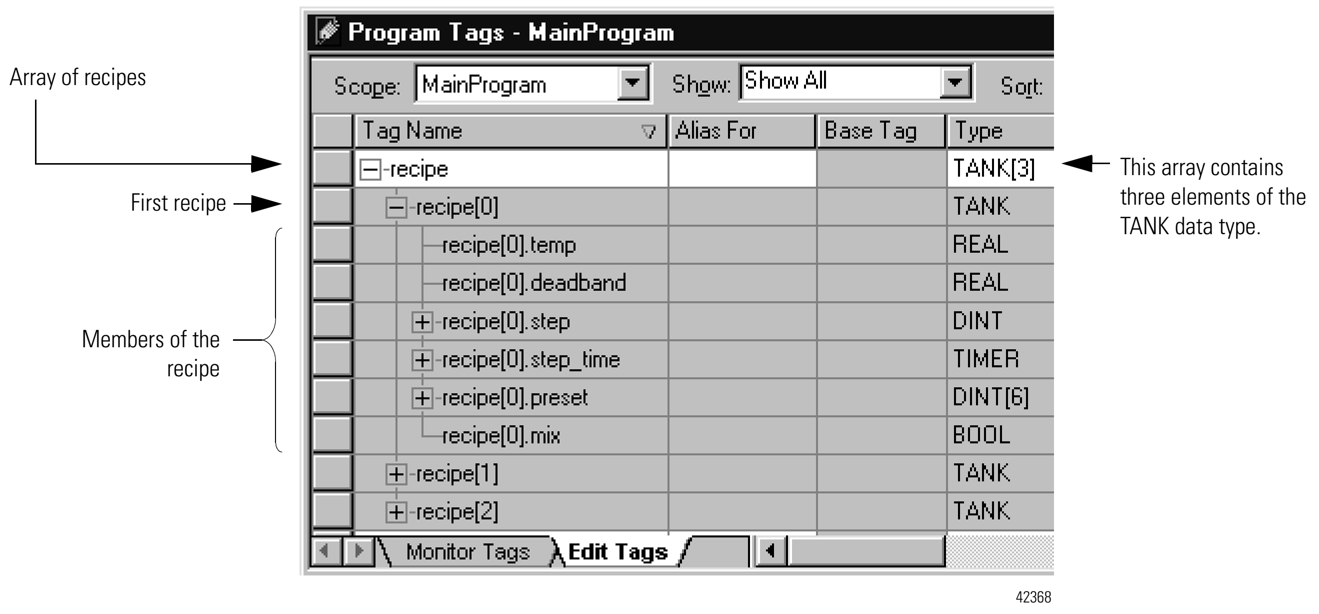
User-defined data type that stores a recipe

In a system of several tanks, each tank can run a variety of recipes. Because the recipe requires a mix of data types (REAL, DINT, BOOL, etc.) a user-defined data type is used.

Name (of data type): TANK

Member Name	Data Type
Temp	REAL
Deadband	REAL
Step	DINT
Step_time	TIMER
Preset	DINT[6]
Mix	BOOL

An array that is based on this data type would look like this:


Array of recipes
First recipe
Members of the recipe
This array contains three elements of the TANK data type.
Program Tags - MainProgram
Monitor Tags
Edit Tags
42368

Scope: MainProgram Show: Show All Sort:

Tag Name	Alias For	Base Tag	Type
⊟-recipe			TANK[3]
⊟-recipe[0]			TANK
recipe[0].temp			REAL
recipe[0].deadband			REAL
⊞-recipe[0].step			DINT
⊞-recipe[0].step_time			TIMER
⊞-recipe[0].preset			DINT[6]
recipe[0].mix			BOOL
⊞-recipe[1]			TANK
⊞-recipe[2]			TANK

EXAMPLE

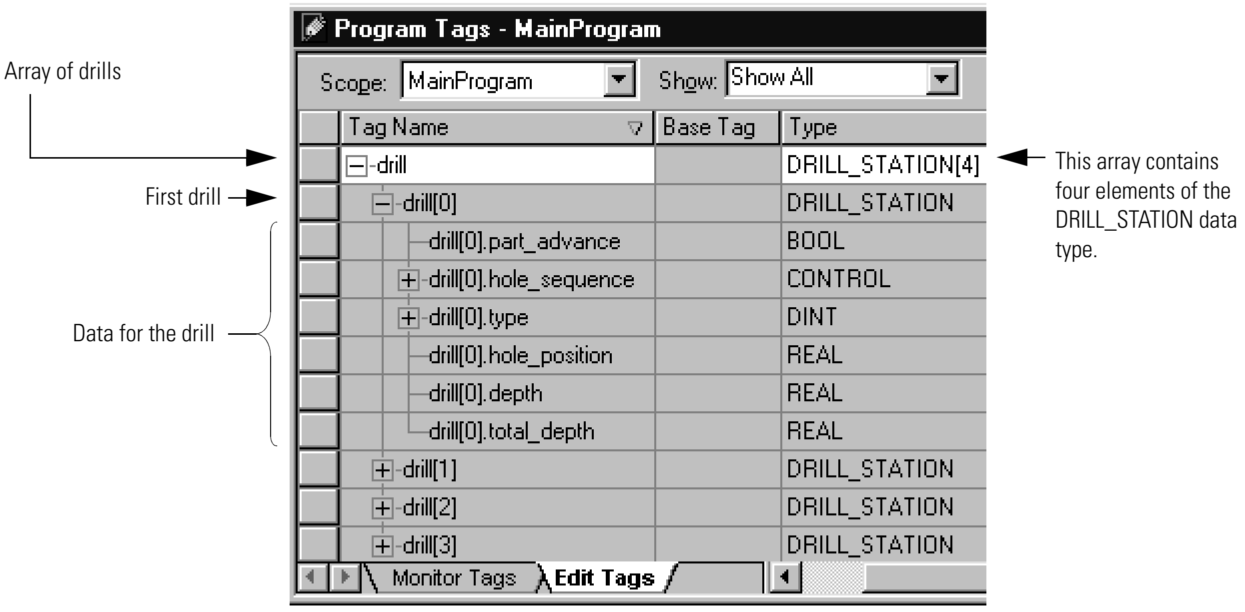
User-defined data type that stores the data that is required to run a machine

Because several drill stations require the following mix of data, use a user-defined data type

Name (of data type): DRILL_STATION

Member Name	**Data Type**
Part_advance	BOOL
Hole_sequence	CONTROL
Type	DINT
Hole_position	REAL
Depth	REAL
Total_depth	REAL

An array that is based on this data type looks like:



Tag Name	Base Tag	Type
−-drill		DRILL_STATION[4]
−-drill[0]		DRILL_STATION
drill[0].part_advance		BOOL
+-drill[0].hole_sequence		CONTROL
+-drill[0].type		DINT
drill[0].hole_position		REAL
drill[0].depth		REAL
drill[0].total_depth		REAL
+-drill[1]		DRILL_STATION
+-drill[2]		DRILL_STATION
+-drill[3]		DRILL_STATION

Guidelines for User-Defined Data Types

When you create a user-defined data type, keep the following in mind:

- If you include members that represent I/O devices, you must use logic to copy the data between the members in the structure and the corresponding I/O tags. Refer to Address I/O Data on page 13.
- If you include an array as a member, limit the array to a single dimension. Multi-dimension arrays are *not* permitted in a user-defined data type.
- When you use the BOOL, SINT, or INT data types, place members that use the same data type in sequence:

More efficient

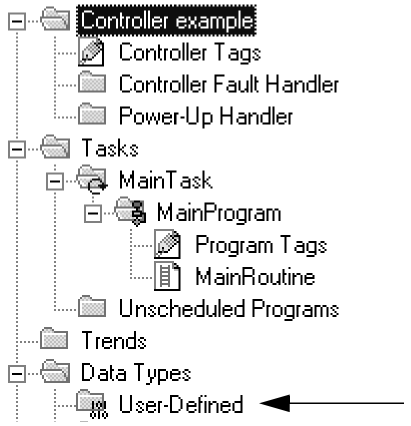
BOOL
BOOL
BOOL
DINT
DINT

Less efficient

BOOL
DINT
BOOL
DINT
BOOL

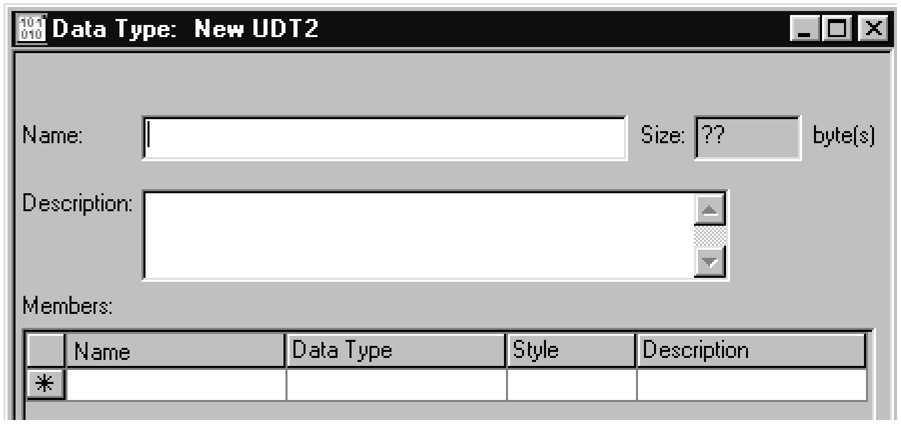
Create a User-Defined Data Type

1. From the User-Defined folder under Data Types, right-click User-Defined and select New Data Type.



2. Type a name for the array, description for the array.

3. For each member of the array, type a name, data type, style, and description (optional).



42196

Limit any arrays to a single dimension.

To display the value (s) of the member in a different style (radix), select the style.

4. Click Apply.

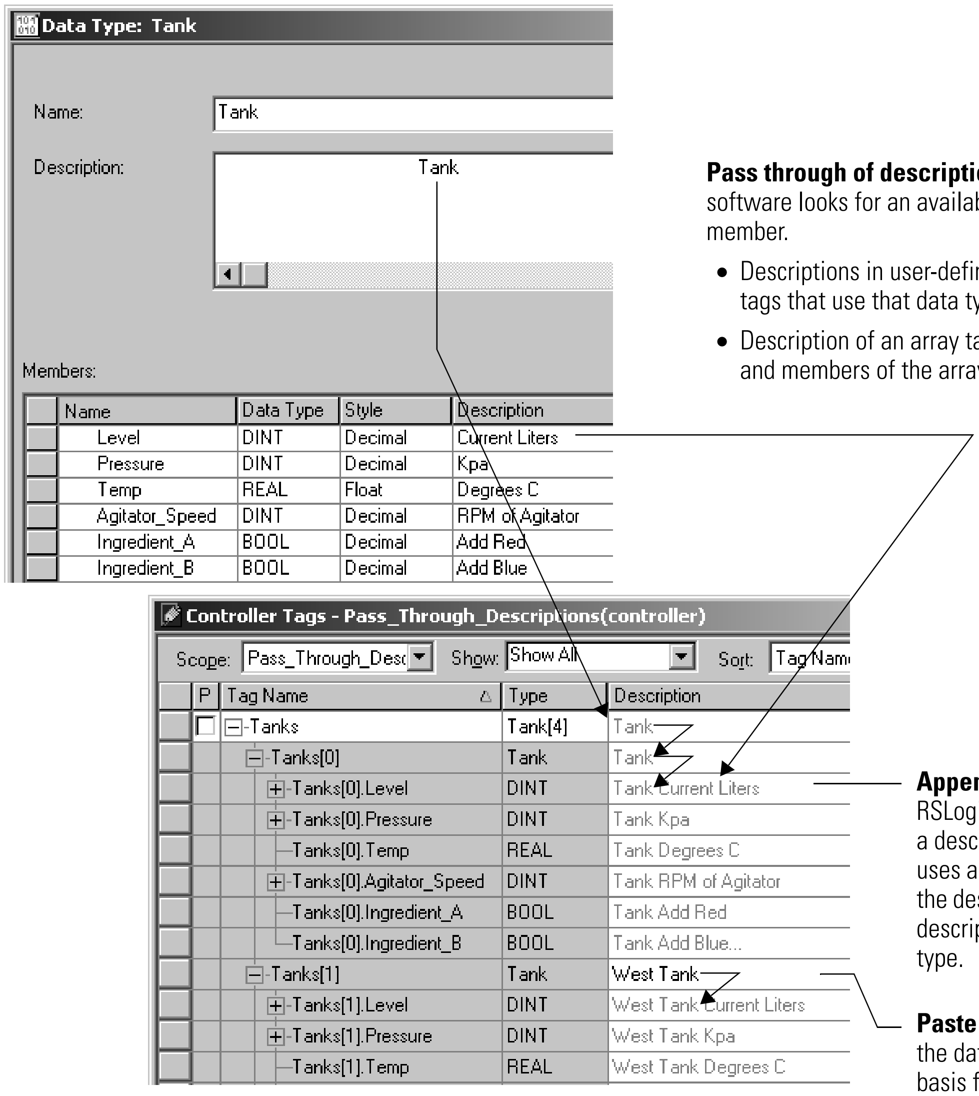
5. Add as many members as needed.

Describe a User-Defined Data Type

RSLogix 5000 software 13.0 or later

RSLogix 5000 software lets you automatically build descriptions out of the descriptions in your user-defined data types. This greatly reduces the amount of time you have to spend documenting your project.

As you organize your user-defined data types, keep in mind the following features of RSLogix 5000 software:



Data Type: Tank

Name: Tank

Description: Tank

Members:

Name	Data Type	Style	Description
Level	DINT	Decimal	Current Liters
Pressure	DINT	Decimal	Kpa
Temp	REAL	Float	Degrees C
Agitator_Speed	DINT	Decimal	RPM of Agitator
Ingredient_A	BOOL	Decimal	Add Red
Ingredient_B	BOOL	Decimal	Add Blue

Controller Tags - Pass_Through_Descriptions(controller)

Scope: Pass_Through_Desc Show: Show All Sort: Tag Nam

P	Tag Name	Type	Description
☐	-Tanks	Tank[4]	Tank
	-Tanks[0]	Tank	Tank
	+-Tanks[0].Level	DINT	Tank Current Liters
	+-Tanks[0].Pressure	DINT	Tank Kpa
	-Tanks[0].Temp	REAL	Tank Degrees C
	+-Tanks[0].Agitator_Speed	DINT	Tank RPM of Agitator
	-Tanks[0].Ingredient_A	BOOL	Tank Add Red
	-Tanks[0].Ingredient_B	BOOL	Tank Add Blue...
	-Tanks[1]	Tank	West Tank
	+-Tanks[1].Level	DINT	West Tank Current Liters
	+-Tanks[1].Pressure	DINT	West Tank Kpa
	-Tanks[1].Temp	REAL	West Tank Degrees C

Pass through of descriptions – When possible, RSLogix 5000 software looks for an available description for a tag, element, or member.

- Descriptions in user-defined data types ripple through to the tags that use that data type.
- Description of an array tag ripples through to the elements and members of the array.

Append description to base tag – RSLogix 5000 software automatically builds a description for each member of a tag that uses a user-defined data type. It starts with the description of the tag and then adds the description of the member from the data type.

Paste pass-through description – Use the data type and array description as a basis for more specific descriptions.

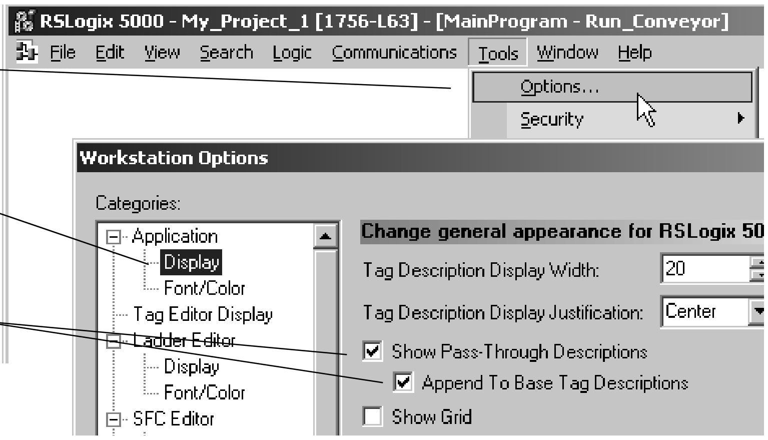
In this example, Tank became West Tank.

RSLogix 5000 software uses different colors for descriptions:

A description in this color	Is a
Gray	Pass-through description
Black	Manually entered description

Turn Pass-Through and Append Descriptions On or Off

1. In RSLogix 5000 software, choose Tools > Options.
2. Select the Application > Display.
3. Turn on (check) or turn off (uncheck) the desired options.



Change general appearance for RSLogix 50

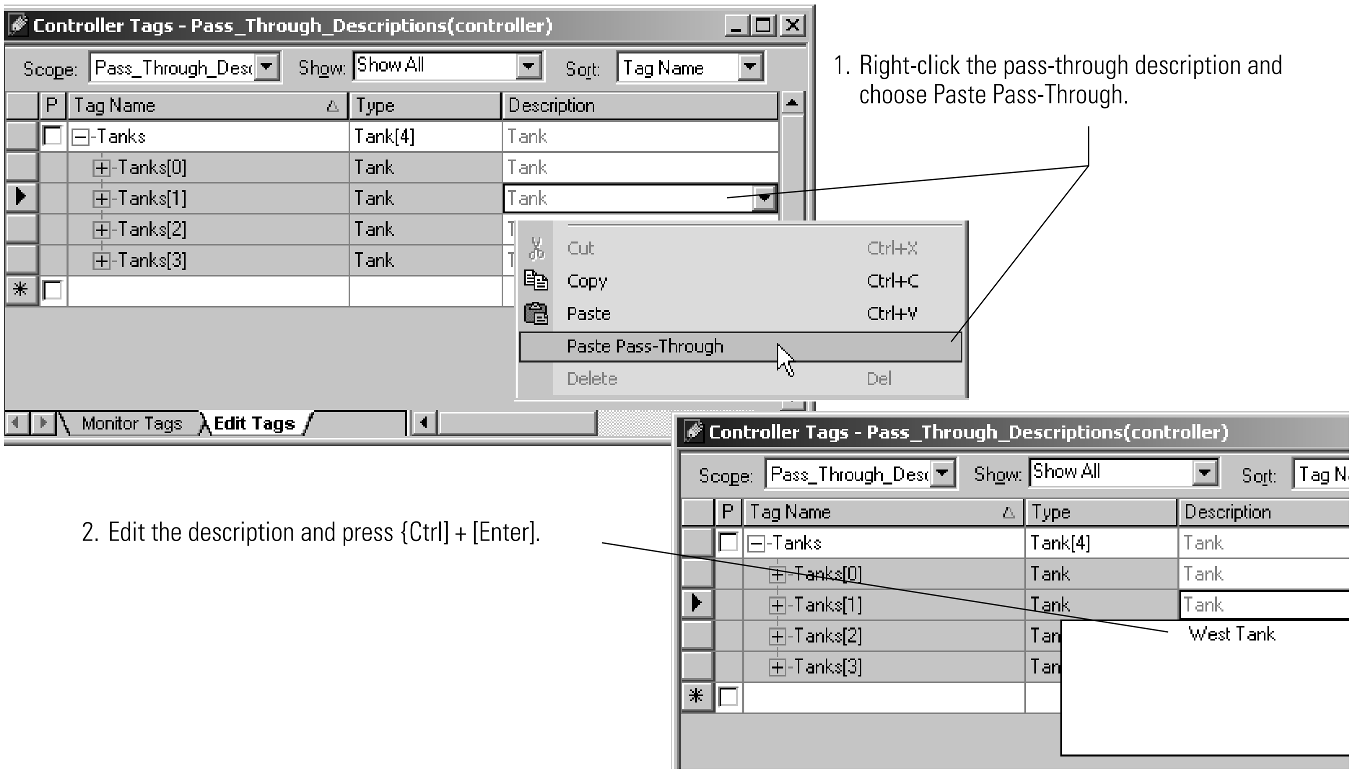
Tag Description Display Width: 20

Tag Description Display Justification: Center

- [✓] Show Pass-Through Descriptions
- [✓] Append To Base Tag Descriptions
- [ ] Show Grid

Paste a Pass-Through Description

To use a pass-through description as the starting point for a more specific description:



P	Tag Name	Type	Description
☐	-Tanks	Tank[4]	Tank
	+Tanks[0]	Tank	Tank
	+Tanks[1]	Tank	Tank
	+Tanks[2]	Tank	T
	+Tanks[3]	Tank	T

1. Right-click the pass-through description and choose Paste Pass-Through.

2. Edit the description and press {Ctrl} + [Enter].

P	Tag Name	Type	Description
☐	-Tanks	Tank[4]	Tank
	+Tanks[0]	Tank	Tank
	+Tanks[1]	Tank	Tank West Tank
	+Tanks[2]	Tan	
	+Tanks[3]	Tan	

Address Tag Data

An tag name follows this format:

Name *[Element]* *.Member* *[Element]* *.Bit*
or
.[Index]

▨ = Optional

Where	Is
Name	Name that identifies this specific tag.
Element	Subscript or subscripts that point to a specific element within an array. • Use the element identifier only if the tag or member is an array. • Use one subscript for each dimension of the array. For example: [5], [2,8], [3,2,7]. To indirectly (dynamically) reference an element, use a tag or numeric expression that provides the element number. • A numeric expression uses a combination of tags, constants, operators, and functions to calculate a value. For example, Tag_1-Tag_2, Tag_3+4, ABS (Tag_4). • Keep the value of the tag or numeric expression within the dimensions of the array. For example, if a dimension of an array contains 10 elements, then the value of the tag or numeric expression must be 0 to 9 (10 elements).
Member	Specific member of a structure. • Use the member identifier only if the tag is a structure. • If the structure contains another structure as one of its members, use additional levels of the.Member format to identify the required member.
Bit	Specific bit of an integer data type (SINT, INT, or DINT).
Index	To indirectly (dynamically) reference a bit of an integer, use a tag or numeric expression that provides the bit number. • A numeric expression uses a combination of tags, constants, operators, and functions to calculate a value. For example, Tag_1-Tag_2, Tag_3+4, ABS(Tag_4). • Keep the value of the tag or numeric expression within the range of bits of the integer tag. For example, if the integer tag is a Dint (32-bits), then the value of the index must be 0 to 31 (32-bits).

Assigning Alias Tags

An alias tag lets you create one tag that represents another tag.

- Both tags share the same value (s).
- When the value (s) of one of the tags changes, the other tag reflects the change as well.

Use aliases in the following situations:

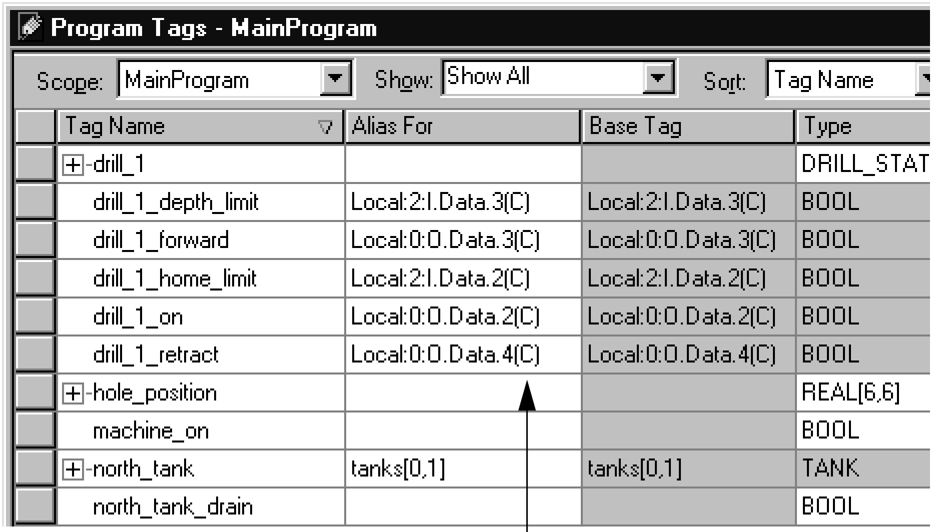
- Program logic in advance of wiring diagrams.
- Assign a descriptive name to an I/O device.
- Provide a more simple name for a complex tag.
- Use a descriptive name for an element of an array.

The tags window displays alias information.

drill_1_depth_limit is an alias for Local:2:I.Data.3 (a digital input point). When the input turns on, the alias tag also turns on.

drill_1_on is an alias for Local:0:O.Data.2 (a digital output point). When the alias tag turns on, the output tag also turns on.

north_tank is an alias for tanks[0,1].



Program Tags - MainProgram

Scope: MainProgram Show: Show All Sort: Tag Name

Tag Name	Alias For	Base Tag	Type
⊞-drill_1			DRILL_STAT
drill_1_depth_limit	Local:2:I.Data.3(C)	Local:2:I.Data.3(C)	BOOL
drill_1_forward	Local:0:O.Data.3(C)	Local:0:O.Data.3(C)	BOOL
drill_1_home_limit	Local:2:I.Data.2(C)	Local:2:I.Data.2(C)	BOOL
drill_1_on	Local:0:O.Data.2(C)	Local:0:O.Data.2(C)	BOOL
drill_1_retract	Local:0:O.Data.4(C)	Local:0:O.Data.4(C)	BOOL
⊞-hole_position			REAL[6,6]
machine_on			BOOL
⊞-north_tank	tanks[0,1]	tanks[0,1]	TANK
north_tank_drain			BOOL

42360

The *(C)* indicates that the tag is at the controller scope.

A common use of alias tags is to program logic before wiring diagrams are available.

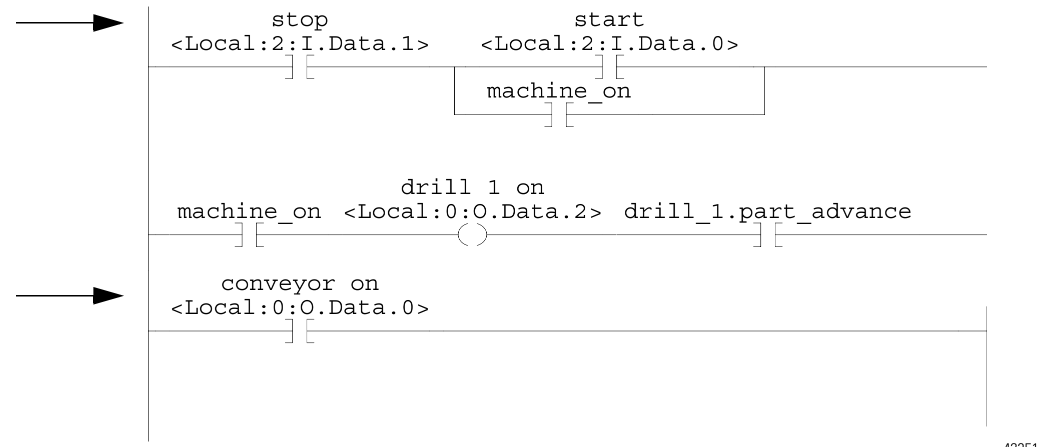
1. For each I/O device, create a tag with a name that describes the device, such as conveyor for the conveyor motor.
2. Program your logic using the descriptive tag names. (You can even test your logic without connecting to the I/O.)
3. Later, when wiring diagrams are available, add the I/O modules to the I/O configuration of the controller.
4. Finally, convert the descriptive tags to aliases for their respective I/O points or channels.

The following logic was initially programmed using descriptive tag names, such as stop and conveyor_on. Later, the tags were converted to aliases for the corresponding I/O devices.

stop is an alias for Local:2:I.Data.1 (the stop button on the operator panel)

conveyor_on is an alias for Local:0:O.Data.0

(the starter contactor for the conveyor motor)



Display Alias Information

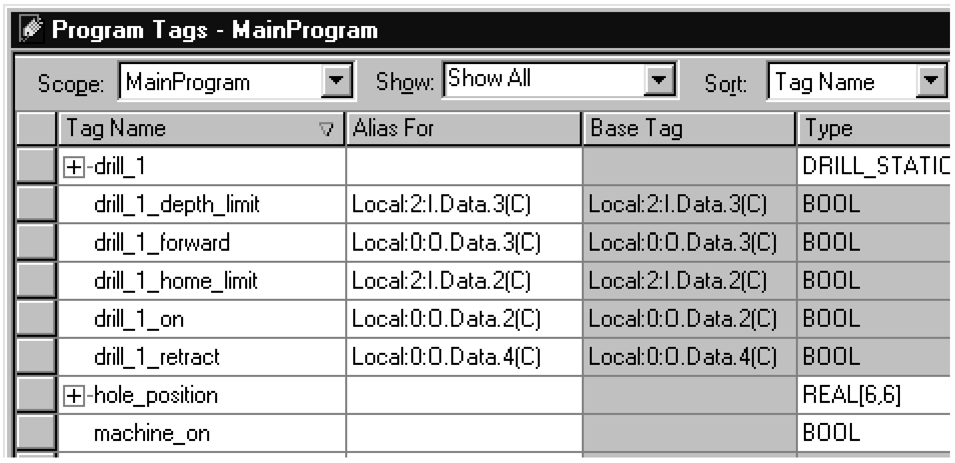
To show (in your logic) the tag to which an alias points, do the following.

1. From the Tools menu, select Options.
2. Select the Ladder Display tab.
3. Check the Show Tag Alias Information check box.
4. Click OK.

Assign an Alias

To assign a tag as an alias tag for another tag, do the following.

1. From the Logic menu, select Edit Tags.



Program Tags - MainProgram

Scope: MainProgram Show: Show All Sort: Tag Name

Tag Name	Alias For	Base Tag	Type
+-drill_1			DRILL_STATIC
drill_1_depth_limit	Local:2:I.Data.3(C)	Local:2:I.Data.3(C)	BOOL
drill_1_forward	Local:0:O.Data.3(C)	Local:0:O.Data.3(C)	BOOL
drill_1_home_limit	Local:2:I.Data.2(C)	Local:2:I.Data.2(C)	BOOL
drill_1_on	Local:0:O.Data.2(C)	Local:0:O.Data.2(C)	BOOL
drill_1_retract	Local:0:O.Data.4(C)	Local:0:O.Data.4(C)	BOOL
+-hole_position			REAL[6,6]
machine_on			BOOL

42360

2. Select the scope of the tag.

3. To the right of the tag name, click the Alias For cell.

   The cell displays a ▼.

4. Click the ▼.

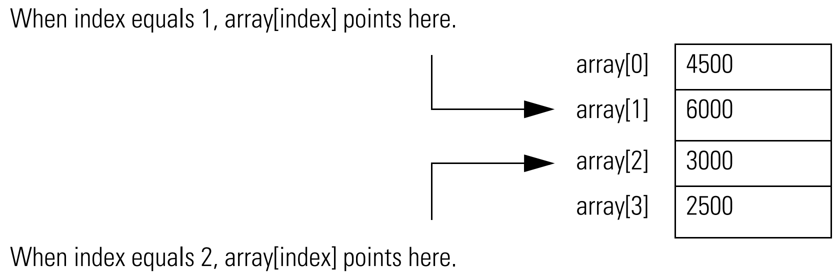
5. Select the tag that the alias will represent:

To	Do this
Select a tag	Double-click the tag name.
Select a bit number	A. Click the tag name. B. To the right of the tag name, click ▼. C. Click the required bit.

6. Press [Enter] or click another cell.

Assigning an Indirect Address

If you want an instruction to access different elements in an array, use a tag in the subscript of the array (an indirect address). By changing the value of the tag, you change the element of the array that your logic references.


When index equals 1, array[index] points here.
array[0]
4500
array[1]
6000
array[2]
3000
array[3]
2500
When index equals 2, array[index] points here.

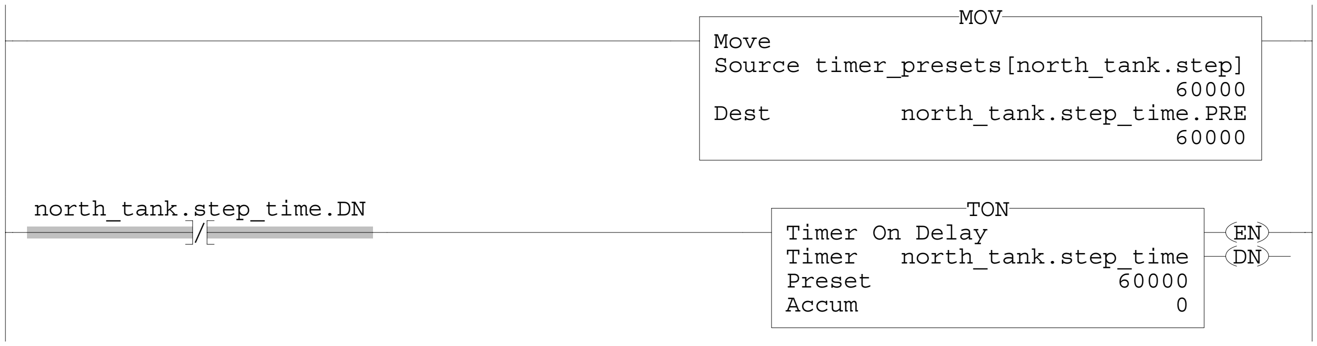
The following table outlines some common uses for an indirect address:

To	Use a tag in the subscript and
Select a recipe from an array of recipes	Enter the number of the recipe in the tag.
Load a specific machine setup from an array of possible setups	Enter the desired setup in the tag.
Load parameters or states from an array, one element at a time	A. Perform the required action on the first element. B. Use an ADD instruction to increment the tag value and point to the next element in the array.
Log error codes	
Perform several actions on an array element and then index to the next element	

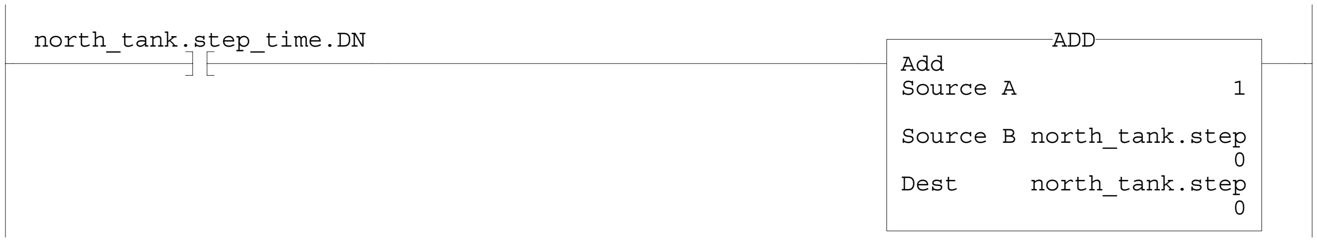
The following example loads a series of preset values into a timer, one value (array element) at a time.

EXAMPLE Step through an array

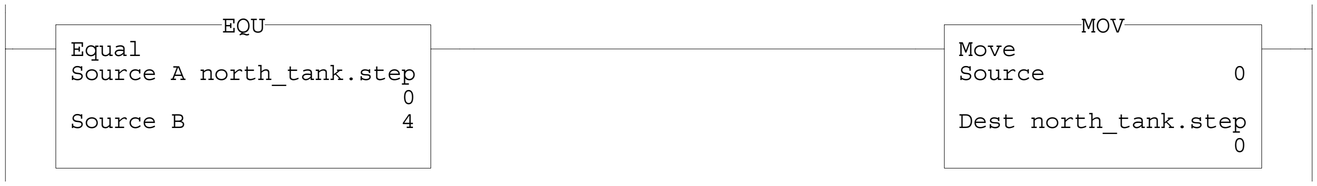
The timer_presets array stores a series of preset values for the timer in the next rung. The north_tank.step tag points to which element of the array to use. For example, when north_tank.step equals 0, the instruction loads timer_presets[0] into the timer (60,000 ms).


MOV
Move
Source timer_presets[north_tank.step]
60000
Dest north_tank.step_time.PRE
60000
north_tank.step_time.DN
TON
Timer On Delay
Timer north_tank.step_time
Preset 60000
Accum 0
EN
DN

When north_tank.step_time is done, the rung increments north_tank.step to the next number and that element of the timer_presets array loads into the timer.



When north_tank.step exceeds the size of the array, the rung resets the tag to start at the first element in the array. (The array contains elements 0 to 3.)



42358

Expressions

You can also use an expression to specify the subscript of an array.

- An expression uses operators, such as + or -, to calculate a value.
- The controller computes the result of the expression and uses it as the array subscript.

You can use these operators to specify the subscript of an array:

Operator	Description
+	Add
-	Subtract/negate
*	Multiply
/	Divide
ABS	Absolute value
AND	AND
FRD	BCD to integer

Operator	Description
MOD	Modulo
NOT	Complement
OR	OR
SQR	Square root
TOD	Integer to BCD
TRN	Truncate
XOR	Exclusive OR

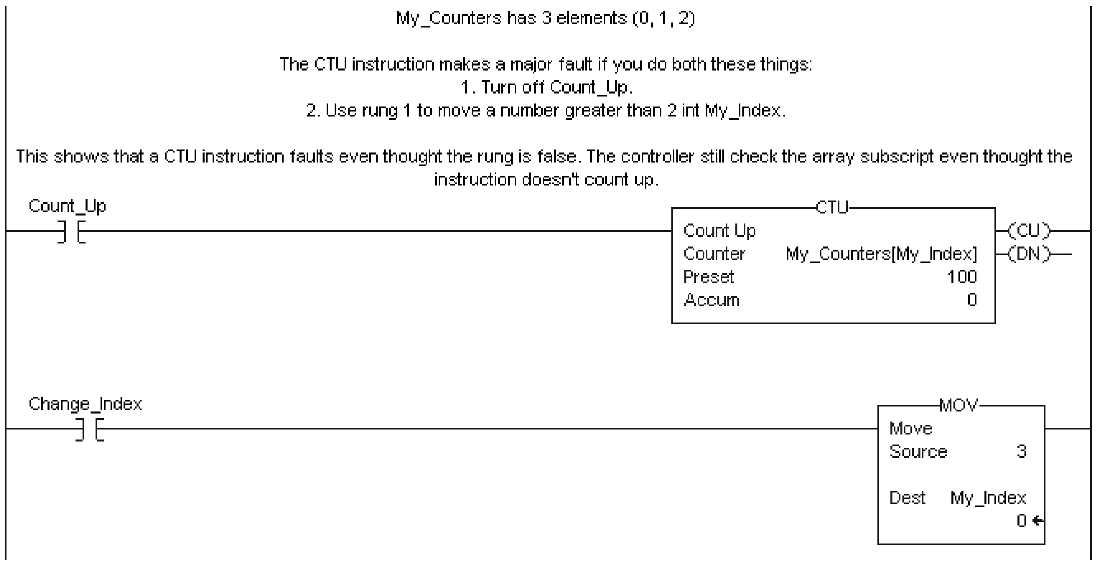
Format your expressions as follows:

If the operator requires	Use this format	Examples
One value (tag or expression)	*operator(value)*	ABS(tag_a)
Two values (tags, constants, or expressions)	*value_a operator value_b*	• tag_b + 5 • tag_c AND tag_d • (tag_e ** 2) MOD (tag_f / tag_g)

Array Subscript Out of Range

Every instruction generates a major fault if the array subscript is out of range. Transitional instructions also generate a major fault even if the rung is false. The controller checks the array subscript in these instructions even if the rung is false.

EXAMPLE



My_Counters has 3 elements (0, 1, 2)

The CTU instruction makes a major fault if you do both these things:
1. Turn off Count_Up.
2. Use rung 1 to move a number greater than 2 int My_Index.

This shows that a CTU instruction faults even thought the rung is false. The controller still check the array subscript even thought the instruction doesn't count up.

```
Count_Up                                CTU
] [                          Count Up
                             Counter   My_Counters[My_Index]   (CU)
                             Preset                      100   (DN)
                             Accum                         0

Change_Index                            MOV
] [                          Move
                             Source          3
                             Dest     My_Index
                                             0
```

For more information on handling major faults, refer to the Major and Minor Faults Programming Manual, publication 1756-PM014.

Tag Documentation

Since you can create four different types of tags, the descriptions you document for each tag will vary. You can create tags of these types:

- Base
- Alias
- Produced
- Consumed

IMPORTANT RSLogix 5000 programming software automatically assigns what are called pass-through descriptions of the tags you have created, descriptions you may or may not want to use.

Base Tag Documentation

When you create a tag without specifying a tag type, RSLogix 5000 will automatically assign your tag a default type of Base.

Since base tags enable you to create your own internal data storage, you can document in your tag description the nature of the data being stored.

Alias Tag Documentation

By creating an Alias tag, you can assign your own name to an existing tag, structure tag member, or bit. In the description of your Alias tag, you can describe the tag to which your alias tag refers.

Produced Tag Documentation

A Produced tag refers to a tag that is consumed by another controller. In the description of your Produced tag, you can describe the remote controllers to which you want to make your Produced tag available through controller-to-controller messaging.

Consumed Tag Documentation

A Consumed tag refers to a tag that is produced by another controller and whose data you want to use in your controller. In the description of your Consumed tag, you can describe the ways in which you want to use a produced tag’s data or the data-producing controller.

Language Switching

With RSLogix 5000 software, version 17, you have the option to display project documentation, such as tag descriptions and rung comments for any supported localized language. You can store project documentation for multiple languages in a single project file rather than in language-specific project files. You define all the localized languages that the project will support and set the current, default, and optional custom localized language. The software uses the default language if the current language's content is blank for a particular component of the project. However, you can use a custom language to tailor documentation to a specific type of project file user.

Enter the localized descriptions in your RSLogix 5000 project, either when programming in that language or by using the import/export utility to translate the documentation off-line and then import it back into the project. Once you enable language switching in RSLogix 5000 software, you can dynamically switch between languages as you use the software.

Project documentation that supports multiple translations within a project includes:

- Component descriptions in tags, routines, programs, user-defined data types, and Add-On Instructions.
- Equipment phases.
- Trends.
- Controllers.
- Alarm Messages (in ALARM_ANALOG and ALARM_DIGITAL configuration).
- Tasks.
- Property descriptions for modules in the Controller Organizer.
- Rung comments, SFC text boxes, and FBD text boxes.

For more information on enabling a project to support multiple translations of project documentation, see the online help.

Notes:

Force I/O

Introduction

Use a force to override data that your logic either uses or produces. For example, use forces to:

- test and debug your logic.
- check wiring to an output device.
- temporarily keep your process functioning when an input device has failed.

Use forces only as a temporary measure. They are not intended to be a permanent part of your application.

Precautions

When you use forces, take these precautions.

ATTENTION

Forcing can cause unexpected machine motion that could injure personnel. Before you use a force, determine how the force will effect your machine or process and keep personnel away from the machine area.

- Enabling I/O forces causes input, output, produced, or consumed values to change.
- Enabling SFC forces causes your machine or process to go to a different state or phase.
- Removing forces may still leave forces in the enabled state.
- If forces are enabled and you install a force, the new force immediately takes effect.

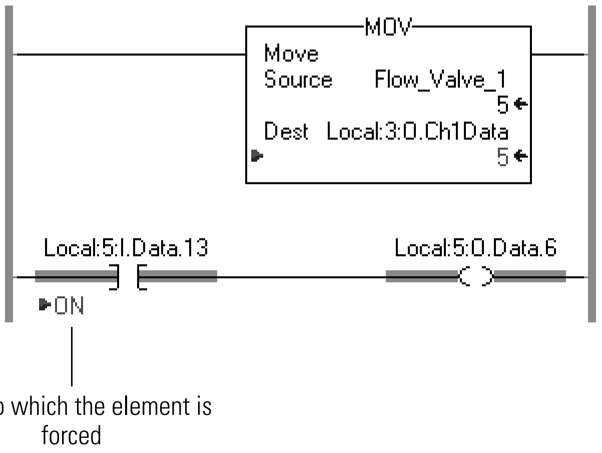
Enable Forces

For a force to take effect, you enable forces. You can only enable and disable forces at the controller level.

- You can enable I/O forces and SFC forces separately or at the same time.
- You cannot enable or disable forces for a specific module, tag collection, or tag element.

IMPORTANT If you download a project that has forces enabled, the programming software prompts you to enable or disable forces after the download completes.

When forces are in effect (enabled), a ▶ appears next to the forced element.



Disable or Remove a Force

To stop the effect of a force and let your project execute as programmed, disable or remove the force.

- You can disable or remove I/O and SFC forces at the same time or separately.
- Removing a force on an alias tag also removes the force on the base tag.

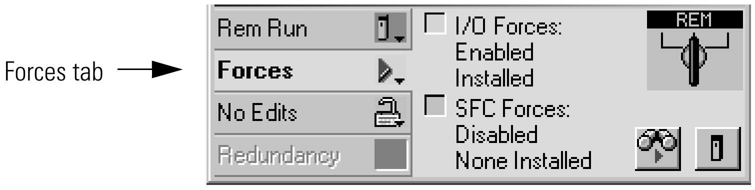
ATTENTION Changes to forces can cause unexpected machine motion that could injure personnel. Before you disable or remove forces, determine how the change will effect your machine or process and keep personnel away from the machine area.

Check Force Status

Before you use a force, determine the status of forces for the controller. You can check force status:

To determine the status of	Use any of the following
I/O forces	• Online toolbar • FORCE LED • GSV instruction
SFC forces	Online toolbar

The Online toolbar shows the status of forces. It shows the status of I/O forces and SFC forces separately.



I/O Forces:
Enabled
Installed
SFC Forces:
Disabled
None Installed

This	Means
Enabled	• If the project contains any forces of this type, they *are* overriding your logic. • If you add a force of this type, the new force immediately takes effect
Disabled	Forces of this type are inactive. If the project contains any forces of this type, they *are not* overriding your logic.
Installed	At least one force of this type exists in the project.
None Installed	No forces of this type exist in the project.

FORCE LED

If your controller has a FORCE LED, use the LED to determine the status of any I/O forces.

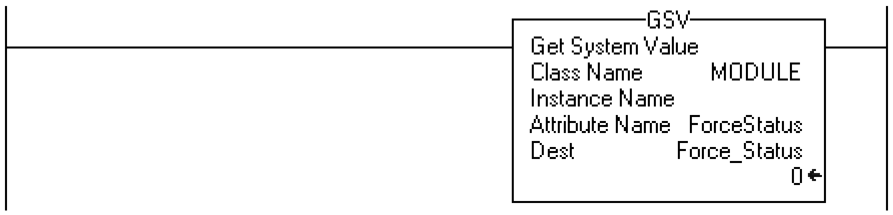
IMPORTANT The FORCE LED shows only the status of I/O forces. It does not show that status of SFC forces.

If the FORCE LED is	Then
Off	• No tags contain force values. • I/O forces are inactive (disabled).
Flashing	• At least one tag contains a force value. • I/O forces are inactive (disabled).
Solid	• I/O forces are active (enabled). • Force values may or may not exist.

GSV Instruction

IMPORTANT The ForceStatus attribute shows only the status of I/O forces. It does not show the status of SFC forces.

This example shows how to use a GSV instruction to get the status of forces.



where:

Force_Status is a DINT tag.

To determine if	Examine this bit	For this value
Forces are installed	0	1
No forces are installed	0	0
Forces are enabled	1	1
Forces are disabled	1	0

When to Use an I/O Force

Use an I/O force to:

- override an input value from another controller (i.e., a consumed tag).
- override an input value from an input device.
- override your logic and specify an output value for another controller (i.e., a produced tag).
- override your logic and specify the state of an output device.

IMPORTANT Forcing increases logic execution time. The more values you force, the longer it takes to execute the logic.

IMPORTANT I/O forces are held by the controller and not by the programming workstation. Forces remain even if the programming workstation is disconnected.

When you force an I/O value:

- You can force all I/O data, except for configuration data.
- If the tag is an array or structure, such as an I/O tag, force a BOOL, SINT, INT, DINT, or REAL element or member.
- If the data value is a SINT, INT, or DINT, you can force the entire value or you can force individual bits within the value. Individual bits can have a force status of:
  - No force
  - Force on
  - Force off
- You can also force an alias to an I/O structure member, produced tag, or consumed tag.
  - An alias tag shares the same data value as its base tag, so forcing an alias tag also forces the associated base tag.
  - Removing a force from an alias tag removes the force from the associated base tag.

Force an Input Value

Forcing an input or consumed tag:

- overrides the value regardless of the value of the physical device or produced tag.
- does not affect the value received by other controllers monitoring that input or produced tag.

Force an Output Value

Forcing an output or produced tag overrides the logic for the physical device or other controller (s). Other controllers monitoring that output module in a listen-only capacity will also see the forced value.

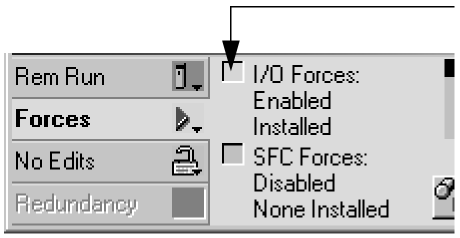
Add an I/O Force

To override an input value, output value, produced tag, or consumed tag, use an I/O force:

ATTENTION

Forcing can cause unexpected machine motion that could injure personnel. Before you use a force, determine how the force will effect your machine or process and keep personnel away from the machine area.

- Enabling I/O forces causes input, output, produced, or consumed values to change.
- If forces are enabled and you install a force, the new force immediately takes effect.



1. What is the state of the I/O Forces indicator?

If	Then note
Off	No I/O forces currently exist.
Flashing	No I/O forces are active. But at least one force already exists in your project. When you enable I/O forces, *all* existing I/O forces will also take effect.
Solid	I/O forces are enabled (active). When you install (add) a force, it immediately takes effect.

2. Open the routine that contains the tag that you want to force.

3. Right-click the tag and choose Monitor… If necessary, expand the tag to show the value that you want to force (e.g., BOOL value of a DINT tag).

4. Install the force value:

To force a	Do this
BOOL value	Right-click the tag and choose Force ON or Force OFF.
Non-BOOL value	In the Force Mask column for the tag, type the value to which you want to force the tag. Then press the Enter key.

5. Are I/O forces enabled? (See step 1.)

If	Then
No	From the Logic menu, choose I/O Forcing > Enable All I/O Forces. Then choose Yes to confirm.
Yes	Stop.

Remove or Disable Forces

ATTENTION

!

Changes to forces can cause unexpected machine motion that could injure personnel. Before you disable or remove forces, determine how the change will effect your machine or process and keep personnel away from the machine area.

If you want to	And	Then
Stop an individual force	Leave other forces enabled and in effect	Remove an Individual Force
Stop all I/O forces but leave all SFC forces active	Leave the I/O forces in the project	Disable All I/O Forces
	Remove the I/O forces from the project	Remove All I/O Forces

Remove an Individual Force

ATTENTION

!

If you remove an individual force, forces remain in the enabled state and any new force immediately takes effect.

Before you remove a force, determine how the change will effect your machine or process and keep personnel away from the machine area.

1. Open the routine that contains the force that you want to remove.
2. What is the language of the routine?

If	Then
SFC	Go to step 4.
Ladder logic	Go to step 4.
Function block	Go to step 3.
Structured text	Go to step 3.

3. Right-click the tag that has the force and choose Monitor…
   If necessary, expand the tag to show the value that is forced, for example, BOOL value of a DINT tag.
4. Right-click the tag or element that has the force and choose Remove Force.

Disable All I/O Forces

From the Logic menu, choose I/O Forcing > Disable All I/O Forces. Then choose Yes to confirm.

Remove All I/O Forces

From the Logic menu, choose I/O Forcing > Remove All I/O Forces. Then choose Yes to confirm.

How Are We Doing?

Your comments on our technical publications will help us serve you better in the future.
Thank you for taking the time to provide us feedback.

You can complete this form and mail (or fax) it back to us or email us at RADocumentComments@ra.rockwell.com.

Pub. Title/Type Logix5000 Controllers I/O and Tag Data

Cat. No.	*1756 ControlLogix, 1768 CompactLogix, 1769 CompactLogix, 1789 SoftLogix, 1794 FlexLogix, PowerFlex 700S with DriveLogix*	Pub. No.	1756-PM004B-EN-P	Pub. Date	July 2008	Part No.	None

Please complete the sections below. Where applicable, rank the feature (1=needs improvement, 2=satisfactory, and 3=outstanding).

Overall Usefulness	1 2 3	How can we make this publication more useful for you?
Completeness (all necessary information is provided)	1 2 3	Can we add more information to help you? procedure/step, illustration, feature, example, guideline, other, explanation, definition
Technical Accuracy (all provided information is correct)	1 2 3	Can we be more accurate? text, illustration
Clarity (all provided information is easy to understand)	1 2 3	How can we make things clearer?
Other Comments		You can add additional comments on the back of this form.

Your Name

Your Title/Function

Location/Phone

Would you like us to contact you regarding your comments?

___No, there is no need to contact me

___Yes, please call me

___Yes, please email me at ____________________

___Yes, please contact me via ____________________

Return this form to: Rockwell Automation Technical Communications, 1 Allen-Bradley Dr., Mayfield Hts., OH 44124-9705

Fax: 440-646-3525 Email: RADocumentComments@ra.rockwell.com

PLEASE FASTEN HERE (DO NOT STAPLE)

Other Comments

PLEASE FOLD HERE

PLEASE REMOVE

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 18235 CLEVELAND OH

POSTAGE WILL BE PAID BY THE ADDRESSEE

Rockwell Automation

1 ALLEN-BRADLEY DR
MAYFIELD HEIGHTS OH 44124-9705

Rockwell Automation Support

Rockwell Automation provides technical information on the Web to assist you in using its products. At http://support.rockwellautomation.com, you can find technical manuals, a knowledge base of FAQs, technical and application notes, sample code and links to software service packs, and a MySupport feature that you can customize to make the best use of these tools.

For an additional level of technical phone support for installation, configuration, and troubleshooting, we offer TechConnect support programs. For more information, contact your local distributor or Rockwell Automation representative, or visit http://support.rockwellautomation.com.

Installation Assistance

If you experience a problem within the first 24 hours of installation, please review the information that's contained in this manual. You can also contact a special Customer Support number for initial help in getting your product up and running.

United States	1.440.646.3434 Monday – Friday, 8am – 5pm EST
Outside United States	Please contact your local Rockwell Automation representative for any technical support issues.

New Product Satisfaction Return

Rockwell Automation tests all of its products to ensure that they are fully operational when shipped from the manufacturing facility. However, if your product is not functioning and needs to be returned, follow these procedures.

United States	Contact your distributor. You must provide a Customer Support case number (call the phone number above to obtain one) to your distributor in order to complete the return process.
Outside United States	Please contact your local Rockwell Automation representative for the return procedure.

www.rockwellautomation.com

Power, Control and Information Solutions Headquarters

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, Vorstlaan/Boulevard du Souverain 36, 1170 Brussels, 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

Publication 1756-PM004B-EN-P - July 2008

Supersedes Publication 1756-PM004A-EN-P - July 2007