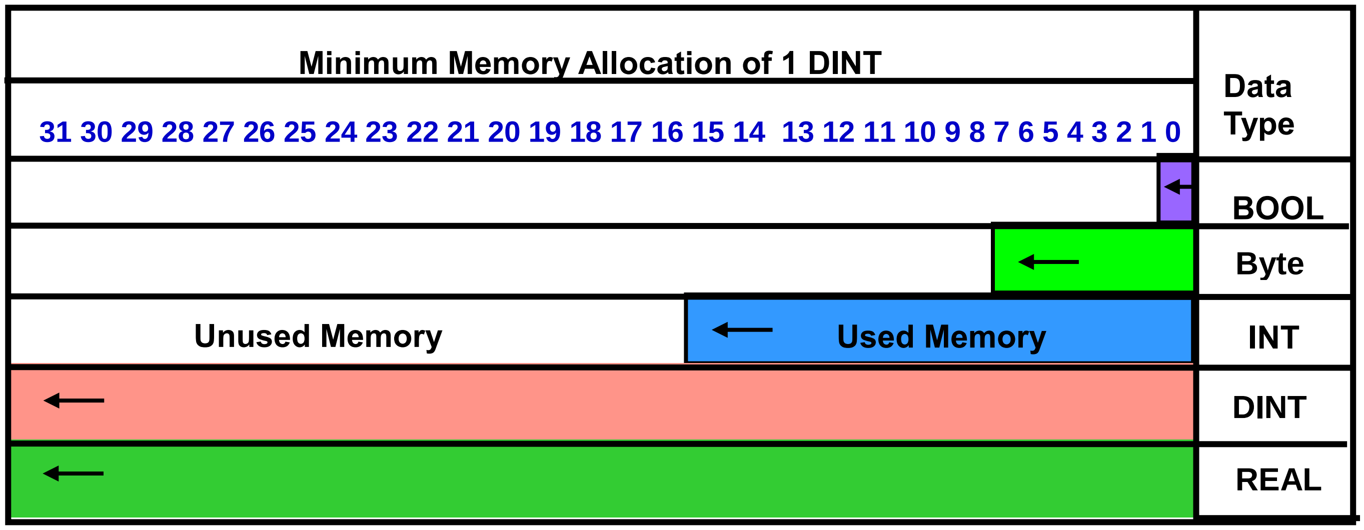
Creating Tags

A conventional PLC uses addresses for data location, and symbols for names. Both configured separately. They also use Data files for storage. Each data file represents a specific type of data. For instance, there is a file for integers, binary, timers and counters etc.

Controllogix is different. There are no data files. Memory is defined by creating **Tags.**

Important: The main tag type in a controllogix project is a Double Integer (DINT).

This is 32 bit and is the **minimum** amount of memory that a tag will use, even if it is a single **BOOL** tag.



Shown in the illustration above is how memory is allocated for a tag of a specific data type. As can be seen, a BOOL tag uses as much memory as a DINT tag. The memory available within a controllogix controller is around 8 Meg so memory usage is not normally a factor. If it is important to save memory, Tag Arrays can be created.

When tags are created, it utilizes the next available area of memory. There are no pre- defined data files.

Name. The tag must be given a name. The name is not case sensitive. When the space bar is pressed between words, an underscore is automatically inserted.

Data Type. There are a number of data types available within logix 5000 such as BOOL, DINT, TIMER, COUNTER and many more. A **User Defined** data type can also be created which may consist of a number of members.

Commonly Used Data Types

The data types shown below are all of single members and are used on a regular basis.

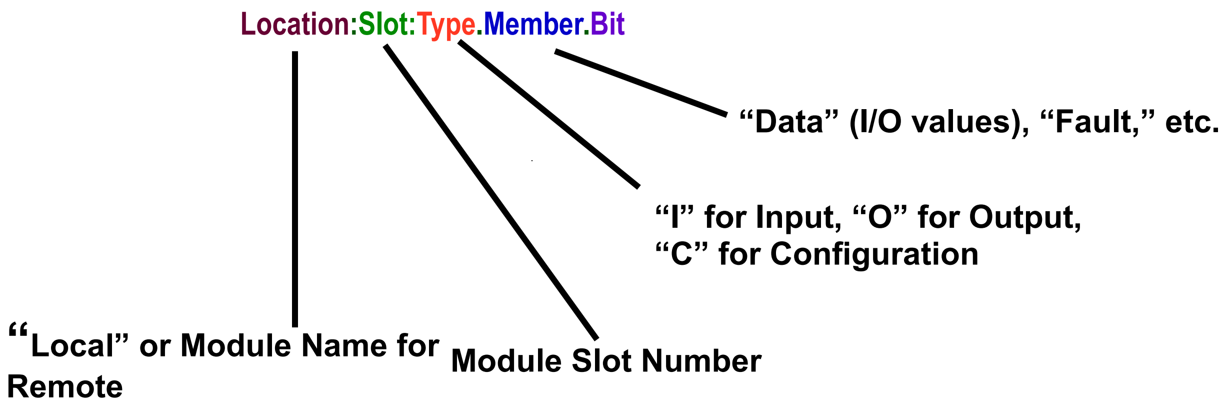
Data Type	Functional Description
BOOL	A single bit where 1 = on and 0 = off (e.g., the state of a pushbutton or sensor)
SINT(Byte)	A short integer (8 bits) between -128 and +127
INT	An integer or word (16 bits) between -32,768 and + 32,767
DINT	A double integer (32 bits), used to store a base integer number in the range of -2,147,483,648 to + 2,147,483,647
LINT	A 64-bit signed integer data type used to represent wall clock time
REAL	A 32-bit floating point value with decimal point.
STRING	A data type that holds character data

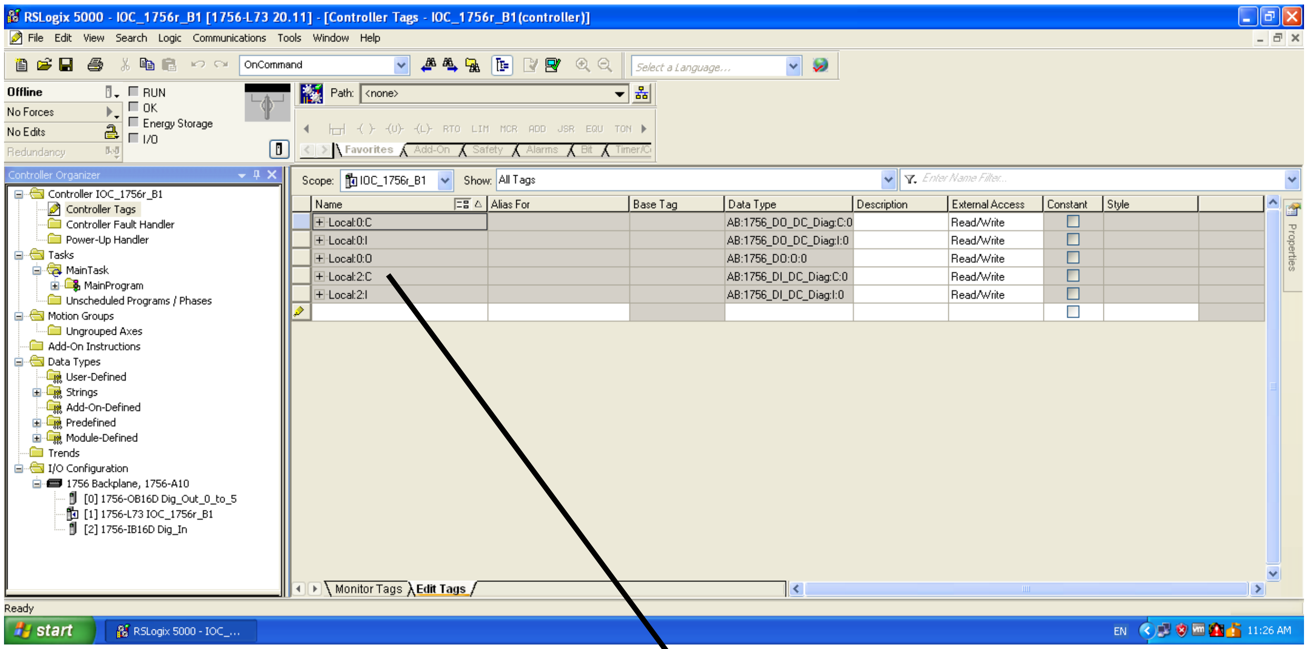
Data Types made up as a Structure

The data types shown below are made up of members of different data types. A timer for example has 5 members. ACC, PRE, both DINT's. EN, TT and DN are all BOOL types.

Data Type	Functional Description
Counter	An Increasing or decreasing total
Timer	Increasing time total in (milliseconds)
Control	Length and position for file level instructions
Message	Control structure for a message instruction

I/O Tags. When an I/O module is created, tags are automatically created at Controller scope. The format is shown below.


Location:Slot:Type.Member.Bit
"Data" (I/O values), "Fault," etc.
"I" for Input, "O" for Output, "C" for Configuration
"Local" or Module Name for Remote
Module Slot Number



RSLogix 5000 - IOC_1756r_B1 [1756-L73 20.11] - [Controller Tags - IOC_1756r_B1(controller)]

File Edit View Search Logic Communications Tools Window Help

Offline · No Forces · No Edits · Redundancy · RUN · OK · Energy Storage · I/O · Path: <none>

Controller Organizer
- Controller IOC_1756r_B1
  - Controller Tags
  - Controller Fault Handler
  - Power-Up Handler
- Tasks
  - MainTask
    - MainProgram
  - Unscheduled Programs / Phases
- Motion Groups
  - Ungrouped Axes
- Add-On Instructions
- Data Types
  - User-Defined
  - Strings
  - Add-On-Defined
  - Predefined
  - Module-Defined
- Trends
- I/O Configuration
  - 1756 Backplane, 1756-A10
    - [0] 1756-OB16D Dig_Out_0_to_5
    - [1] 1756-L73 IOC_1756r_B1
    - [2] 1756-IB16D Dig_In

Scope: IOC_1756r_B1 Show: All Tags

Name	Alias For	Base Tag	Data Type	Description	External Access	Constant	Style
+ Local:0:C			AB:1756_DO_DC_Diag:C:0		Read/Write	☐	
+ Local:0:I			AB:1756_DO_DC_Diag:I:0		Read/Write	☐	
+ Local:0:O			AB:1756_DO:O:0		Read/Write	☐	
+ Local:2:C			AB:1756_DI_DC_Diag:C:0		Read/Write	☐	
+ Local:2:I			AB:1756_DI_DC_Diag:I:0		Read/Write	☐	

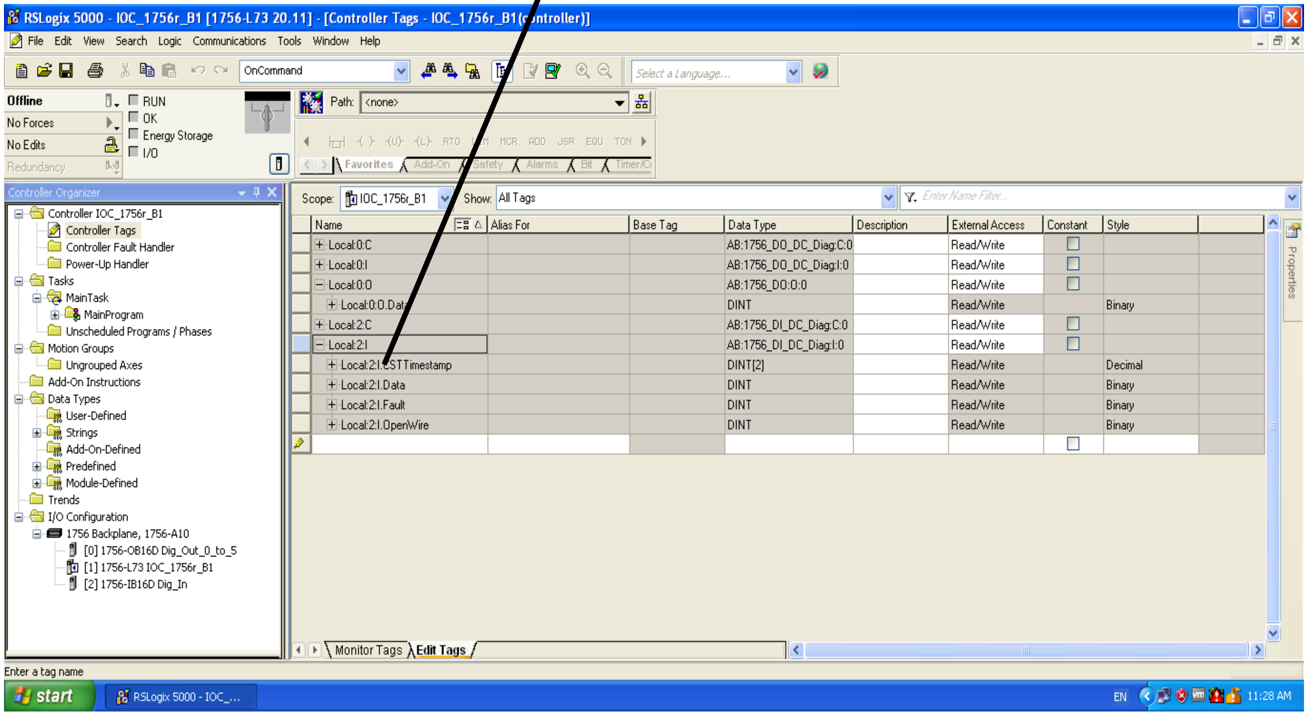
Monitor Tags / Edit Tags

Ready

start · RSLogix 5000 - IOC_... · EN · 11:26 AM

Default Tag Names for configured modules

Modules Expanded



RSLogix 5000 - IOC_1756r_B1 [1756-L73 20.11] - [Controller Tags - IOC_1756r_B1(controller)]

File Edit View Search Logic Communications Tools Window Help

Offline · No Forces · No Edits · Redundancy · RUN · OK · Energy Storage · I/O · Path: <none>

Controller Organizer
- Controller IOC_1756r_B1
  - Controller Tags
  - Controller Fault Handler
  - Power-Up Handler
- Tasks
  - MainTask
    - MainProgram
  - Unscheduled Programs / Phases
- Motion Groups
  - Ungrouped Axes
- Add-On Instructions
- Data Types
  - User-Defined
  - Strings
  - Add-On-Defined
  - Predefined
  - Module-Defined
- Trends
- I/O Configuration
  - 1756 Backplane, 1756-A10
    - [0] 1756-OB16D Dig_Out_0_to_5
    - [1] 1756-L73 IOC_1756r_B1
    - [2] 1756-IB16D Dig_In

Scope: IOC_1756r_B1 Show: All Tags

Name	Alias For	Base Tag	Data Type	Description	External Access	Constant	Style
+ Local:0:C			AB:1756_DO_DC_Diag:C:0		Read/Write	☐	
+ Local:0:I			AB:1756_DO_DC_Diag:I:0		Read/Write	☐	
− Local:0:O			AB:1756_DO:O:0		Read/Write	☐	
+ Local:0:O.Data			DINT		Read/Write		Binary
+ Local:2:C			AB:1756_DI_DC_Diag:C:0		Read/Write	☐	
− Local:2:I			AB:1756_DI_DC_Diag:I:0		Read/Write	☐	
+ Local:2:I.CSTTimestamp			DINT[2]		Read/Write		Decimal
+ Local:2:I.Data			DINT		Read/Write		Binary
+ Local:2:I.Fault			DINT		Read/Write		Binary
+ Local:2:I.OpenWire			DINT		Read/Write		Binary

Monitor Tags / Edit Tags

Enter a tag name

start · RSLogix 5000 - IOC_... · EN · 11:28 AM

There are 2 scopes for creating tags.

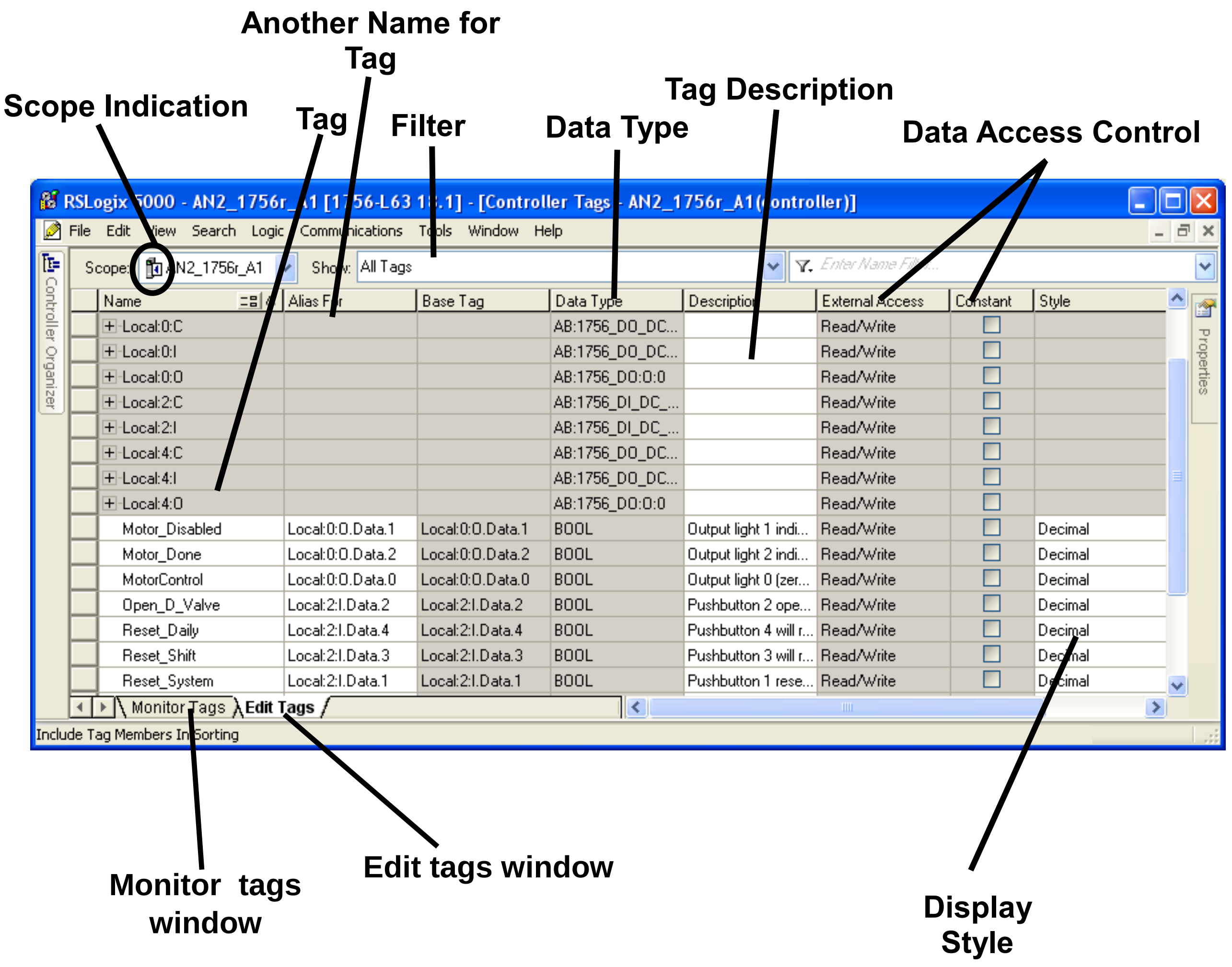
Program Scope. Tags created at program scope are only available to be used by the routines within the program where they are created.

Controller Scope. Tags created at controller scope can be used by any program and routine within the project. Do not create tags at program scope with the same name as tags at controller scope. This will cause a conflict.

Tags can be created in the tags display window or a new tag can be created at the ladder instruction that it will reference.

An **Alias Tag** is another name for an existing tag. Used extensively to give tag names to I/O points as an I/O tag name does not relate to a project component.

Shown below is the tags collection window where all tags can be created, edited, deleted and monitored. There are separate windows for program tags and controller tags.



Scope: AN2_1756r_A1 Show: All Tags

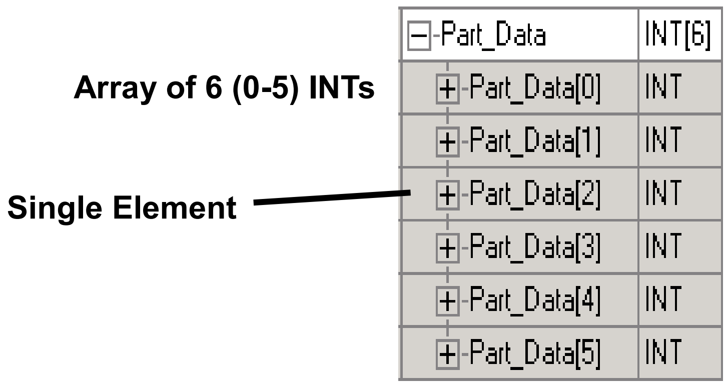
Name	Alias For	Base Tag	Data Type	Description	External Access	Constant	Style
+ Local:0:C			AB:1756_DO_DC...		Read/Write	☐	
+ Local:0:I			AB:1756_DO_DC...		Read/Write	☐	
+ Local:0:O			AB:1756_DO:O:0		Read/Write	☐	
+ Local:2:C			AB:1756_DI_DC_...		Read/Write	☐	
+ Local:2:I			AB:1756_DI_DC_...		Read/Write	☐	
+ Local:4:C			AB:1756_DO_DC...		Read/Write	☐	
+ Local:4:I			AB:1756_DO_DC...		Read/Write	☐	
+ Local:4:O			AB:1756_DO:O:0		Read/Write	☐	
Motor_Disabled	Local:0:O.Data.1	Local:0:O.Data.1	BOOL	Output light 1 indi...	Read/Write	☐	Decimal
Motor_Done	Local:0:O.Data.2	Local:0:O.Data.2	BOOL	Output light 2 indi...	Read/Write	☐	Decimal
MotorControl	Local:0:O.Data.0	Local:0:O.Data.0	BOOL	Output light 0 (zer...	Read/Write	☐	Decimal
Open_D_Valve	Local:2:I.Data.2	Local:2:I.Data.2	BOOL	Pushbutton 2 ope...	Read/Write	☐	Decimal
Reset_Daily	Local:2:I.Data.4	Local:2:I.Data.4	BOOL	Pushbutton 4 will r...	Read/Write	☐	Decimal
Reset_Shift	Local:2:I.Data.3	Local:2:I.Data.3	BOOL	Pushbutton 3 will r...	Read/Write	☐	Decimal
Reset_System	Local:2:I.Data.1	Local:2:I.Data.1	BOOL	Pushbutton 1 rese...	Read/Write	☐	Decimal

Tag Arrays

When a tag is created, it will occupy an area of memory. With tags smaller than a DINT, a full DINT will be used.

One way of saving memory is to create an Array. This is a tag made up of so many elements of the same data type. An array element will occupy a consecutive area of memory which has the effect of saving memory. For instance, an array of BOOL with 32 elements, would only occupy one DINT. Separate BOOL tags would occupy 32 DINT's.

An example is shown below.



Part_Data	INT[6]
Part_Data[0]	INT
Part_Data[1]	INT
Part_Data[2]	INT
Part_Data[3]	INT
Part_Data[4]	INT
Part_Data[5]	INT

Below it can be seen how a 6 element array of INT, occupies 3 DINT's, but if 6 separate tags were created, it would occupy 6 DINT's.

Part_Data[1]	**Part_Data[0]**
Part_Data[3]	**Part_Data[2]**