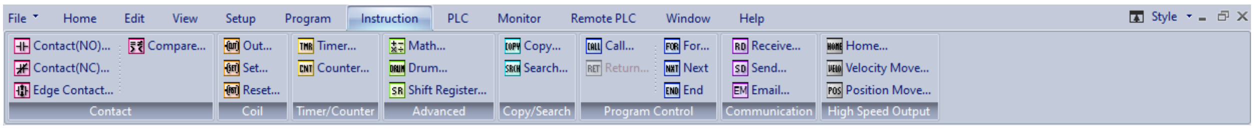
Instruction Menu

Description

The **Instruction Menu** is located on the [Main Menu]() of this application.



Contact Instructions

The **Contact Instructions** are used to provide an **Input** to **Activate** or **Deactivate** a rung. These **Contacts** are basically a virtual **ON/OFF** switch that can be configured to operate as needed. The **CLICK Programming Software** offers four **Contact** choices. To learn more about these **Contacts**, move the pointer over the **Icon** to see the name and click on the **Icon** to open the specific topic for that **Contact**.

Coil Instructions

The **Coil Instructions** are used to provide an **Output** that responds directly to the state of the rung. The **CLICK Programming Software** currently offers four **Coil** choices. To learn more about these **Coils**, move the pointer over the **Icon** to see the name and click on the **Icon** to open the specific topic for that **Coil**.

OUT SET RST

Timer/Counter Instructions

The **Timer Instruction** is used to provide an **Output** that responds according to a preset time value. The **Timer** starts timing when the when the rung is energized and can be configured to activate or deactivate the output when the preset time cycle is completed. The **Counter Instruction** is used to provide an **Output** that responds according to a preset count value. The **Counter** counts the **ON/OFF** transitions of the enabling rung to activate or deactivate the output when the preset count is completed.

To learn more about the **Timer** or **Counter Instructions**, move the pointer over the **Icon** to see the name and click on the **Icon** to open the specific topic for the instruction.

TMR CNT

Advanced Instructions

The **Advanced Instructions** include instructions that provide an **Output** based on mathematical calculations, sequences triggered by events or time, or register shifts triggered by a bit status depending how its been configured. The **CLICK Programming Software** offers three **Advanced Instruction** choices. To learn more about these instructions, move the pointer over the **Icon** to see the name and click on the **Icon** to open the specific topic for that instruction.

+ − × ÷ | DRUM | SR

Copy/Search Instructions

The **Copy/Search Instructions** include instructions that copy or search within registers. The **CLICK Programming Software** offers three **Copy/Search Instruction** choices. To learn more about these instructions, move the pointer over the **Icon** to see the name and click on the **Icon** to open the specific topic for that instruction.

COPY | SRCH

Program Control Instructions

The **Program Control Instructions** include instructions that control program flow. The **CLICK Programming Software** offers five **Program Control Instruction** choices. To learn more about these instructions, move the pointer over the **Icon** to see the name and click on the **Icon** to open the specific topic for that instruction.

CALL | RET | FOR | NXT | END

Communication Instructions

The **Communication Instructions** include instructions that exchange data with other systems. The **CLICK Programming Software** offers three **Communication Instruction** choices. To learn more about these instructions, move the pointer over the **Icon** to see the name and click on the **Icon** to open the specific topic for that instruction.

RD | SD | EM

High Speed Output Instructions

The **CLICK Programming Software** offers three **High Speed Output Instruction** choices. To learn more about these instructions, move the pointer over the **Icon** to see the name and click on the **Icon** to open the specific topic for that instruction.

HOME VELO POS

Contact Instructions

Description

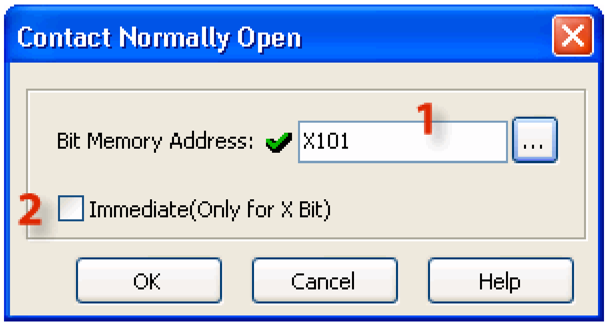
The **Contact Instructions** are used to provide an **Input** to **Activate** or **Deactivate** a rung. These **Contacts** are basically a virtual **ON/OFF** switch that can be configured to operate as needed. The **CLICK Programming Software** offers four **Contact** choices. To learn more about these **Contacts**, move the pointer over the **Icon** to see the name and click on the **Icon** to open the specific topic for that **Contact**.

Normally Open Contact

Definition

The **Normally Open Contact** mimics the behavior of a physical contact and changes in response to the status of a Bit Memory Address. The **Normally Open Contact** is **ON** when the related bit is **ON**.

Setup



Contact Normally Open

Bit Memory Address: ✓ X101 1

2 ☐ Immediate(Only for X Bit)

✓ **Entry required or invalid entry** ✓ **Valid entry**

1 **Bit Memory Address:** The **Normally Open Contact** requires a Bit Memory Address. The **Bit Memory Address** can be typed directly into the Address field on the dialog or it can be selected from the **Address Picker**.

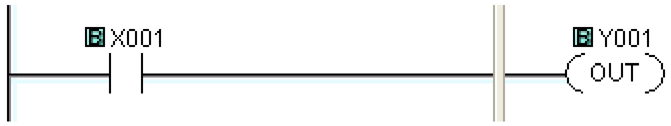
Note: Click the **Browse Button** ⋯ to open the Address Picker.

2 Immediate (Only for X Bit): Use this checkbox **ONLY** for **X Bit Addresses**. The status of all **X Addresses** are updated by reading the status of physical input points in the beginning of each program scan. If the program scan time is getting long, you may want to read the current status of a physical input point when this instruction is executed. In this case, use the **Immediate checkbox** to declare the **X Input** to be a **Normally Open Immediate Contact**. This option is grayed out until an **X address** is chosen.

Example Program

Example Program

In the following example, when **X001** is **ON**, **Y001** will also be **ON**.



Related Topics:

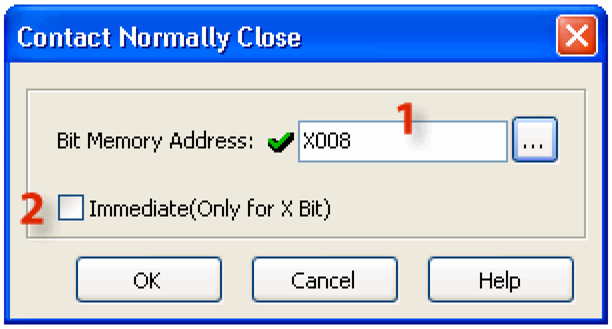
Instruction List
Checkmarks
Memory Addresses

Normally Closed Contact

Definition

The **Normally Closed Contact** mimics the behavior of a physical contact and changes in response to the status of a Bit Memory Address. The **Normally Closed Contact** is **ON** when the related bit is **OFF**.

Setup



✔ Entry required or invalid entry ✔ Valid entry

1 Bit Memory Address: The **Normally Closed Contact** requires a [Bit Memory Address](). The **Bit Memory Address** can be typed directly into the Address field on the dialog or it can be selected from the **Address Picker**.

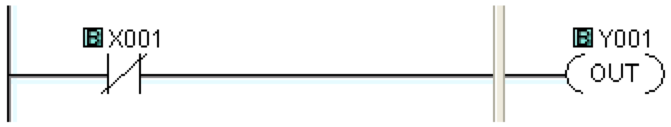
Click on the **Browse Button**… to open the [Address Picker]().

2 Immediate (Only for X Bit): Use this checkbox **ONLY** for **X Bit Addresses**. The status of all **X Addresses** are updated by reading the status of physical input points in the beginning of each program scan. If the program scan time is getting long, you may want to read the current status of a physical input point when the instruction is executed. In this case, use the **Immediatecheckbox** to declare the **X Input** to be a **Normally Closed ImmediateContact**. This option is grayed out until an **X address** is chosen.

Example Program

Example Program

In the following example, when **X001** is **OFF**, **Y001** will be **ON**.



<u>Related Topics:</u>

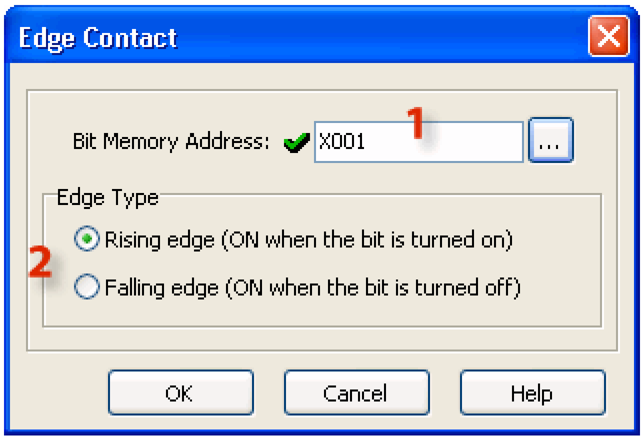
[Instruction List]()
[Checkmarks]()

Shortcuts: Rising Edge = SHIFT + F2 / Falling Edge = SHIFT + F3

Definition

The **Edge Contact** turns **ON** when the related bit transitions from **OFF** to **ON** (**Rising Edge**) or **ON** to **OFF** (**Falling Edge**).

Setup



Edge Contact

1

Bit Memory Address: ✓ X001

Edge Type

2

[✓ Rising edge (ON when the bit is turned on)]

Falling edge (ON when the bit is turned off)

✓ Entry required or invalid entry ✓ Valid entry

1 Bit Memory Address: The **Edge Contact** requires a [Bit Memory Address](). The **Bit Memory Address** can be typed directly into the Address field on the dialog or it can be selected from the **Address Picker**.

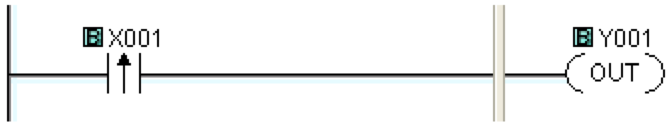
Click on the **Browse Button** … to open the [Address Picker]().

2 Edge Type: Select **Rising Edge** or **Falling Edge** contact using the radio buttons on the dialog.

Example Programs

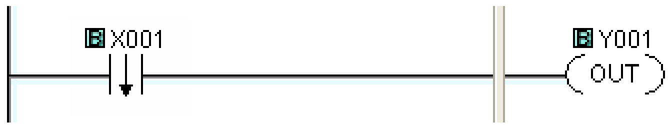
Example Program 1

In the following example, each time **X001** makes an **OFF-to-ON transition**, **Y001** will energize for **one scan**.



Sample Program 2

In the following example, each time **X001** makes an **ON-to-OFF transition**, **Y001** will energize for **one scan**.



Related Topics:

Instruction List
Checkmarks
Memory Addresses

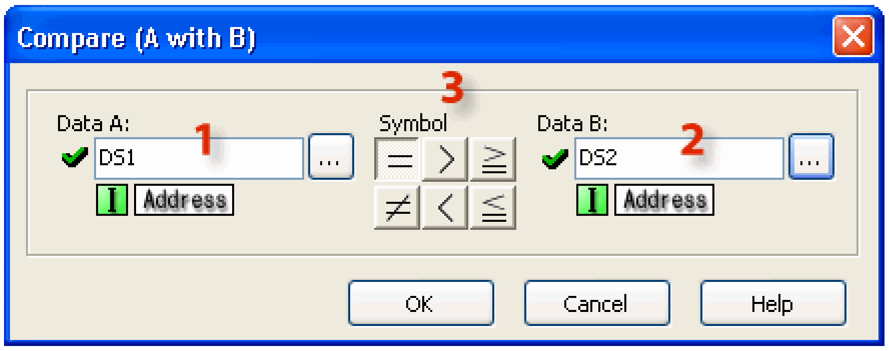
Compare Contact

Definition

The **Compare A with B** instruction uses a **Mathematical Operator** as a basis of comparison of two data values. A mathematical statement is developed using the instruction dialog. When the Data A and Data B values satisfy the selected mathematical relationship, the **Compare Contact** turns **ON**.

Setup

The fields **Data A** and **Data B** can contain memory addresses or **Constants**. **Constants** are created by typing the value directly into the **Data A** or **Data B** fields. Use the typing guidelines for the Data Type you are selecting. See the available **Data** combination to determine if two data types can be compared.



Data A: DS1

Data B: DS2

1 Data A: Select the first **Data Memory Address** or the **Constant** to be compared. The **Data Memory Address** or the **Constant** can be typed directly into the **Address** field on the dialog or the **Data Memory Address** can be selected from the **Address Picker**.

Click on the **Browse Button** ⋯ to open the Address Picker.

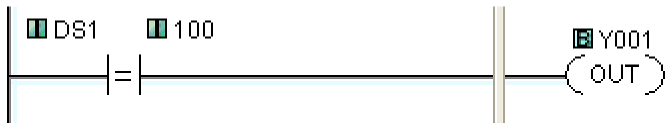
2 Data B: Select the second **Data Memory Address** or the **Constant** to be compared.

3 Symbol: Select the **Mathematical Operator** that defines the intended relationship between **Data A** and **Data B**.

Example Programs

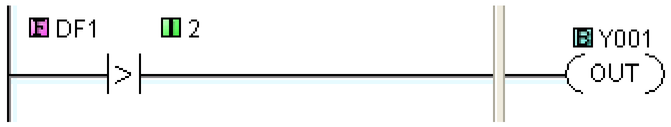
Example Program 1: Compare Integer Data

In the following example, when **DS1** is **100**, **Y001** will energize.



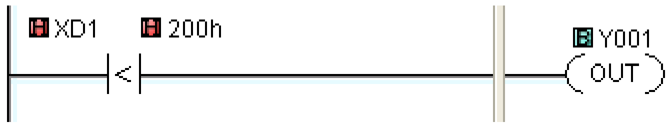
Example Program 2: Compare floating point data

In the following example, when **DF1** is more than **2**, **Y001** will energize. As you can see, you can compare floating point data with integer or floating point data.



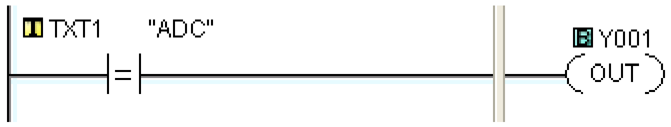
Example Program 3: Compare Hex data

In the following example, when **XD1** is less than **200h**, **Y001** will energize.



Example Program 4: Compare Text data

In the following example, when **TXT1** to **TXT3** are **"ADC"** (TXT1 = "A", TXT2 = "D", TXT3 = "C"), **Y001** will energize.



Related Topics:

[Data Types](#)
[Data Compatibility Table](#)
[Memory Addresses](#)

Coil Instructions

Description

The **Coil Instructions** are used to provide an **Output** that responds directly to the state of the rung. The **CLICK Programming Software** currently offers four **Coil** choices. To learn more about these **Coils**, move the pointer over the **Icon** to see the name and click on the **Icon** to open the specific topic for that **Coil**.

-(OUT) -(SET) -(RST)

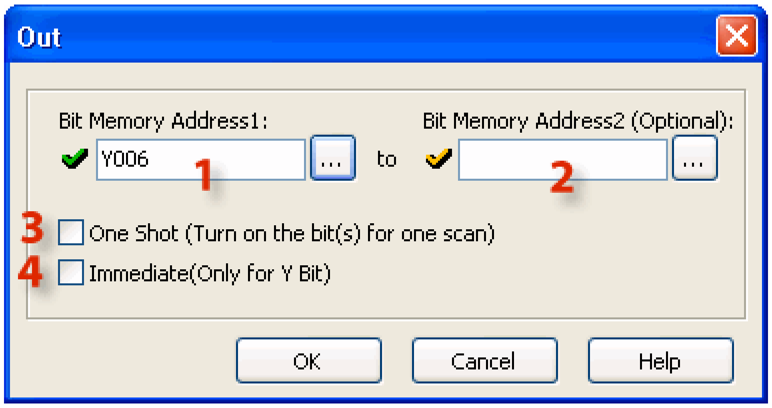
Topic: CL167

Out Coil

Definition

An **Out** instruction turns **ON** its associated **Bit Memory** when the status of the rung is true.
The **Out** instruction turns **OFF** its associated **Bit Memory** when the status of the rung is false.
An **Out** instruction is capable of turning **ON** and **OFF** more than one **Bit Memory Address** at the same time.

Setup



Bit Memory Address1: ✓ Y006 to Bit Memory Address2 (Optional): ✓

☐ One Shot (Turn on the bit(s) for one scan)
☐ Immediate(Only for Y Bit)

✓ Entry required or invalid entry ✓ Optional ✓ Valid entry

1Bit Memory Address 1: Choose a **Bit Memory Address** to associate with the **Out** instruction. Either type the **Bit Memory Address** directly in the address field or use the **Browse Button** to open the **Address Picker**.

2 Bit Memory Address 2: The **Out** instruction permits a consecutive range of bit memory addresses to change state simultaneously. Choose a **Bit Memory Address 2** to identify the end of the range of **Bit Memory Addresses** or leave this field blank if you only want one output to respond to this instruction.

Note: Use this feature with care since it is permitted to reuse **Bit Memory Addresses** within the range specified.

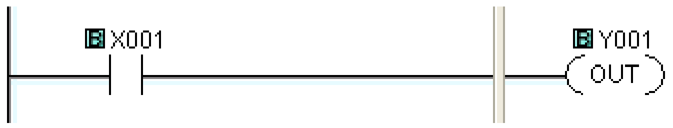
3One Shot: Choose **One Shot** to cause the **Bit Memory Address** to turn **ON** for one scan only or leave unchecked to allow the **Bit Memory Address** to be processed during each scan. If selected, the **One Shot Icon** will appear adjacent to the **Coil** in the **Ladder Editor**.

4 Immediate: Choose **Immediate** if you want this bit processed asynchronously in relation to the scan. This feature is only available for actual **Y outputs**. This option is grayed out until a **Y** address is chosen. The **Immediate Icon** will appear adjacent to the **Coil** in the **Ladder Editor**.

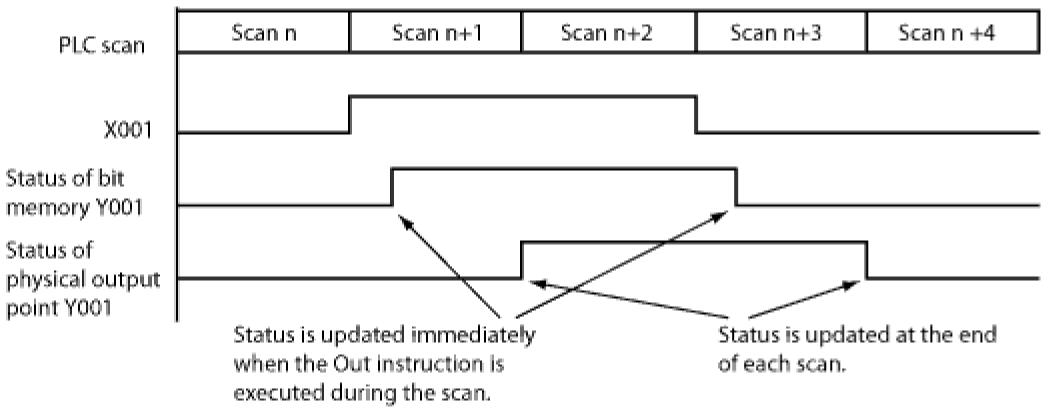
Example Programs

Example Program 1: Default (No option)

In the following example, when input status bit **X001** is **ON**, the **Out** instruction turns **ON** output status bit **Y001** immediately. However, the **CLICK PLC** updates the physical output point (**Y001** is the 1st output point on the **CPU** module) at the end of each scan. Please refer to the timing diagram to see the difference.



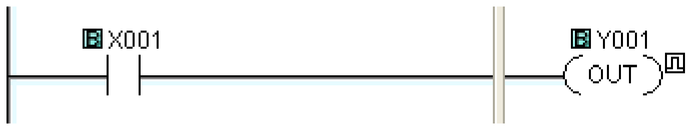
Timing Diagram



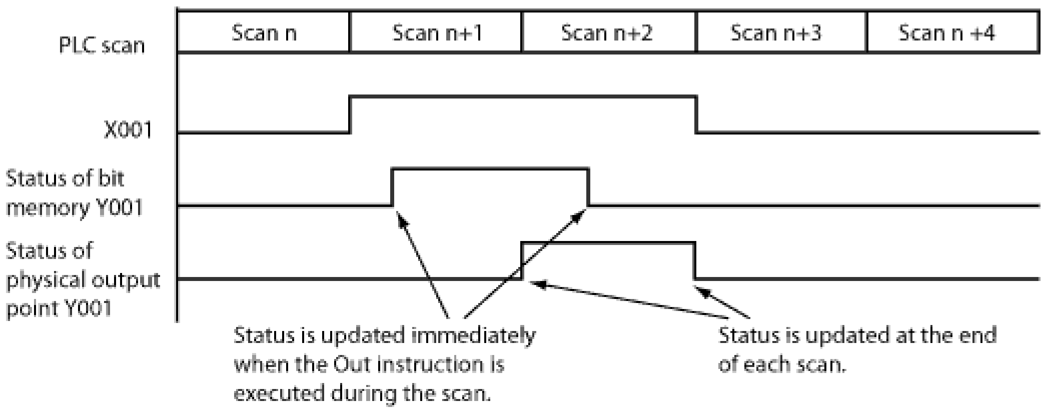
PLC scan
Scan n
Scan n+1
Scan n+2
Scan n+3
Scan n +4
X001
Status of bit memory Y001
Status of physical output point Y001
Status is updated immediately when the Out instruction is executed during the scan.
Status is updated at the end of each scan.

Example Program 2: One Shot

In the following example, when input status bit **X001** is **ON**, the **Out** instruction turns **ON** output status bit **Y001** for one scan. Please refer to the timing diagram to see the difference.



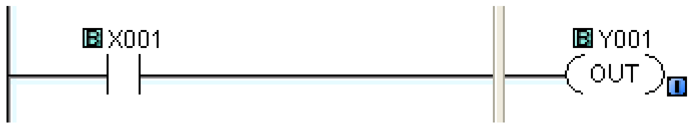
Timing Diagram



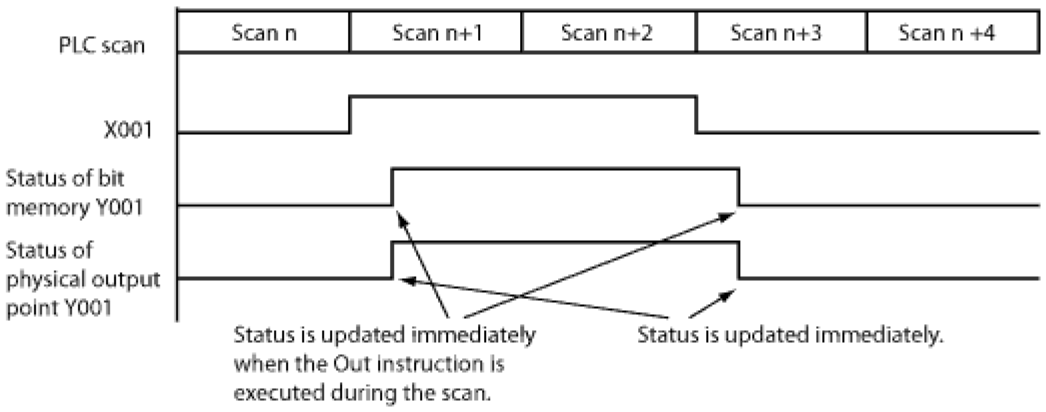
PLC scan
Scan n
Scan n+1
Scan n+2
Scan n+3
Scan n +4
X001
Status of bit memory Y001
Status of physical output point Y001
Status is updated immediately when the Out instruction is executed during the scan.
Status is updated at the end of each scan.

Example Program 3: Immediate (Available only for the Y memory type)

In the following example, when input status bit **X001** is **ON**, the **Out** instruction turns **ON** both the output status bit **Y001** and the physical output point (**Y001** is the 1st output point on the **CPU** module) immediately. Please refer to the timing diagram to see the difference.



Timing Diagram


PLC scan
Scan n
Scan n+1
Scan n+2
Scan n+3
Scan n +4
X001
Status of bit memory Y001
Status of physical output point Y001
Status is updated immediately when the Out instruction is executed during the scan.
Status is updated immediately.

Topic: CL168

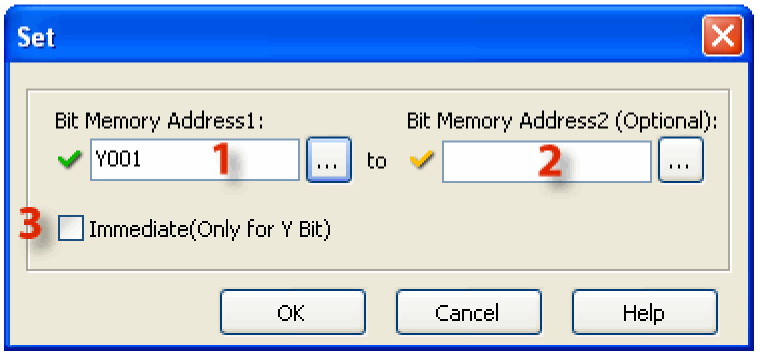
Set Coil

Send Feedback

CLICK

Definition

The **Set** instruction turns **ON** the associated **Bit Memory** when the status of the rung is **True**. The **Bit Memory** stays on after the rung becomes **False**. The **Set** instruction can turn **ON** more than one **Bit Memory** at the same time.



✓ Entry required or invalid entry ✓ Optional ✓ Valid entry

Setup

1 Bit Memory Address1: Choose a **Bit Memory Address** to associate with the **Set** instruction. Either type the **Bit Memory Address** directly in the address field or use the **Browse Button** ... to open the Address Picker.

2 Bit Memory Address2 (Optional): The **Set** instruction permits a consecutive range of bit memory addresses to be **Set** simultaneously. Choose a **Bit Memory Address** to identify the end of the range of **Bit Memory Addresses** or leave this field blank if you only want just one **Bit Memory Address** to respond to this instruction.

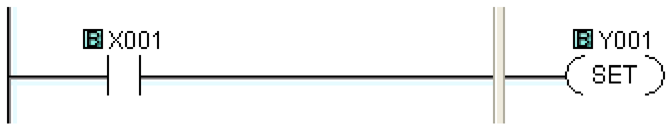
3 Immediate (Only for Y Bit): Choose **Immediate** if you want to turn on the **Output(s)** immediately. This feature is only available for actual **Y outputs**. This option is grayed out until a **Y** address is chosen. The **Immediate Icon** will appear adjacent to the **Set Coil** in the **Ladder Editor**.

Note: Each **Set Coil** in your **Ladder Program** should be accompanied by a **Reset Coil** to turn **OFF** the addresses that were turned **ON** by the **Set Coil**.

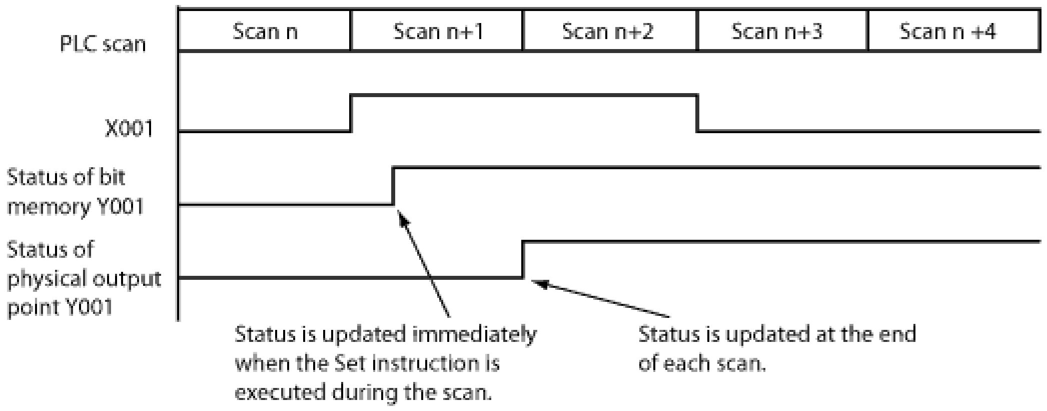
Example Programs

Example Program 1: Default

In the following example, when input status bit **X001** is **ON**, the Set instruction turns **ON** the output status bit **Y001** immediately. After that, **Y001** stays **ON** even if **X001** is turned **OFF**.

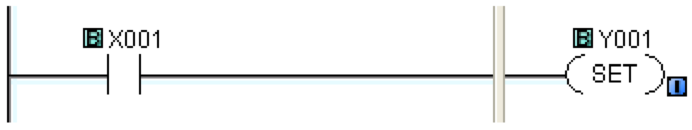


Timing Diagram

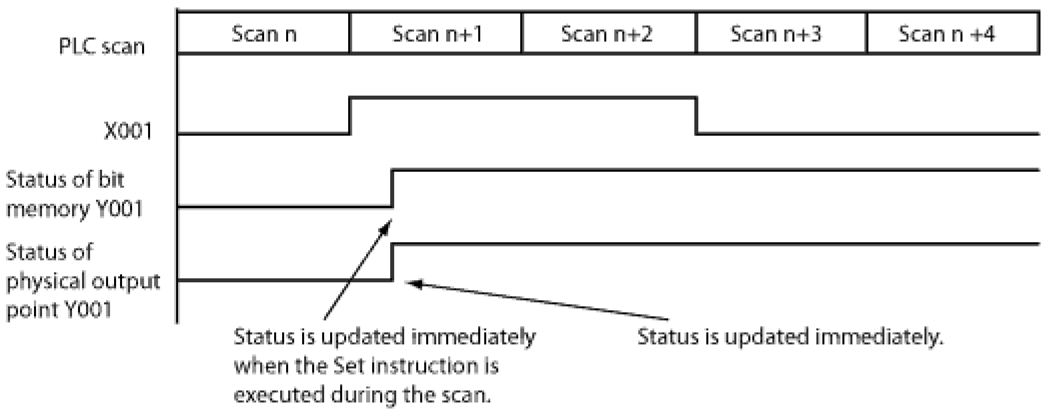


Example Program 2: Immediate (Available only for the Y memory type)

In the following example, when input status bit **X001** is **ON**, the **Set** instruction turns **ON** both the output status bit **Y001** and the physical output point (**Y001** is the 1st output point on the **CPU** module) immediately. Please refer to the timing diagram to see the difference.



Timing Diagram


PLC scan
Scan n
Scan n+1
Scan n+2
Scan n+3
Scan n +4
X001
Status of bit memory Y001
Status of physical output point Y001
Status is updated immediately when the Set instruction is executed during the scan.
Status is updated immediately.

Related Topics:

Reset Coil
Memory Addresses

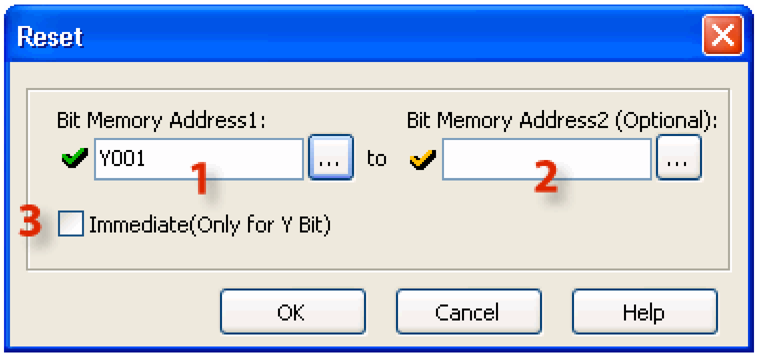
Topic: CL169

Reset Coil

Send Feedback

Definition

The **Reset** instruction shall turn off the associated bit memory when the status of the rung is true. The Bit Memory stays OFF after the rung becomes False. A **Reset** instruction shall turn off more than one bit memory at the same time.



✓ Entry required or invalid entry ✓ Optional ✓ Valid entry

Setup

1 Bit Memory Address 1: Choose a **Bit Memory Address** to associate with the **Reset** instruction. Either type the **Bit Memory Address** directly in the address field or use the **Browse Button** to open the **Address Picker**.

2 Bit Memory Address 2: The **Reset** instruction permits a consecutive range of bit memory addresses to be **Reset** simultaneously. Choose a **Bit Memory Address 2** to identify the end of the range of **Bit Memory Addresses** or leave this field blank if you only want just one output to respond to this instruction.

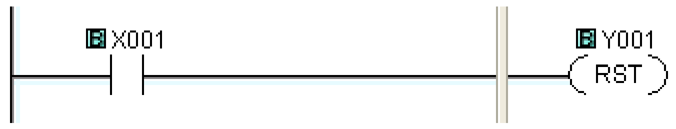
3 Immediate (Only for Y Bit): Choose the **Immediate Icon** if you want to turn **OFF** the **Output(s)** immediately. This feature is only available for actual **Y outputs**. This option is grayed out until a **Y** address is chosen. The **Immediate Icon** will appear adjacent to the **Reset Coil** in the **Ladder Editor**.

Note: Each **Set Coil** in your **Ladder Program** should be accompanied by a **Reset Coil** to turn **OFF** the addresses that were turned **ON** by the **Set Coil**.

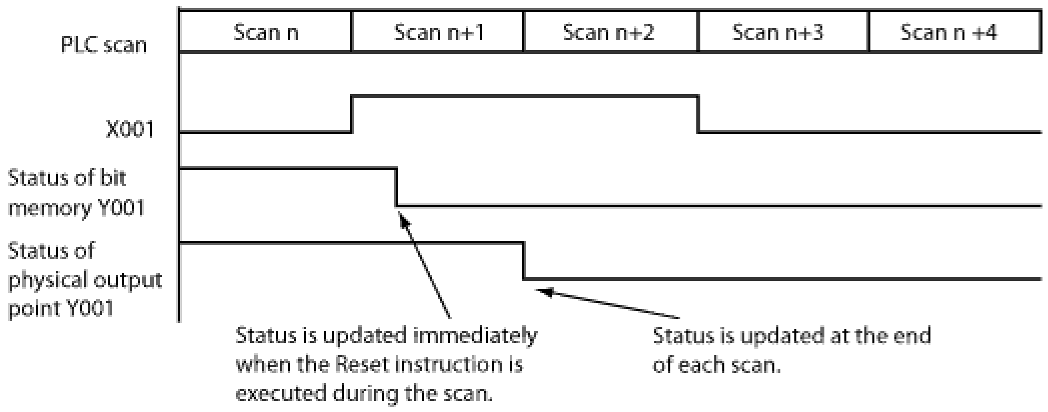
Example Program

Example Program 1: Default (No option)

In the following example, when input status bit **X001** is **ON**, the **Reset** instruction turns **OFF** the output status bit **Y001** immediately. **Y001** stays **OFF** even if **X001** is turned **OFF**.

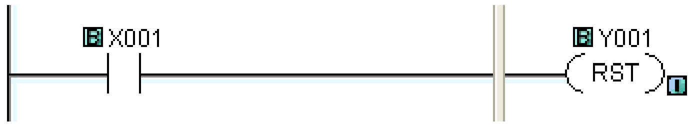


Timing Diagram

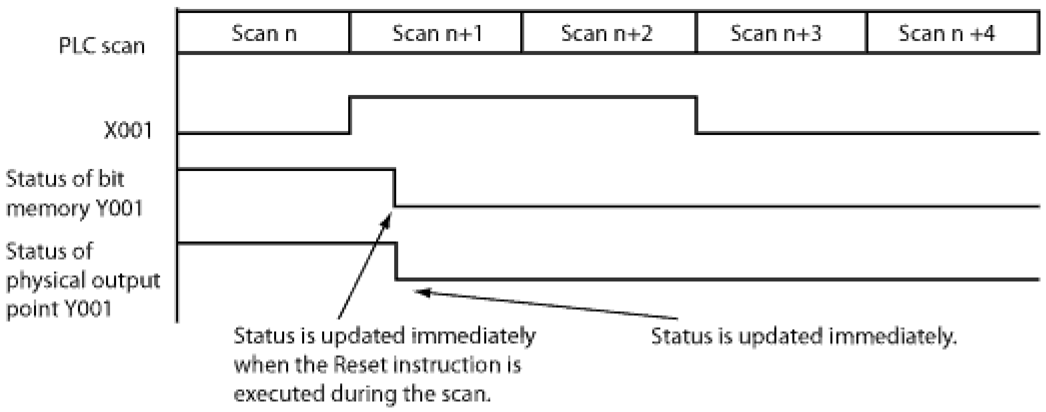

PLC scan
Scan n
Scan n+1
Scan n+2
Scan n+3
Scan n +4
X001
Status of bit memory Y001
Status of physical output point Y001
Status is updated immediately when the Reset instruction is executed during the scan.
Status is updated at the end of each scan.

Example Program 2: Immediate (Available only for the Y memory type)

In the following example, when input status bit **X001** is **ON**, the **Reset** instruction turns **OFF** both the output status bit **Y001** and the physical output point (**Y001** is the 1st output point on the **CPU** module) immediately. Please refer to the timing diagram to see the difference.



Timing Diagram



Related Topics:

Set Coil

Timer / Counter Instructions

Description

The **Timer Instruction** is used to provide an **Output** that responds according to a preset time value. The **Timer** starts timing when the when the rung is energized and can be configured to activate or deactivate the output when the preset time cycle is completed. The **Counter Instruction** is used to provide an **Output** that responds according to a preset count value. The **Counter** counts the **ON/OFF** transitions of the enabling rung to activate or deactivate the output when the preset count is completed.

To learn more about the **Timer** or **Counter Instructions**, move the pointer over the **Icon** to see the name and click on the **Icon** to open the specific topic for the instruction.

TMR CNT

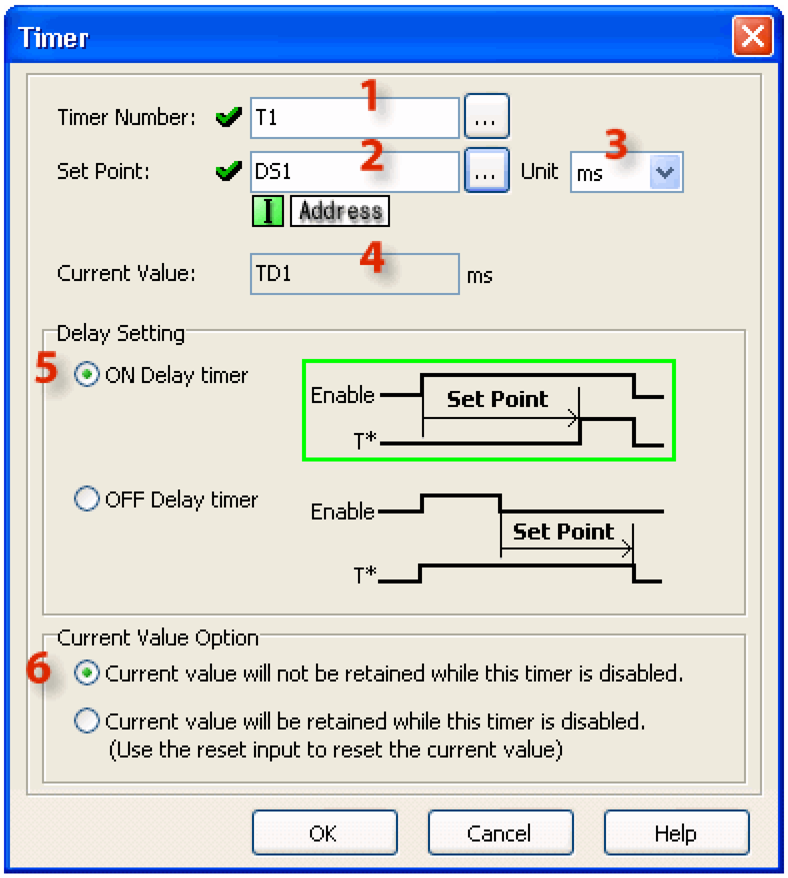
Timer

Definition

An **ON DelayTimer** measures the time duration that begins with a transition of the enable rung from **OFF** to **ON**. Beyond this transition point, the **Timer** increases the **Current Value**, when it reaches the **Set Point**, the **Timer Bit** is turned **ON**.

An **OFF DelayTimer** measures the time duration that begins with a transition of the enable rung from **ON** to **OFF**. Beyond this transition point, the **Timer** increases the **Current Value**, when it reaches the **Set Point**, the **Timer Bit** is turned **OFF**.

Warning: After the Off-Delay Counter has been finished, if the Setpoint value is then changed to a value which is GREATER than the Current time value, then the output of the timer will come on again until the new, higher Setpoint value is reached.


Timer

1 Timer Number: ✓ T1

2 Set Point: ✓ DS1 — 3 Unit: ms

I Address

4 Current Value: TD1 ms

Delay Setting

5 [✓ ON Delay timer] OFF Delay timer

Current Value Option

6 [✓ Current value will not be retained while this timer is disabled.]

Current value will be retained while this timer is disabled. (Use the reset input to reset the current value)

✔ Entry required or invalid entry ✔ Valid entry

Setup

1 Timer Number: Assign **Timer Number**. **Timer Numbers** are assigned from the Bit Memory Address range.

2 Set Point: Assign **Data Memory Address** to hold the **Set Point** value or a **Constant**. **DS** is the only eligible Data Memory Address range.

3 Unit: Assign the time measurement **Units** using the Drop Down List (**ms, sec, min, hour, day**).

4 Current Value: When the **Timer Number** is assigned (see **1** above), the **Current Value** is automatically assigned to the corresponding **Timer Data** register. **T1** corresponds to **TD1**, **T2** corresponds to **TD2**, etc.

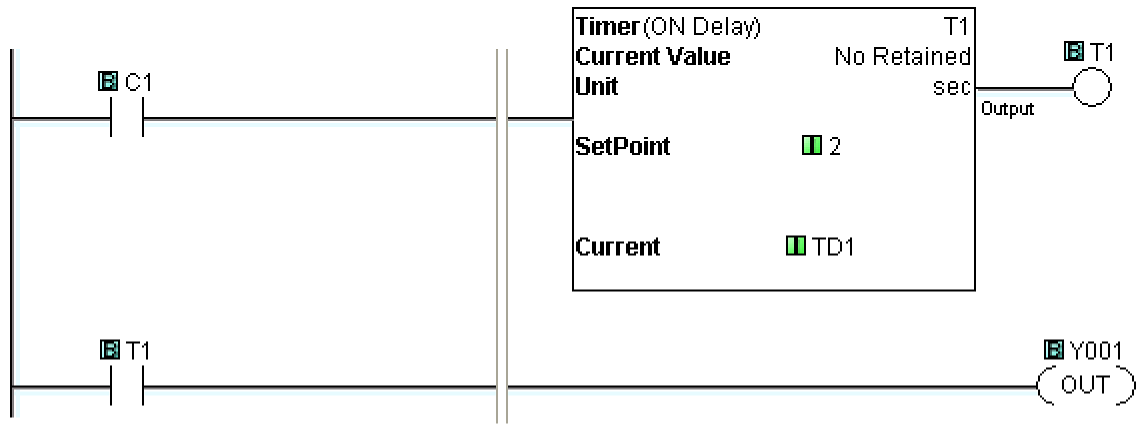
5 Delay Setting: The **Timer** can be set up as an **ON Delay Timer** or an **OFF Delay Timer**. Use the radio buttons to make the selection.

6 Current Value Option: The **Current Value** either resets automatically when the **Timer** is disabled or it requires a **Reset**. Make the selection using the radio buttons. If you select **Reset**, an additional rung connection will appear in the **Ladder Editor**. Use the **Line** tool to connect the **Reset** connection to the left rail and add enabling contacts as necessary. This **Option** is available only for the **ON Delay Timer** mode.

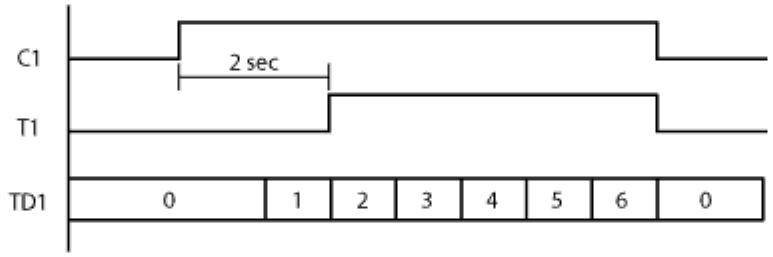
Example Programs

Example Program 1: ON Delay Timer

In the following example, timer status bit memory **T1** is turned **ON two seconds** after **C1** is turned **ON**. Then the second rung turns **Y001 ON**.

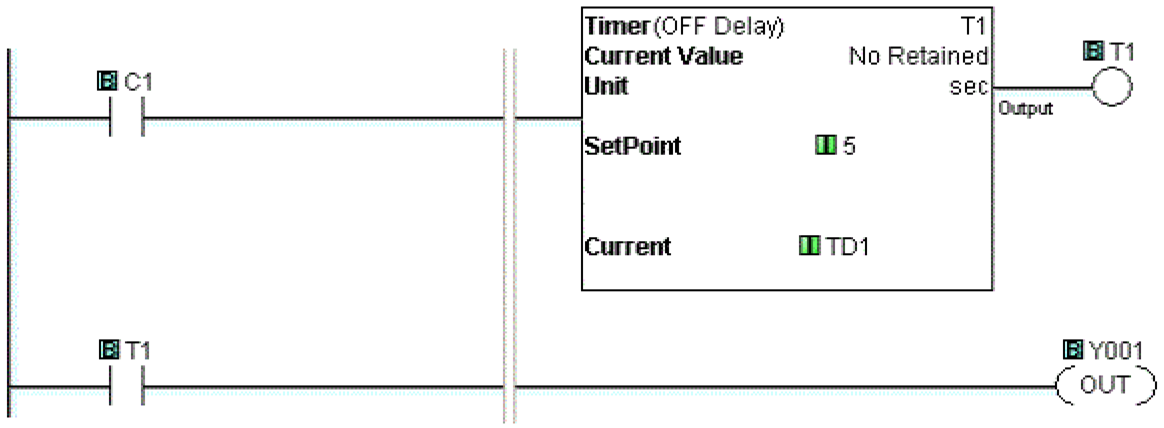


Timing Diagram

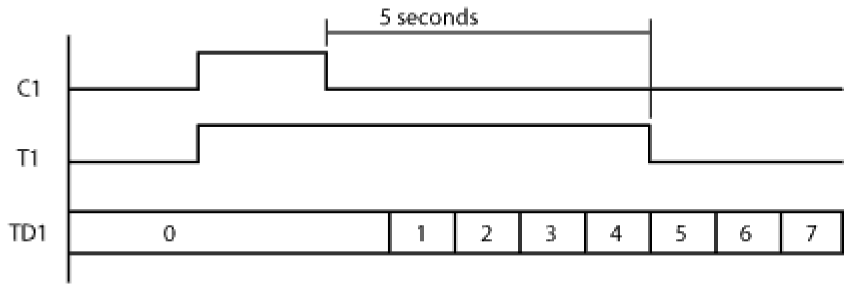


Example Program 2: OFF Delay Timer

In the following example, timer status bit memory **T1** is turned **ON** immediately after **C1** is turned **ON**. Then **T1** stays on **5 seconds** after **C1** is turned **OFF**. While **T1** is **ON**, the second rung keeps **Y001 ON**.

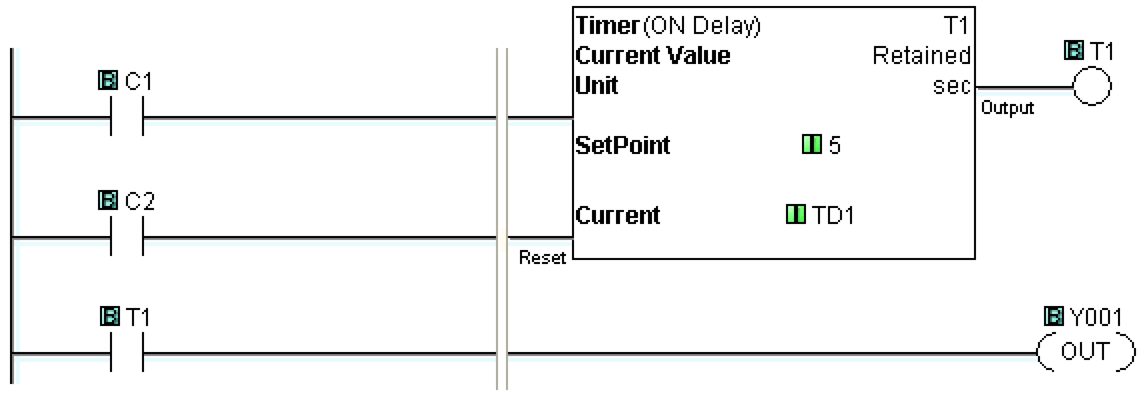


Timing Diagram

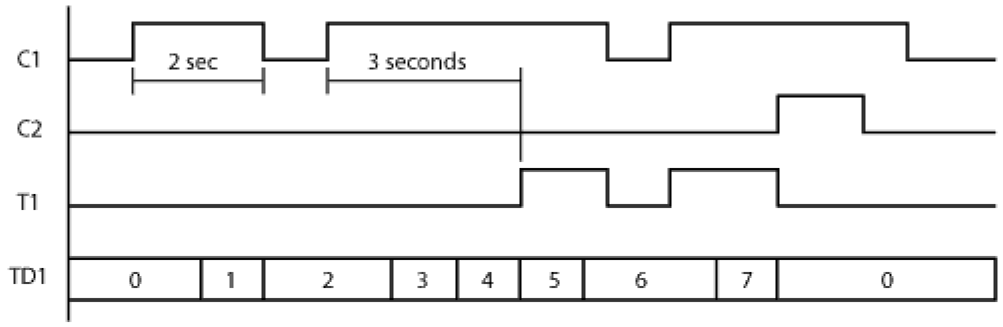


Example Program 3: Retained Current Value

In the following example, it is not necessary to keep **C1 ON** for **five seconds** continuously to turn timer status bit memory **T1 ON**. Instead, **C1** needs to be **ON** for **five seconds** in total.
While **T1** is **ON**, the second rung keeps **Y001 ON**. When **C2** is **ON**, the timer is **Reset**.



Timing Diagram



Related Topics:

Memory Addresses

Topic: CL166

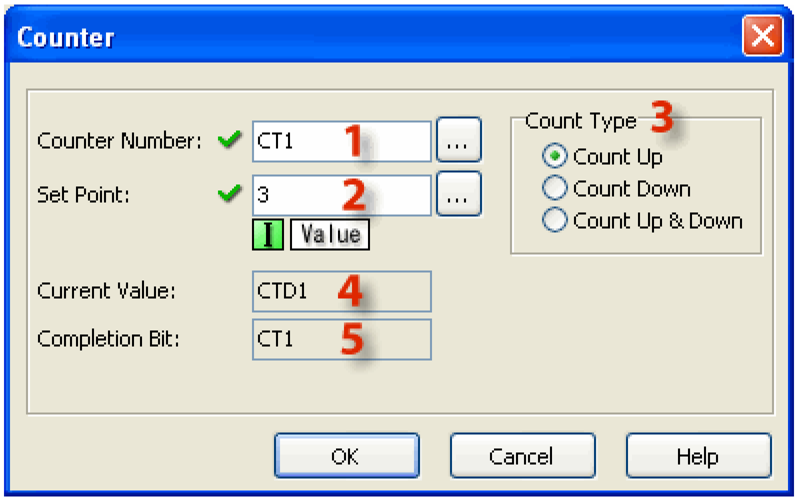
Counter

Definition

When enabled, a **Counter** instruction counts up or down (depending on user settings) until it reaches the **Set Point**. The **Counter** counts in response to the transition from **OFF** to **ON** of the enabling rung. If the user selects **Count Up** & **Down**, the **Counter** will have two enabling rungs, one for up counts and one for down counts.

The current count is held in the register shown in the **Current Value** field. When the **Current Value** reaches the **Set Point**, the **Completion Bit** is turned **ON**. The **Completion Bit** is turned **OFF** when the **Reset** rung is enabled.

Setup



Counter Number: ✓ CT1 1

Set Point: ✓ 3 2

Value

Count Type 3

[✓ Count Up] Count Down Count Up & Down

Current Value: CTD1 4

Completion Bit: CT1 5

✓ Entry required or invalid entry ✓ Valid entry

1 Counter Number: Assign a **Counter Number**. This number can only be the **CT Memory** type. Either type the **CT** number directly into the **Counter Number** field or click the **Browse Button** to open the **Address Picker**.

2 Set Point: Enter a **Constant** (Integer only). Or, assign a **Data Memory Address** to hold the **Set Point** value. **DS** and **DD** are the eligible Data Memory Address range.

3 Count Type: Use the radio buttons to select **Count Up**, **Count Down**, or **Count Up & Down**.

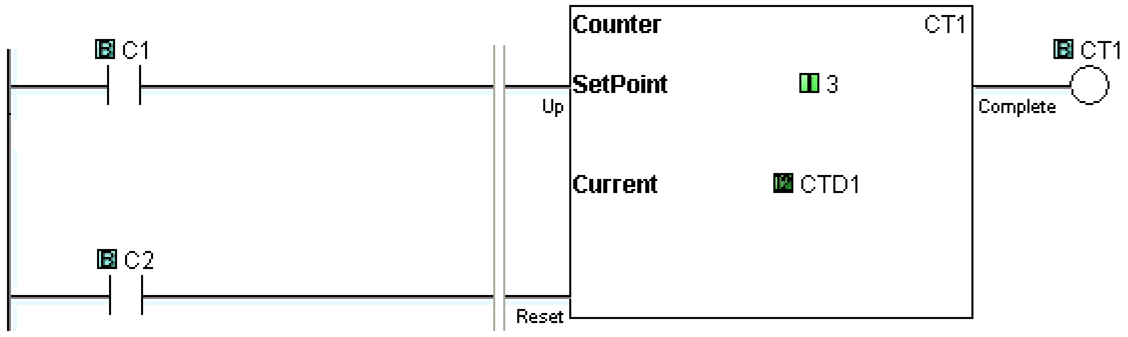
4 Current Value: When the **Counter Number** is assigned (see **1** above), the **Current Value** is automatically assigned to the corresponding **Counter Data** register. **CT1** corresponds to **CTD1**, **CT2** corresponds to **CTD2**, etc.

5 Completion Bit: The **Completion Bit** is the same as the **Counter Number**.

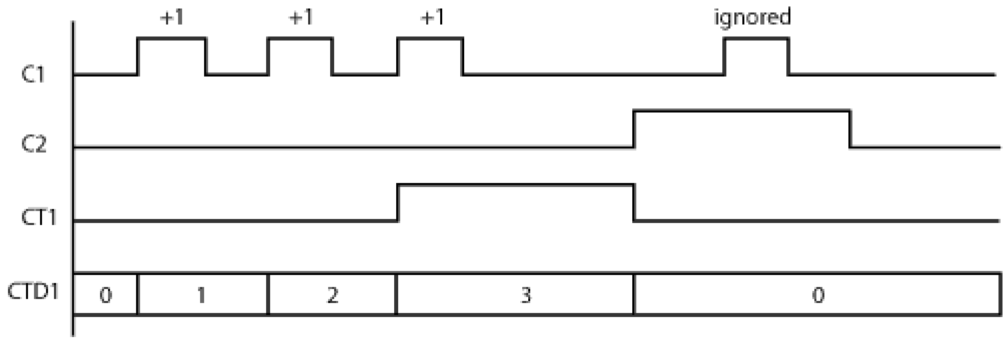
Example Program

Example Program 1: Up Counter

In the following example, after **C1** makes **OFF-to-ON** transition **3** times, the counter **CT1** counts up and the counter status bit **CT1** is turned **ON**. When **C2** is **ON**, the counter is **Reset** and does not accept the count up signal.

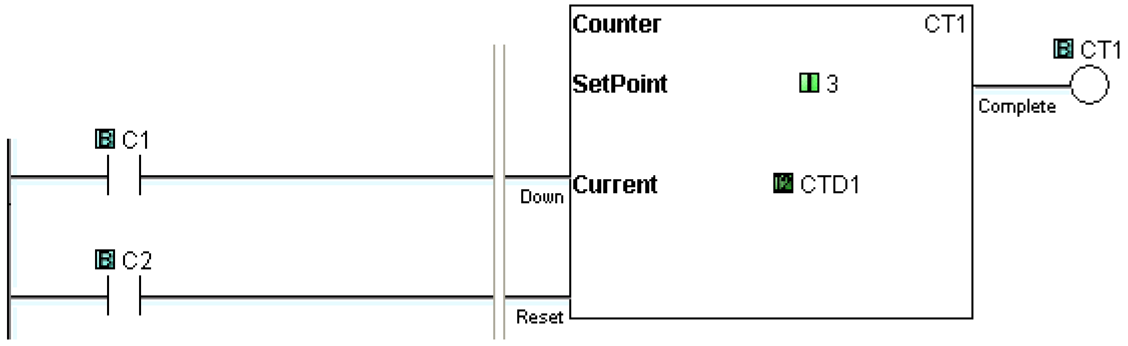


Timing Diagram

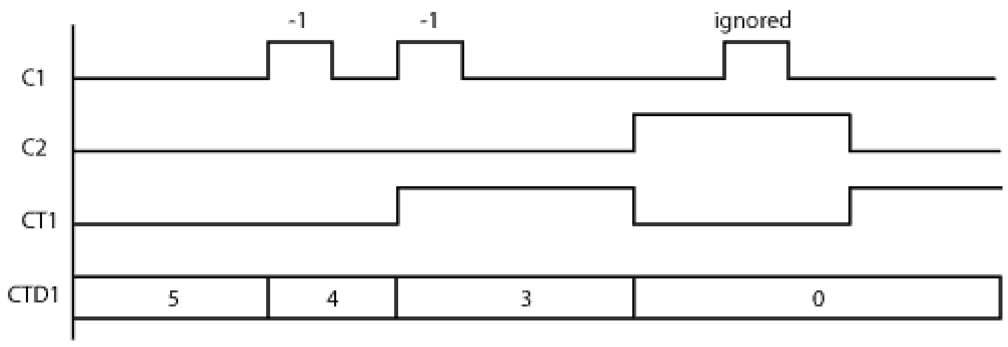


Example Program 2: Down Counter

In the following example, the current counter value **CTD1** is set to **5** in the beginning. After **C1** makes **OFF-to-ON** transition 2 times, the current counter value **CTD1** reaches the **Set Point** (3 in this example) and the counter status bit **CT1** is turned **ON**. When **C2** is **ON**, the counter is **reset**. (**CT1** is **OFF** and **CTD1** is set to **0**.) However, **CT1** is turned on immediately after **C1** is turned off because the current counter value **CTD1** is lower than the **Set Point**.

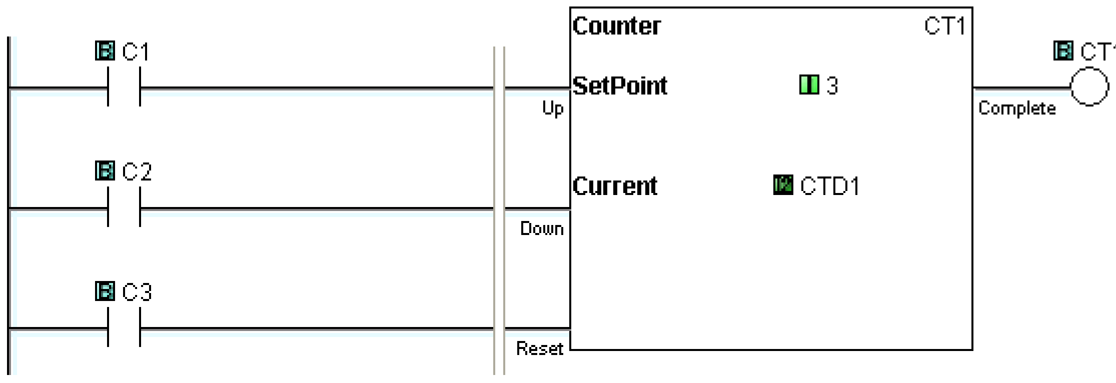


Timing Diagram

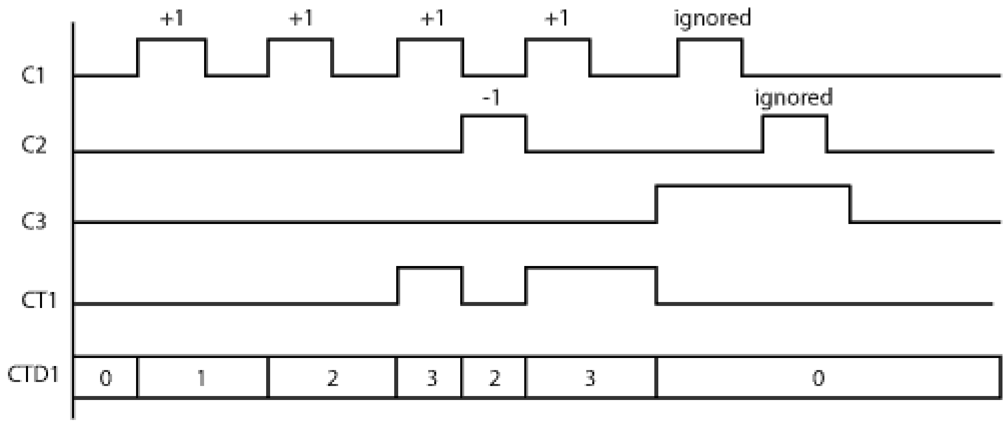


Example Program 3: Up/Down Counter

In the following example, when **C1** makes an **OFF-to-ON** transition, the current counter value **CTD1** increases by **one**. When **C2** makes an **OFF-to-ON** transition, the current counter value **CTD1** decreases by **one**. When the current counter value **CTD1** is the same or more than the **Set Point** (3 in this example), the counter status bit **CT1** is **ON**. When **C3** is **ON**, the counter is reset. (**CT1** is **OFF** and **CTD1** is set to **0**.)



Timing Diagram



<u>Related Topics:</u>

[Memory Addresses](#)

Advanced Instructions

Description

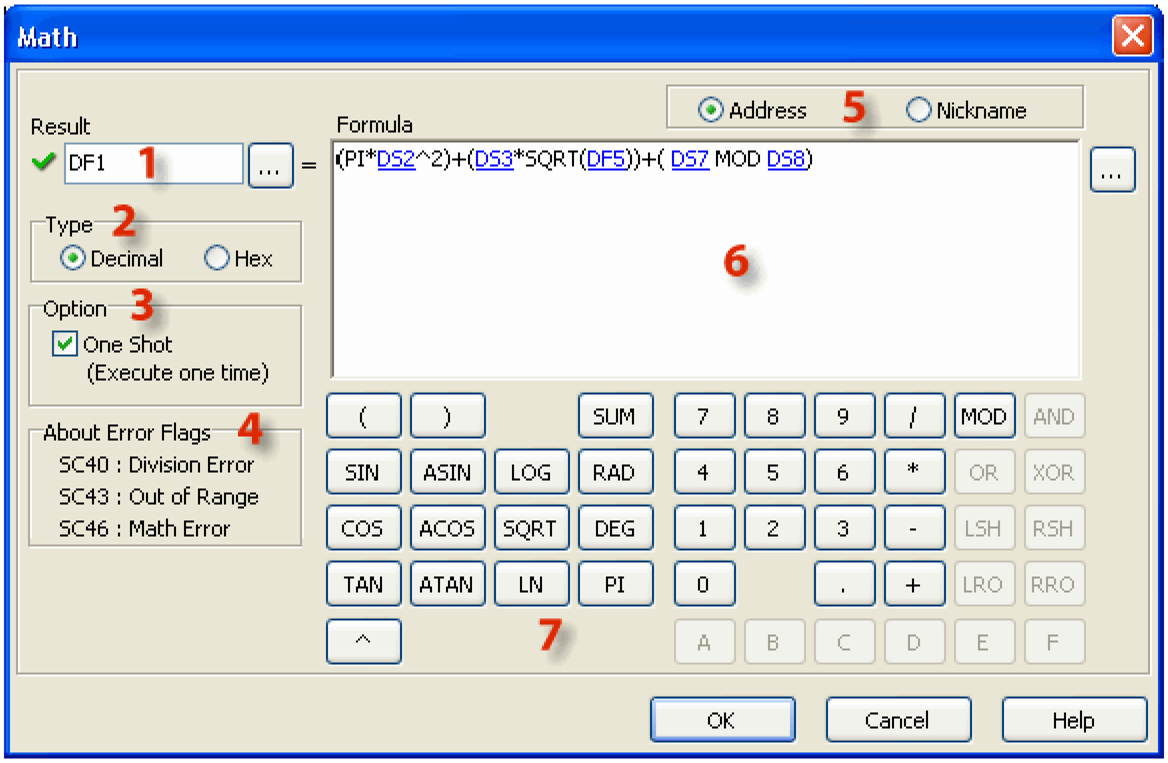
The **Advanced Instructions** include instructions that provide an **Output** based on mathematical calculations, sequences triggered by events or time, or register shifts triggered by a bit status depending how its been configured. The **CLICK Programming Software** offers three **Advanced Instruction** choices. To learn more about these instructions, move the pointer over the **Icon** to see the name and click on the **Icon** to open the specific topic for that instruction.

Math (Decimal)

Description

The **Math** instruction solves a user-defined formula during the execution of the **Ladder Program**. The formula is developed on the **Math** dialog using the on-screen keypad, the computer keyboard, and **Address Picker**. Two sets of mathematical operators are available. One set is appropriate for use with decimal values, and the other is for use with hexadecimal values. Also see [Math (Hex)](#). Parenthetical expressions can be nested up to eight levels deep. If the **Floating Point Data Type** is used in any operation, then all operations will be based on **Floating Point** math. The solution will be stored in the data format selected for the **Result**.

Decimal Setup



Result: ✓ DF1

Type: [✓ Decimal] Hex

Option: [✓ One Shot] (Execute one time)

About Error Flags
SC40 : Division Error
SC43 : Out of Range
SC46 : Math Error

[✓ Address] Nickname

Formula: (PI*DS2^2)+(DS3*SQRT(DF5))+(DS7 MOD DS8)

✔ **Entry required or invalid entry** ✔ **Valid entry**

1 Result: Assign a **Memory Address** where the **Result** will be stored. The **Result** value will be adjusted to the data type of the **Memory Address**. Click the **Browse Button** to open **Address Picker**.

2 Type: Selecting **Decimal** or **Hex** determines the mathematical operations that are available on the **Math** instruction dialog. Most of the operators are unique to either **Decimal** or **Hex** math.

Note: Changing this selection after beginning to develop the **Formula** will **erase** the **Formula**.

3 One Shot: Select **One Shot** to solve the formula only once after each **OFF-to-ON** transition of the enabling rung.

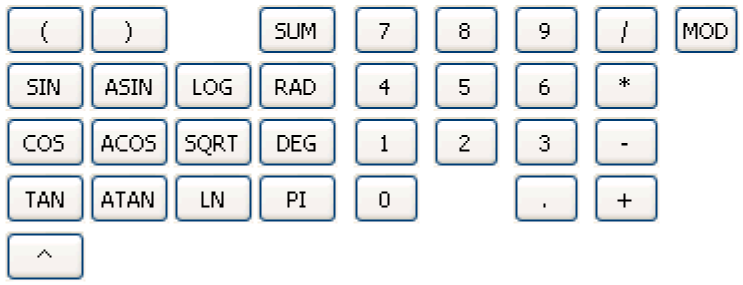
4 Error Flags: These **System Bits** turn **ON** when the specified condition has occurred.

5 Address or Nickname: Data Registers can be identified in the **Formula** by the **Memory Address** or the **Nickname**.

6Formula Pad: Create the mathematical expression using the onscreen keypad, the keyboard, and **Address Picker**. Click the **Browse Button** to open **Address Picker**.

7 Decimal Keypad: The **Decimal Keypad** includes numerical keys, **Sum** and **Modulo** operators, parentheses, and certain algebraic and transcendental operators. Trig functions can be solved in **Radians**.

Decimal Keypad



Key	Name	Definition	Usage Example
( )	Left and Right Parentheses	Used for grouping terms. Must be used in pairs.	(DF1 + 5)

SIN	Sine	Sine Θ = opposite / hypotenuse
		Θ = Radian
			SIN(DF1)
			SIN(RAD(DF1))

COS	Cosine	Cosine Θ = adjacent / hypotenuse
		Θ = Radian
			COS(DF1)
			COS((RAD(DF1))

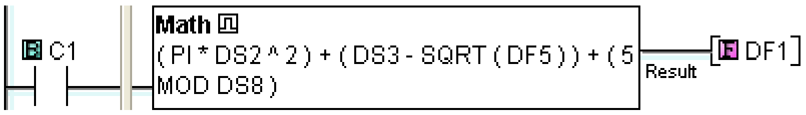
TAN	Tangent	Tangent Θ = sine / cosine Θ = Radian	TAN(DF1)
ASIN	Arc Sine	Inverse Sine	ASIN(DF1)
ACOS	Arc Cosine	Inverse Cosine	ACOS(DF1)
ATAN	Arc Tangent	Inverse Tangent	ATAN(DF1)
^	Power (Exponent)	*a* ^ *b* = *a* raised to the power *b*.	DF1^DF2
LOG	Log (Base 10)	If *x* = *b* ^ *y,* *y = log_bx*	LOG(DF1)

SQRT	Square Root	The Square Root of *b* is the number *a* if *a* * *a* = *b*.	SQRT(DF1)
LN	Log (Natural)	If *x* = *e* ^ *y*, *y* = *ln_bx*	LN(DF1)
SUM	Summation	Adds a group of values in a specified range of Memory Addresses.	SUM(DF1:DF10)
RAD	Radians	Converts degrees to radians. radians = degrees * (pi/180)	RAD(DF1)
DEG	Degrees	Converts radians to degrees. degrees = radians * (180/pi)	DEG(DF1)
PI	Pi	Constant that equals the ratio of a circle's circumference to its diameter.	3.1415927
7 8 9 4 5 6 1 2 3 0	Number Keys	The numbers 0 to 9.	0123456789
.	Decimal	Decimal notation, "."	1.234
/ * - +	Arithmetic Operators	Divide Multiply Subtract Add	DF1 / DF2 DF1 * DF2 DF1 - DF2 DF1 + DF2
MOD	Modulo	*a* MOD *b* refers to the arithmetic remainder after *a* is divided by *b*.	DF1 MOD DF2

Example Decimal Program

Example Program: Integer and Floating Point Math

In the following example, when **C1** transitions from **OFF-to-ON**, the **Formula** is solved and the **Result** is stored in **DF1** as a **Floating Point** number. The **Result** is converted to the **Data Type** of the assigned **Memory Address**.



```
(PI * DS2 ^ 2) + (DS3 - SQRT (DF5)) + (5 MOD DS8)
```

Related Topics:

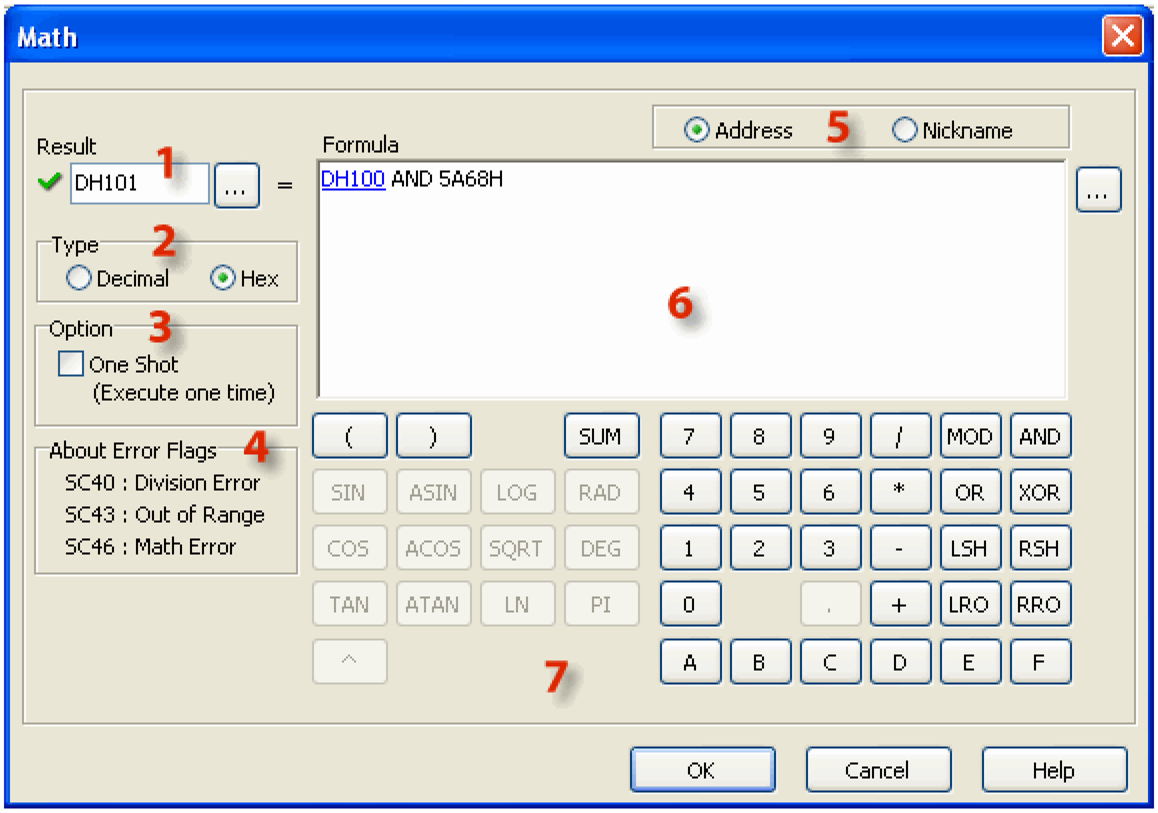
[Hex Math Instruction]
[Data Types]
[Memory Addresses]

Topic: CL203

Math (Hex)

Description

The **Math** instruction solves a user-defined formula during the execution of the **Ladder Program**. The formula is developed on the **Math** dialog using the onscreen keypad, the computer keyboard, and **Address Picker**. Two sets of mathematical operators are available. One set is appropriate for use with decimal values, and the other is for use with hexadecimal values. See also Math (Decimal). Parenthetical expressions can be nested up to eight levels deep. If the **Floating Point Data Type** is used in any operation, then all operations will be based on **Floating Point** math. The solution will be stored in the data format selected for the **Result**.


Math
1
2
3
4
5
6
7
( ) SUM 7 8 9 / MOD AND
SIN ASIN LOG RAD 4 5 6 * OR XOR
COS ACOS SQRT DEG 1 2 3 - LSH RSH
TAN ATAN LN PI 0 . + LRO RRO
^ A B C D E F
OK
Cancel
Help

Result ✓ DH101 =

Type: Decimal [✓ Hex]

Option: One Shot (Execute one time) ☐

About Error Flags
SC40 : Division Error
SC43 : Out of Range
SC46 : Math Error

[✓ Address] Nickname

Formula: DH100 AND 5A68H

✓ **Entry required or invalid entry** ✓ **Valid entry**

1 Result: Assign a **Memory Address** where the **Result** will be stored. The **Result** will be stored in the hex data format. Click the **Browse Button** to open **Address Picker**.

2 Type: Selecting **Decimal** or **Hex** determines the mathematical operations that are available on the **Math** instruction dialog. Most of the operators are unique to either **Decimal** or **Hex** math.

Note: Changing this selection after beginning to develop the formula will erase the formula.

Hex Setup

3 One Shot: Select **One Shot** to solve the formula only once after each **OFF-to-ON** transition of the enabling rung.

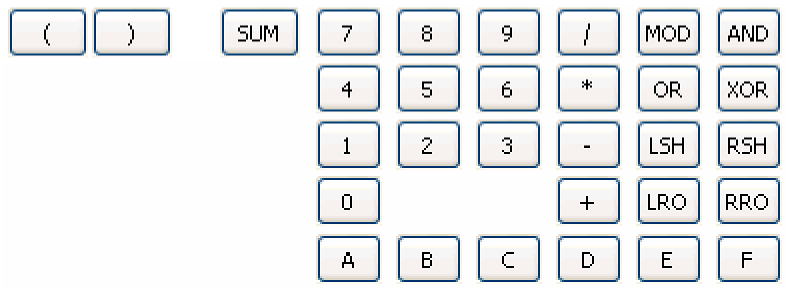
4 Error Flags: These **System Bits** turn **ON** when the specified condition has occurred.

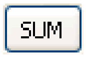
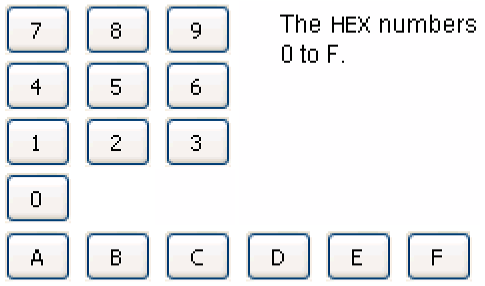
5 Address or Nickname:Data Registers can be identified in the **Formula** by the **Memory Address** or the **Nickname**.

6 Formula Pad: Create the mathematical expression using the onscreen keypad, the keyboard, and **Address Picker**. Click the **Browse Button** to open **Address Picker**.

7 Hex Keypad: The **Hex Keypad** includes the numerical and letter keys to create **Hex** numbers, **Sum** and **Modulo**, parentheses, and certain logical operators and bit operators. See Hex Keypad Details below.

Hex Keypad Details



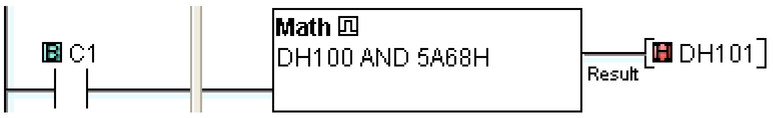
Key	Name	Definition	Usage Example
	Left and Right Parentheses	Used for grouping terms. Must be used in pairs.	(DH1 + 5h)
	Summation	Adds a group of values in a specified range of Memory Addresses.	SUM(DH1:DH10)
 The HEX numbers 0 to F.		Number Keys	1h 1234h AB59h

/ * - +	Arithmetic Operators	Divide Multiply Subtract Add	DH1 / DH2 DH1 * DH2 DH1 - DH2 DH1 + DH2
MOD	Modulo	a MOD b refers to the arithmetic remainder after a is divided by b.	DH1 MOD DH2
OR	OR	Logical OR	DH1 OR DH2
LSH	LSH	Shift Left	LSH(DH1,1h)
LRO	LRO	Rotate Left	LRO(DH1,1h)
AND	AND	Logical AND	DH1 AND DH2
XOR	XOR	Logical XOR	DH1 XOR DH2
RSH	RSH	Shift Right	RSH(DH1,1h)
RRO	RRO	Rotate Right	RRO(DH1,2h)

Example Hex Program

Example Program: Hex Math

In the following example, when **C1** transitions from **OFF-to-ON**, the **Formula** is solved and the **Result** is stored in **DH101** as a **Hex** number.



Formula
DH101 = DH100 AND 5A68H

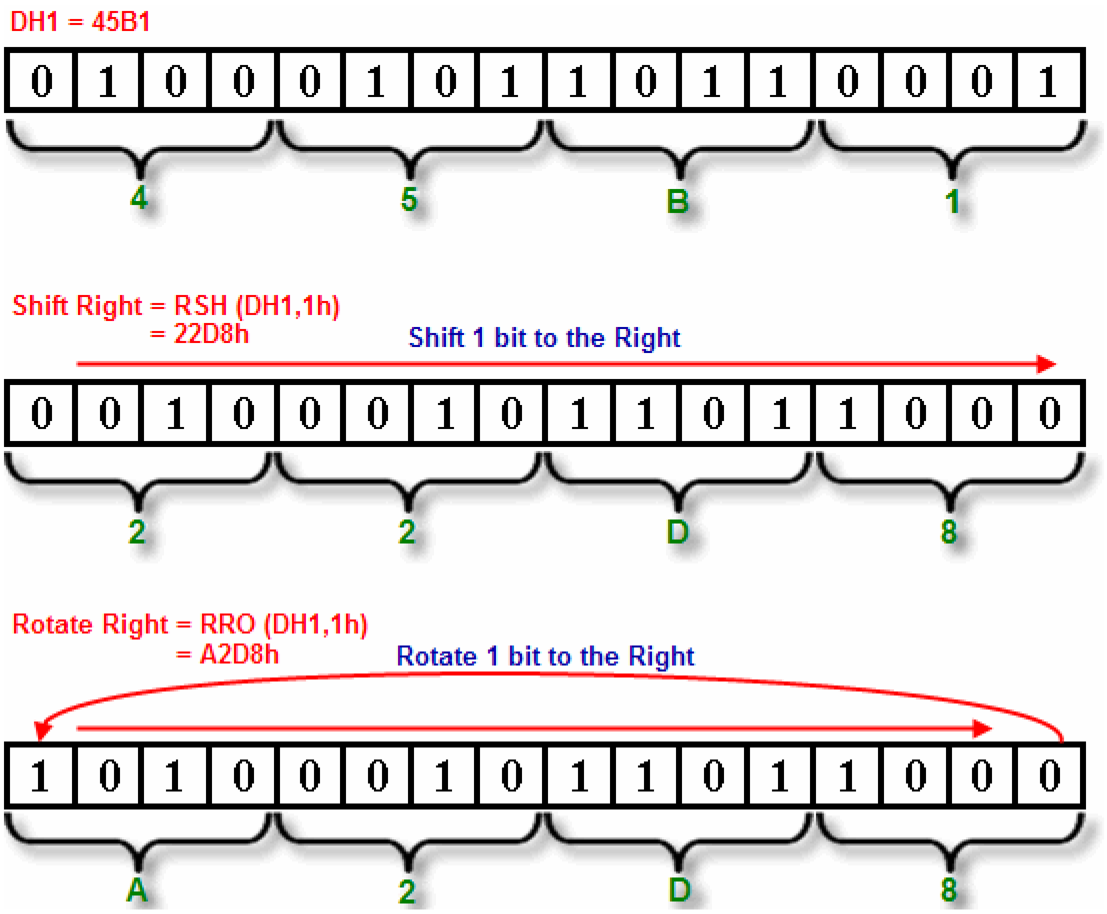
If **DH100** = **1AEAh** and **C1** = **ON**, then **DH101** = **1A68h**

Shift and Rotate

DH1 = 45B1

1. Shift Right: DH2 = RSH (DH1,1h)
   → DH2 = 22D8h

2. Rotate Right: DH2 = RRO (DH1,1h)
   → DH2 = A2D8h



<u>**Related Topics:**</u>

[Decimal Math Instruction]

[Data Types]

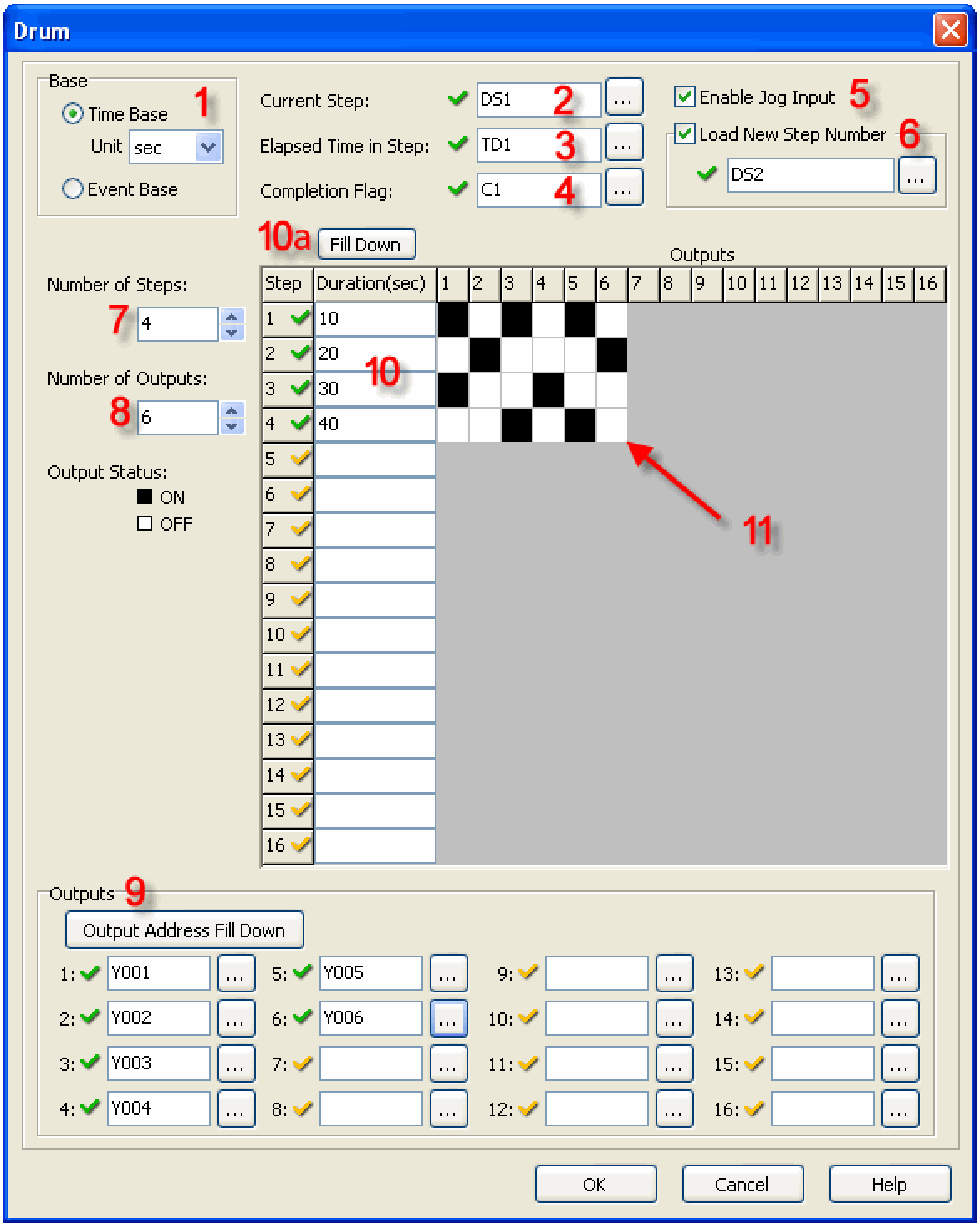
[Memory Addresses]

Drum Instruction: Time Base

Definition

The **Drum** instruction simulates an electromechanical drum sequencer, using either a **Time Based** or an **Event Base** sequencing strategy. Each **Drum** instruction is capable of sequencing through **1 to 16 steps** and turning **ON** as many as **16 outputs** in a user defined pattern. Outputs can be either physical outputs or internal control relays. A **Flag Bit** is turned **ON** to indicate the completion of the sequence.

Setup



Base: [✓ Time Base] Event Base
Unit: sec

Current Step: DS1
Elapsed Time in Step: TD1
Completion Flag: C1

[✓] Enable Jog Input
[✓] Load New Step Number: DS2

Number of Steps: 4
Number of Outputs: 6

Output Status: ■ ON ☐ OFF

Step	Duration(sec)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	10	■	☐	■	☐	■	☐										
2	20	☐	■	☐	☐	☐	■										
3	30	■	☐	☐	■	☐	☐										
4	40	☐	☐	■	☐	■	☐										
5																	
6																	
7																	
8																	
9																	
10																	
11																	
12																	
13																	
14																	
15																	
16																	

Outputs

1:	Y001	5:	Y005	9:		13:	
2:	Y002	6:	Y006	10:		14:	
3:	Y003	7:		11:		15:	
4:	Y004	8:		12:		16:	

✓ **Entry required or invalid entry** ✓ **Optional** ✓ **Valid entry**

1 Base: The **Drum** instruction operates either on a **Time Base** or an **Event Base**. If **Time Base** is selected, the increment of time is also chosen from the drop down list. The available time increments are milliseconds, seconds, minutes, hours, or days.

2 Current Step: Identify the **Memory Address** where the **Current Step** number will be stored. This must be a **DS** type **Memory Address**.

3 Elapsed time in step: Identify the **Memory Address** where the **Elapsed Time in Step** will be stored. This must be a **TD Type Memory Address**.

4 Completion flag: Assign a **Control Relay** (C bit) to be the **Completion Flag**.

5 Enable Jog Input: Selecting **Enable Jog Input** turns on an additional rung that allows the user to jump to the next **Step** each time the **Jog** input transitions from **OFF-to-ON**.

6 Load New Step Number: Selecting **Load New Step** turns on an additional rung "**Step No Input**" that allows the user to jump to a specified **Step** in the sequence. Assign a **Memory Address** to hold the desired new**Step Number**. This must be a **DS** type **Memory Address**.The **Step No Input** is edge triggered. When the **Step No Input** transitions from **OFF-to-ON**, the value of the new **Step Number** is copied to the **Current Step**.

7 Number of Steps: Select a number from **1** to **16**. This will determine the number of steps in the sequence.

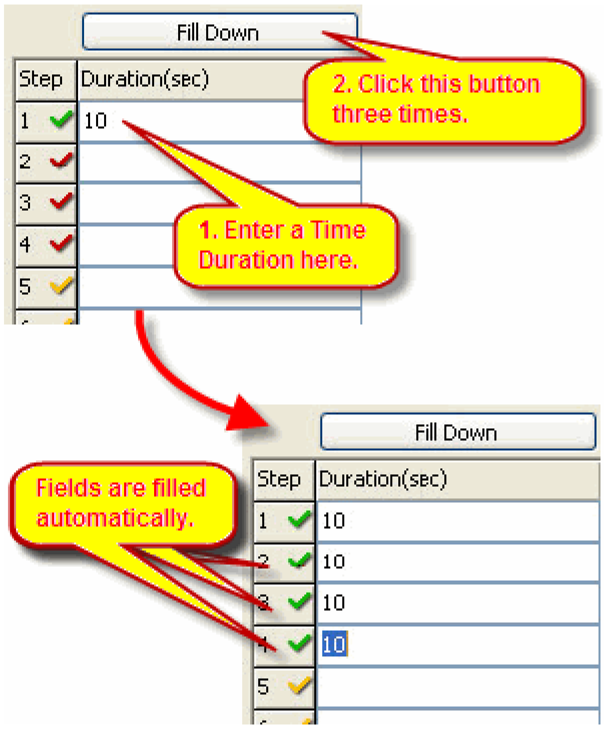
8 Number of Outputs: Select a number from **1** to **16**. This will determine the number of outputs available during the sequence.

9 Outputs: Assign **Outputs**. **Outputs** can be physical **Outputs** and/or internal control relays. The number of **Outputs** should match the number chosen at item **8**, above.
You can use the **Output Address Fill Down** button Output Address Fill Down to enter consecutive **Addresses** automatically. Enter the first **Memory Address** and keep clicking the **Output Address Fill Down** button.

10 Duration: Assign a time **Duration** for each **Step**. The number of **Steps** should match the number chosen at item **7**, above.

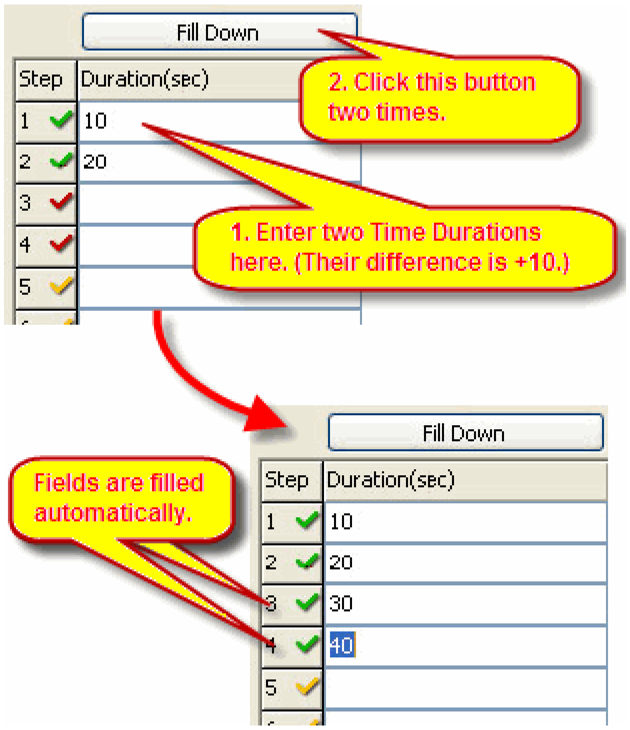
10a Fill Down: The **Fill Down** button Fill Down may help you enter the time **Durations** quickly.

<u>Example 1: Fill with the same Time Duration</u>


Fill Down
Step
Duration(sec)
2. Click this button three times.
1. Enter a Time Duration here.
Fields are filled automatically.

Step	Duration(sec)
1 ✓	10
2 ✓	10
3 ✓	10
4 ✓	10
5 ✓	

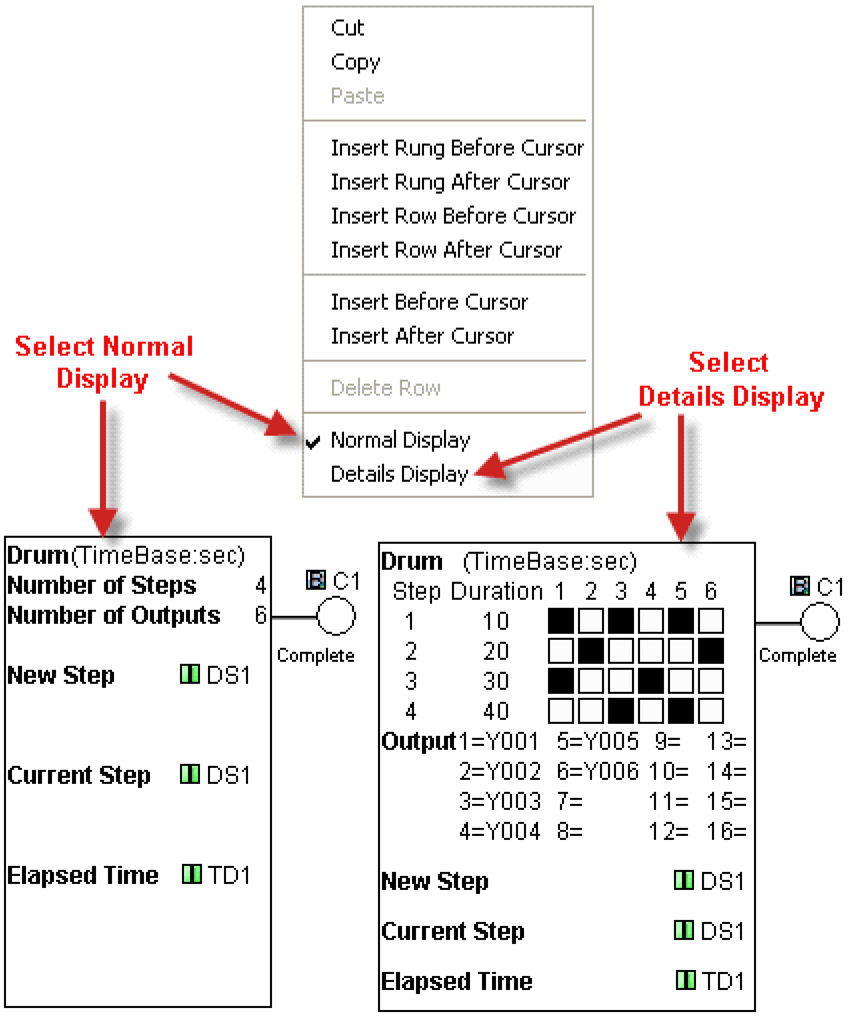
Example 2: Fill with Increasing Time Durations


Fill Down
Step
Duration(sec)
2. Click this button two times.
1. Enter two Time Durations here. (Their difference is +10.)
Fields are filled automatically.

Step	Duration(sec)
1 ✓	10
2 ✓	20
3 ✓	30
4 ✓	40
5 ✓	

11 Output Pattern: Click in the grid to develop the sequence the **Drum** instruction will follow. Each cell corresponds to a combination of one **Step** number and one **Output** number.

Normal or Details Display



Drum(TimeBase:sec)

Number of Steps	4
Number of Outputs	6
New Step	DS1
Current Step	DS1
Elapsed Time	TD1

Drum (TimeBase:sec)

Step	Duration	1	2	3	4	5	6
1	10	■	□	■	□	■	□
2	20	□	■	□	□	□	■
3	30	■	□	□	■	□	□
4	40	□	□	■	□	■	□

```
Output1=Y001  5=Y005  9=   13=
      2=Y002  6=Y006 10=   14=
      3=Y003  7=     11=   15=
      4=Y004  8=     12=   16=
```

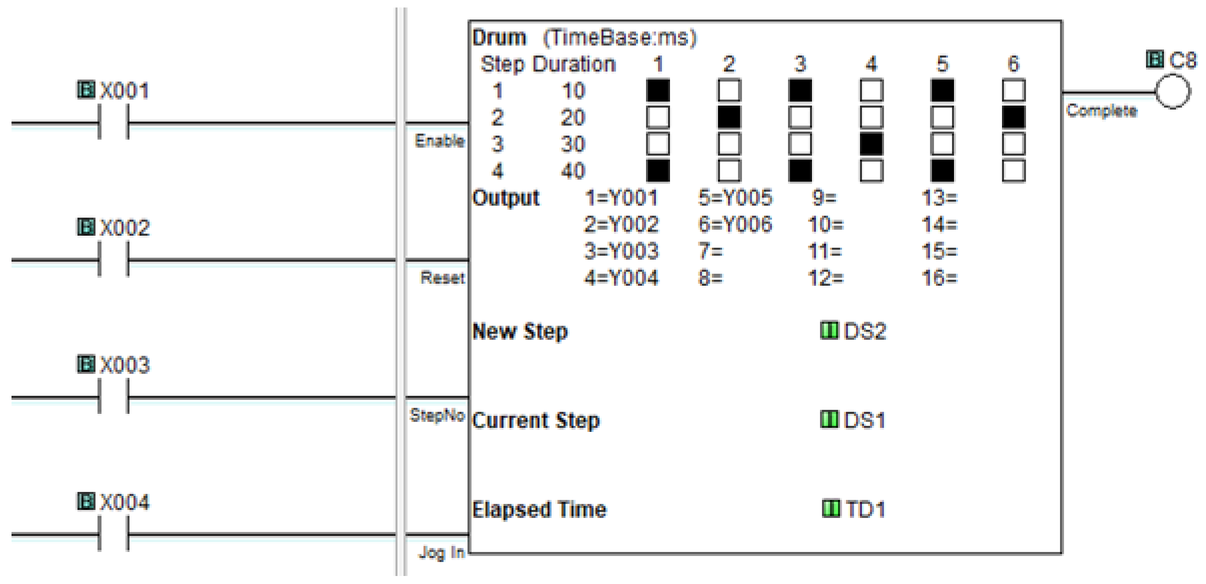
New Step	DS1
Current Step	DS1
Elapsed Time	TD1

Note: Right click the rung symbol to open the pop-up menu and toggle between **Normal Display** and **Details Display**.

Example Program

Example Program: Time Base

In the following example, when **X001** is **ON**, the **Drum** instruction begins its sequence at **Step 1**. If **X001** remains **ON**, the sequence will continue until complete, unless interrupted by **X002**, **X003** or **X004**. The duration of **Step 1** was assigned to be **10** seconds on the **Setup Dialog**. **Step 2** is **20** seconds in duration, **Step 3** is **30** seconds, and **Step 4** is **40** seconds. At the end of **Step 4**, the sequence is complete, and the **C1** bits turns **ON**. **X002** must turn **ON** to **Reset** the **Drum** instruction. If **X003** turns **ON**, the **Drum** instruction jumps to the step number loaded in **DS2**. If **X004** turns **ON**, the **Drum** instruction jumps to the next step in the sequence.

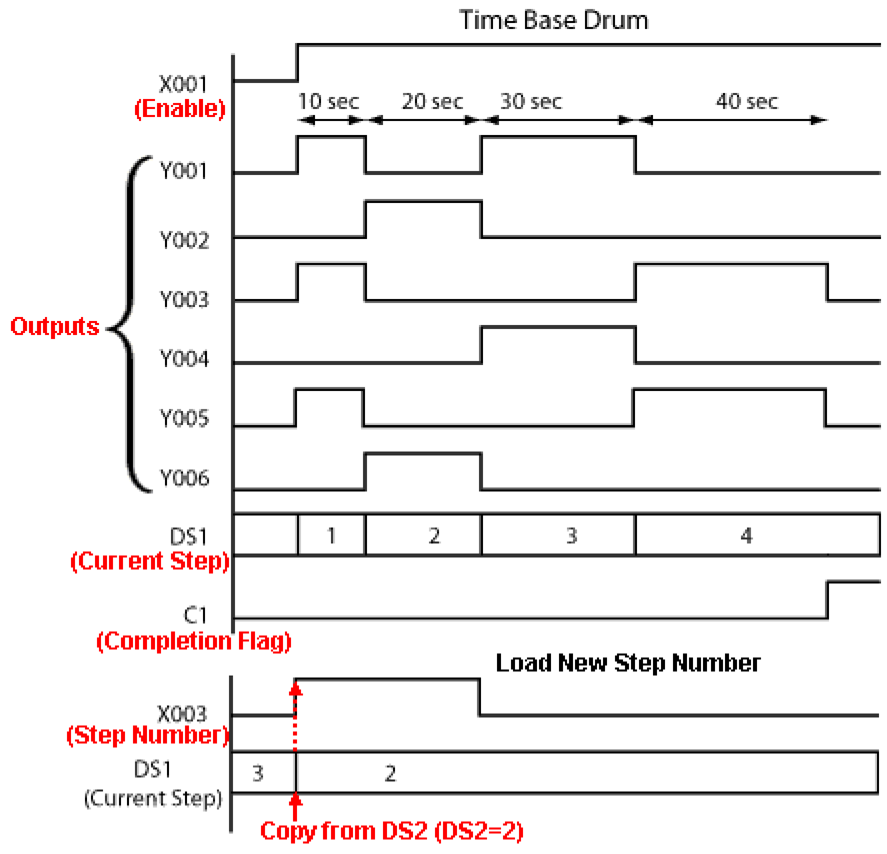


Drum (TimeBase:ms)

Step	Duration	1	2	3	4	5	6
1	10	■	□	■	□	■	□
2	20	□	■	□	□	□	■
3	30	□	□	□	■	□	□
4	40	■	□	■	□	■	□

Output				
	1=Y001	5=Y005	9=	13=
	2=Y002	6=Y006	10=	14=
	3=Y003	7=	11=	15=
	4=Y004	8=	12=	16=

New Step	DS2
Current Step	DS1
Elapsed Time	TD1



Related Topics:

Drum Instruction: Event Base

Memory Addresses

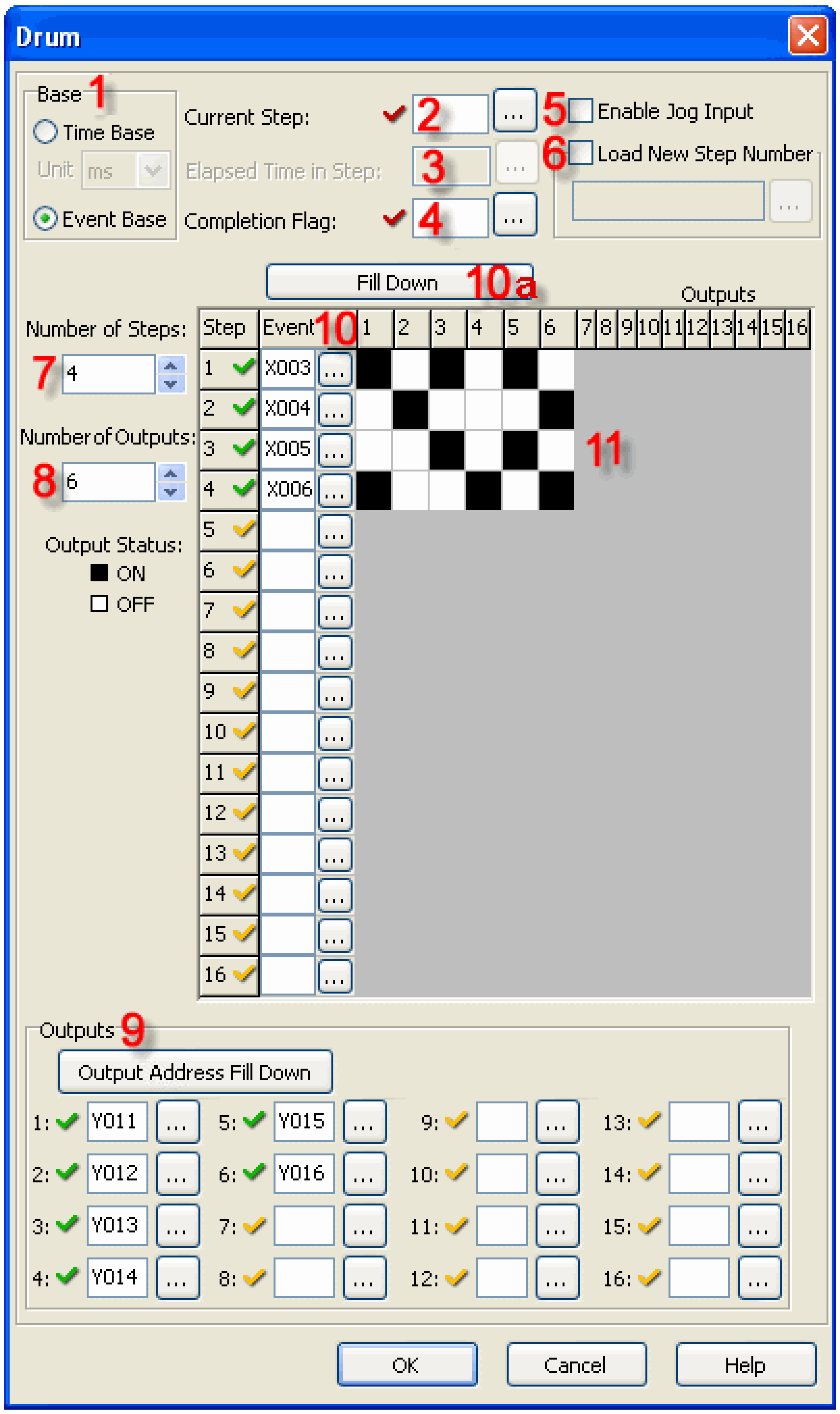
Data Types

Drum Instruction: Event Base

Definition

The **Drum** instruction simulates an electromechanical drum sequencer, using either a **Time Base** or an **Event Base** sequencing strategy. Each **Drum** instruction is capable of sequencing through **1 to 16 steps** and turning **ON** as many as **16 outputs** in a user defined pattern. Outputs can be either physical outputs or internal control relays. A **Flag Bit** is turned **ON** to indicate the completion of the sequence.

Event Base Setup



Base: ( ) Time Base, Unit ms, [✓ Event Base]

Current Step: ✓

Elapsed Time in Step:

Completion Flag: ✓

☐ Enable Jog Input

☐ Load New Step Number

Number of Steps: 4

Number of Outputs: 6

Output Status: ■ ON ☐ OFF

Step	Event	1	2	3	4	5	6
1 ✓	X003	■	☐	■	☐	■	☐
2 ✓	X004	☐	■	☐	☐	☐	■
3 ✓	X005	☐	☐	■	☐	■	☐
4 ✓	X006	■	☐	☐	■	☐	■
5 ✓							
6 ✓							
7 ✓							
8 ✓							
9 ✓							
10 ✓							
11 ✓							
12 ✓							
13 ✓							
14 ✓							
15 ✓							
16 ✓							

Outputs

1: ✓ Y011	5: ✓ Y015	9: ✓	13: ✓
2: ✓ Y012	6: ✓ Y016	10: ✓	14: ✓
3: ✓ Y013	7: ✓	11: ✓	15: ✓
4: ✓ Y014	8: ✓	12: ✓	16: ✓

✓ **Entry required or invalid entry** ✓ **Optional** ✓ **Valid entry**

1 Base: The **Drum** instruction operates either on a **Time Base** or an **Event Base**. If **Event Base** is selected, the time increment drop down list is grayed out.

2 Current Step: Identify the **Memory Address** where the **Current Step** number will be stored.

3 Elapsed time in step: (This parameter is not used in the **Event Base** mode. Therefore, it has been grayed out since **CLICK Programming Software Version 1.11**.)

4 Completion flag: Assign a **Control Relay** (**C bit**) to be the **Completion Flag**.

5 Enable Jog Input: Selecting **Enable Jog Input** turns on an additional rung that allows the user to jump to the next **Step** each time the **Jog** rung transitions from **OFF-to-ON**.

6 Load new step number: Selecting **Load New Step** turns on an additional rung that allows the user to jump to a specified **Step** in the sequence. Assign a **Memory Address** to hold the desired **Step Number**.

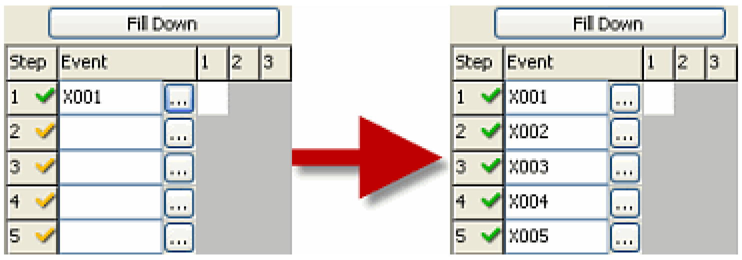
7 Number of Steps: Select a number from 1 to 16. This will determine the number of steps in the sequence.

8 Number of Outputs: Select a number from 1 to 16. This will determine the number of outputs available during the sequence.

9 Outputs: Assign **Outputs**. **Outputs** can be physical **Outputs** and/or internal control relays. The number of **Outputs** should match the number chosen at item **8**, above. You can use the **Output Address Fill Down** button Output Address Fill Down to enter consecutive **Addresses** automatically. Enter the first **Memory Address** and keep clicking the **Output Address Fill Down** button.

10 Event: Assign the bit-level **Memory Address** that will initiate each **Step**. The number of **Steps** will match the number chosen at **7**, above.

10a Fill Down: The **Fill Down** button Fill Down will enter consecutive **Addresses** automatically. Enter the first **Address** and keep clicking the **Fill Down** button.

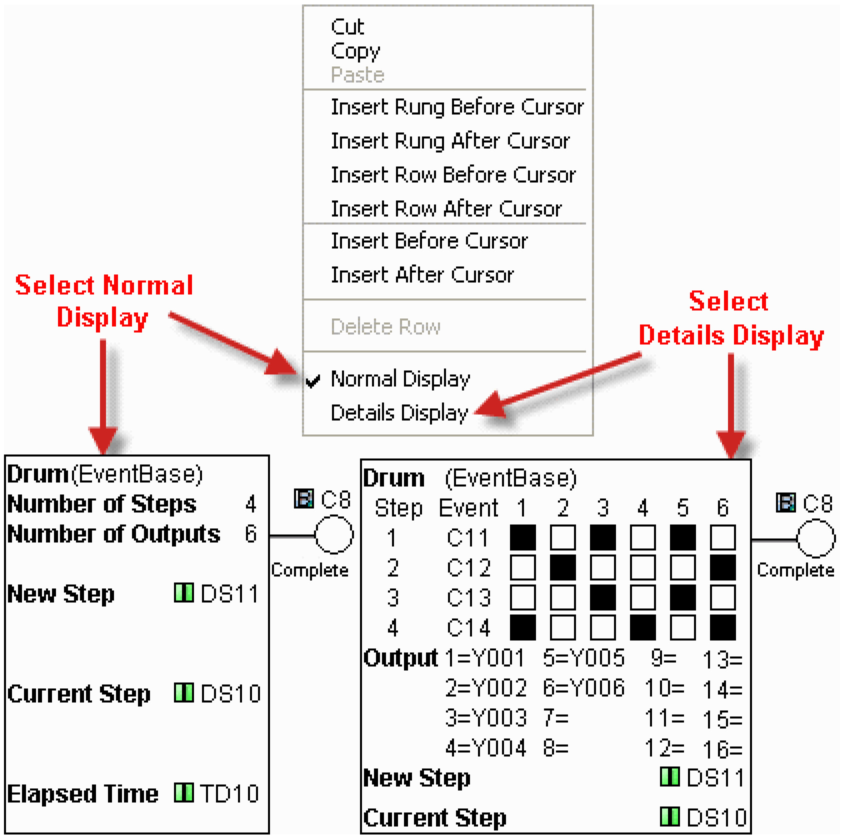


Step	Event	1	2	3
1 ✓	X001			
2 ✓				
3 ✓				
4 ✓				
5 ✓				

Step	Event	1	2	3
1 ✓	X001			
2 ✓	X002			
3 ✓	X003			
4 ✓	X004			
5 ✓	X005			

11 Output Pattern: Click in the grid to develop the sequence the **Drum** instruction will follow. Each cell corresponds to a combination of one **Step** number and one **Output** number.

Normal or Detail Display



Drum(EventBase)
Number of Steps 4
Number of Outputs 6
New Step DS11
Current Step DS10
Elapsed Time TD10

Drum (EventBase)

Step	Event	1	2	3	4	5	6
1	C11	■	□	■	□	■	□
2	C12	□	■	□	□	□	■
3	C13	□	□	■	□	■	□
4	C14	■	□	□	■	□	■

Output				
	1=Y001	5=Y005	9=	13=
	2=Y002	6=Y006	10=	14=
	3=Y003	7=	11=	15=
	4=Y004	8=	12=	16=

New Step DS11
Current Step DS10

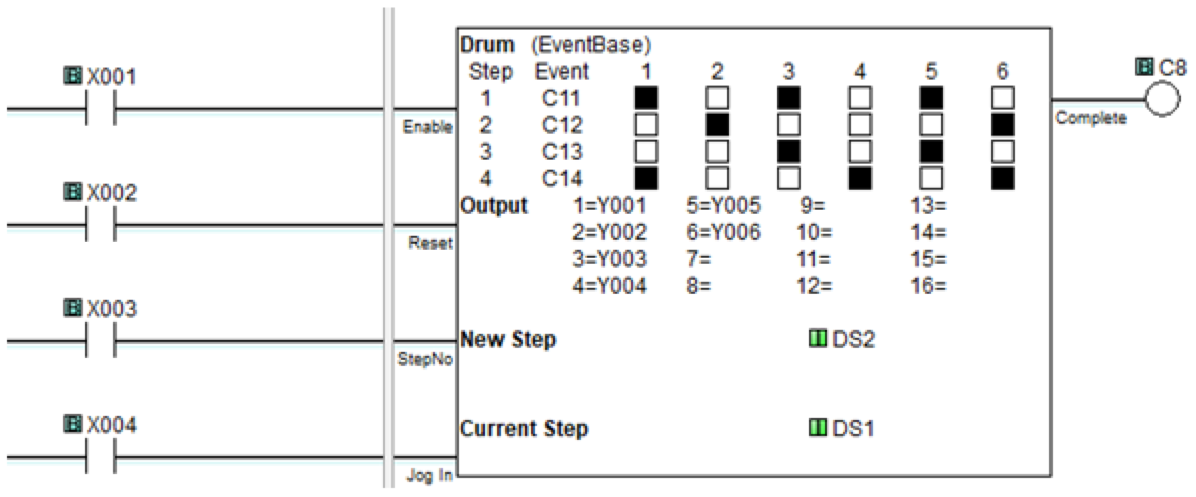
Note: Right click the rung symbol to open the pop-up menu and toggle between **Normal Display** and **Details Display**.

Example Program

Example Program: Event Base

In the following example:

- When **X001** is **ON**, the Drum instruction is enabled and **Step 1** begins. The outputs corresponding to Step 1 will turn ON and remain ON until Step 1 ends.
- When **C11** turns **ON**, Step 1 ends and Step 2 begins. C12, C13, and C14 end Steps 2, 3, and 4.
- **X001** must remain **ON** to enable the sequence to continue.
- At the end of **Step 4**, the sequence is complete, and the C8 bit turns ON.
- If **X003** turns **ON**, the Drum instruction jumps to the step number loaded in DS2.
- If **X004** turns **ON**, the Drum instruction jumps to the next step in the sequence.



Drum (EventBase)

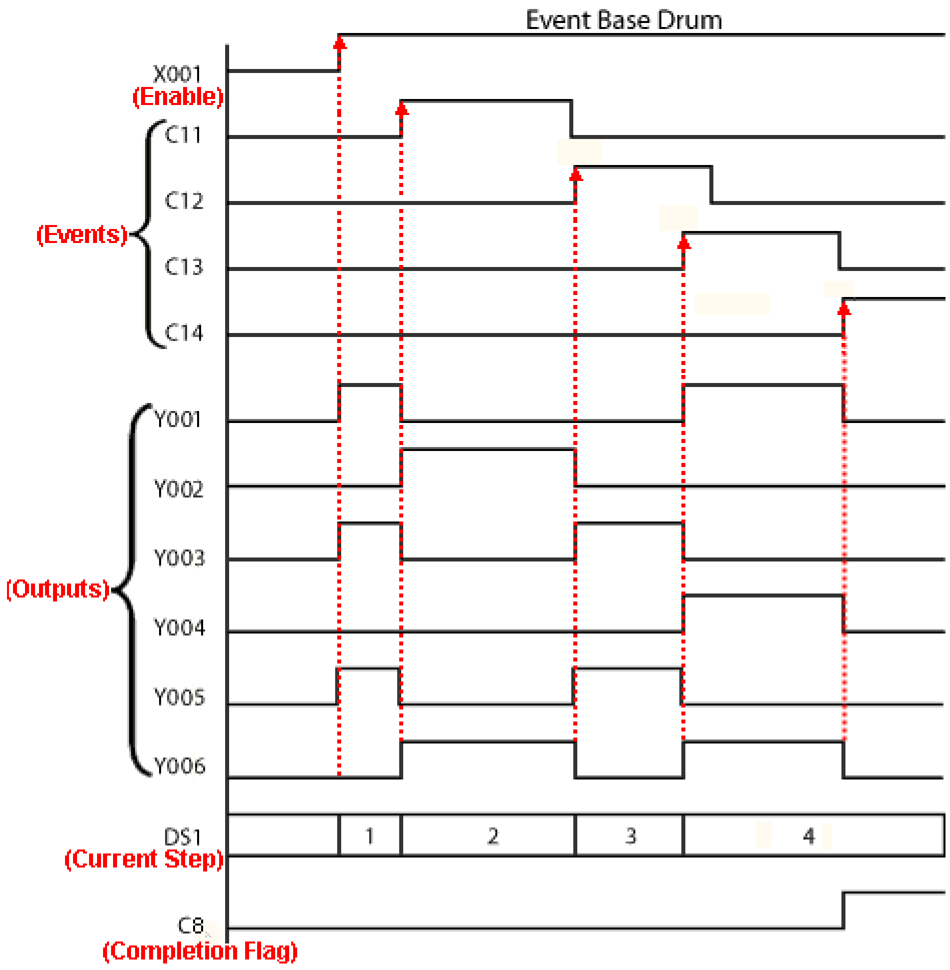
Step	Event	1	2	3	4	5	6
1	C11	■	□	■	□	■	□
2	C12	□	■	□	□	□	■
3	C13	□	□	■	□	■	□
4	C14	■	□	□	■	□	■

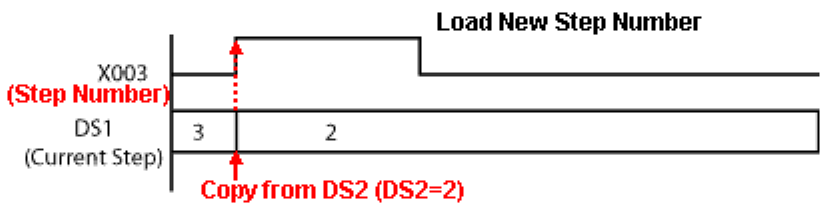
Output

1=Y001	5=Y005	9=	13=
2=Y002	6=Y006	10=	14=
3=Y003	7=	11=	15=
4=Y004	8=	12=	16=

New Step DS2

Current Step DS1





Related Topics:

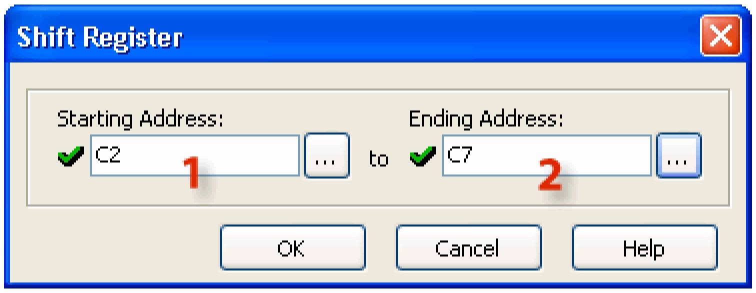
Drum Instruction: Time Base
Memory Addresses
Data Types

Shift Register

Definition

The **Shift Register** instruction **Shifts** a range of control bits with each **OFF-to-ON** transition of the **Clock** pulse. If the **Starting Address** is lower than the **Ending Address**, the **Shift register** will **Shift** from the **Starting Address** to the **Ending Address**. If the **Ending Address** is lower than **Starting Address** then **Shift Register** will **Shift** from **Ending Address** to the **Starting Address**.

Setup


Shift Register

Starting Address: ✓ C2 1 ... to Ending Address: ✓ C7 2 ...

✓ Entry required or invalid entry ✓ Valid entry

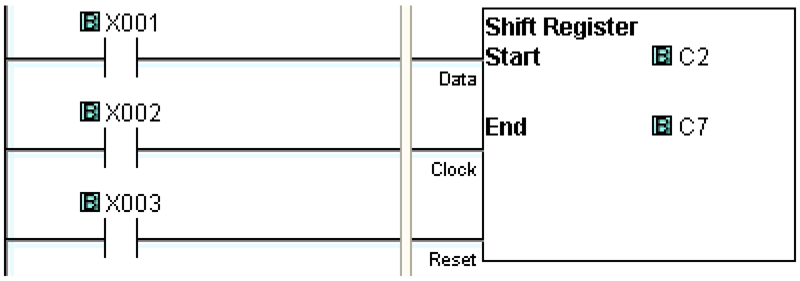
1Starting Address: Identify the first **ControlRelay** in the **ShiftRegister**.

2 Ending Address: Identify the last **ControlRelay** in the **ShiftRegister**.

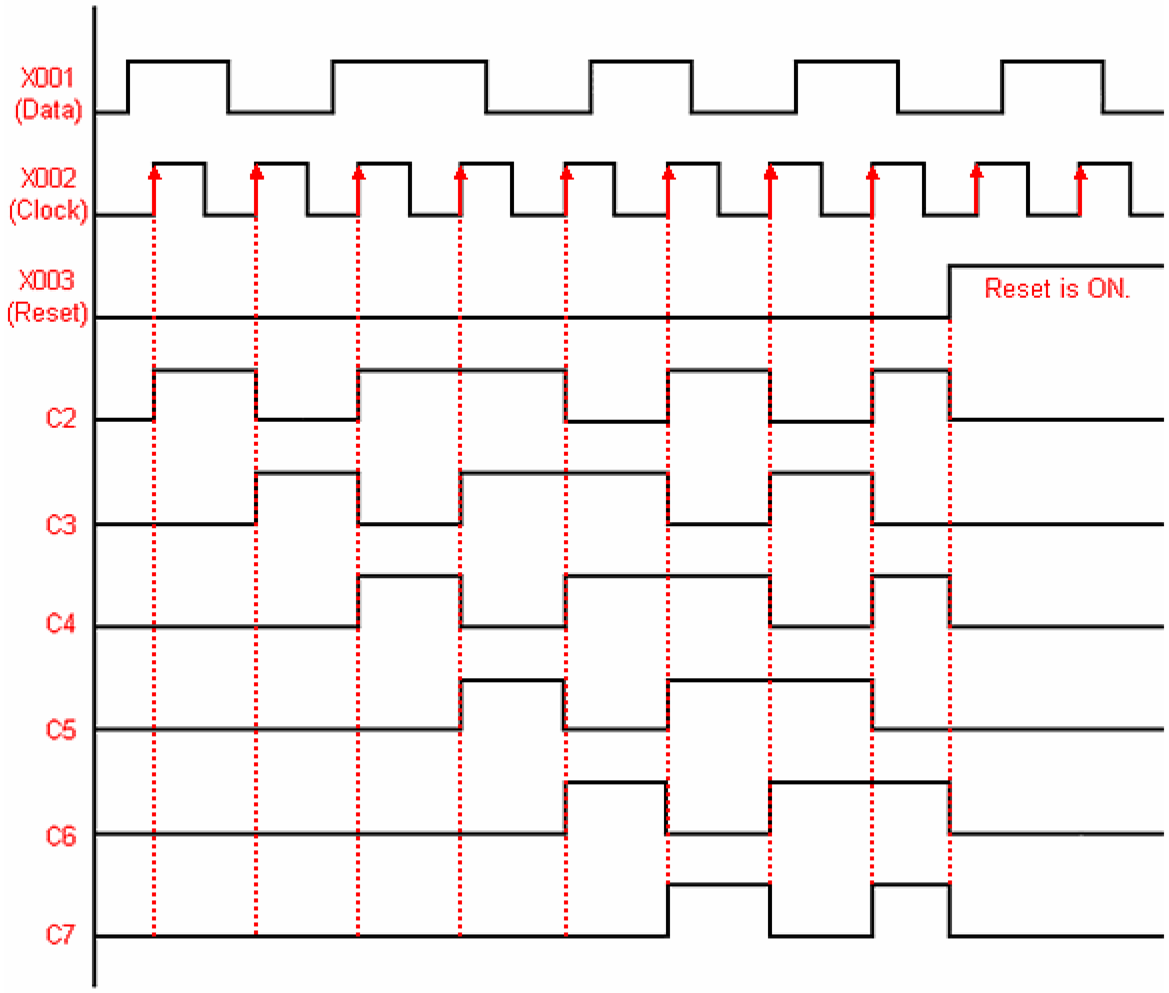
Example Program

Example Program: Shift Right

In the following example, **X001** represents the **Data** bit. When **X001** is **ON**, **C2** is **ON**. When **X001** is **OFF**, **C2** is **OFF**. **X002** represents the **Clock** bit. Each time **X002** transitions from **OFF-to-ON**, the current value (**OFF or ON**) in bits **C2** through **C7** shifts to the right by one memory location. **X003** represents the **Reset** bit. When **X003** is **ON** all bits in the **ShiftRegister** (**C2** through **C7** in the example) are turned **OFF**.



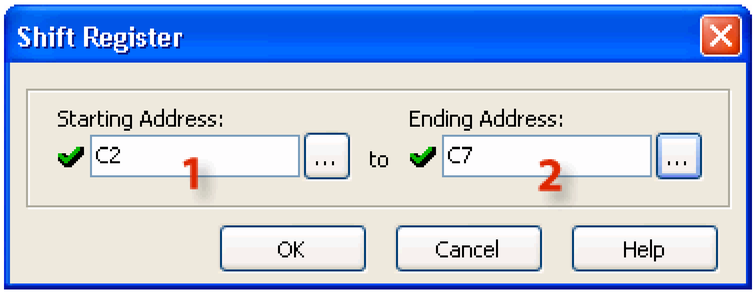
Timing Diagram



Definition

The **Shift Register** instruction **Shifts** a range of control bits with each **OFF-to-ON** transition of the **Clock** pulse. If the **Starting Address** is lower than the **Ending Address**, the **Shift register** will **Shift** from the **Starting Address** to the **Ending Address**. If the **Ending Address** is lower than **Starting Address** then **Shift Register** will **Shift** from **Ending Address** to the **Starting Address**.

Setup



✔ **Entry required or invalid entry** ✔ **Valid entry**

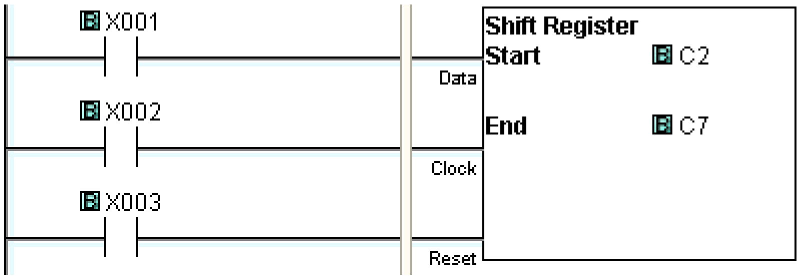
1Starting Address: Identify the first **ControlRelay** in the **ShiftRegister**.

2 Ending Address: Identify the last **ControlRelay** in the **ShiftRegister**.

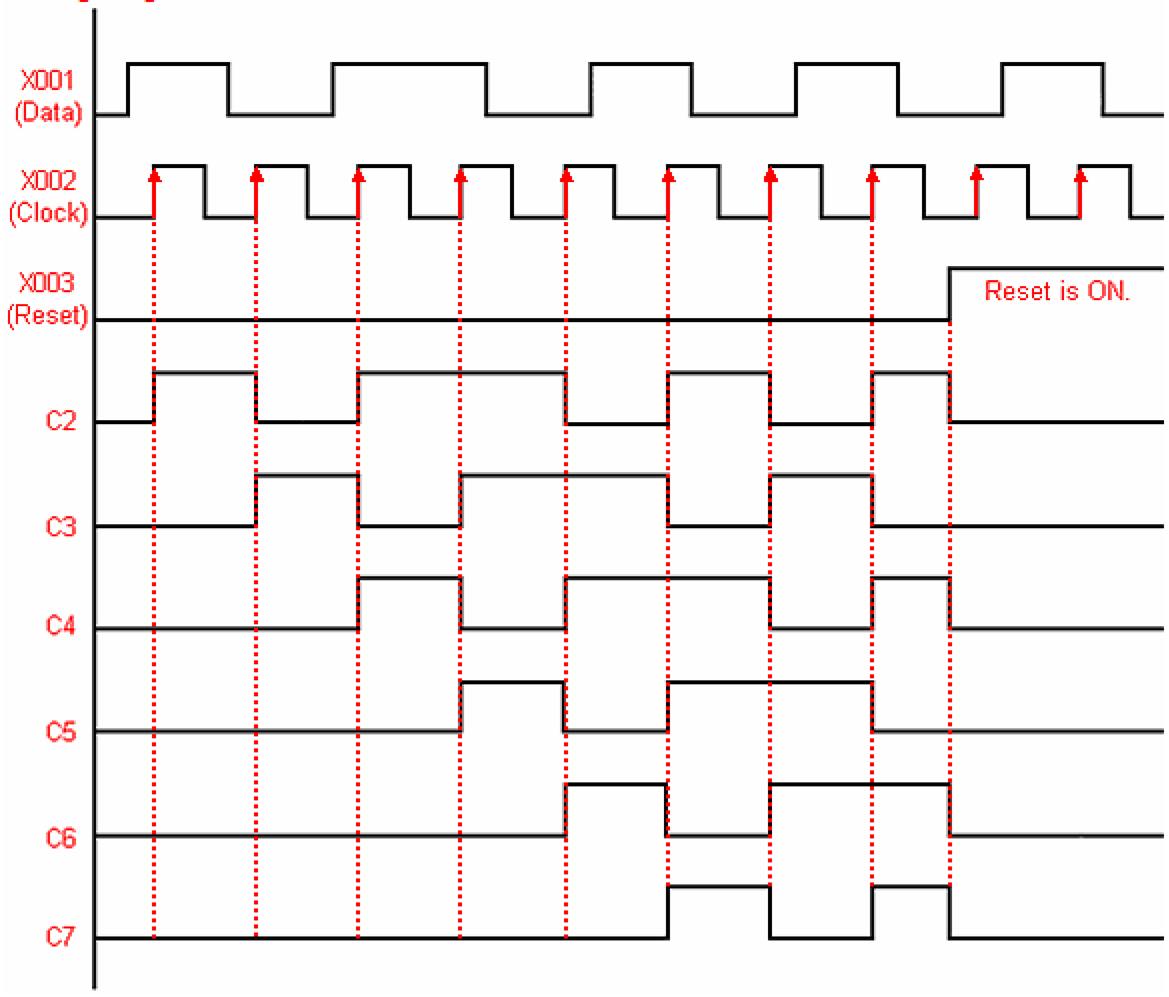
Example Program

Example Program: Shift Right

In the following example, **X001** represents the **Data** bit. When **X001** is **ON**, **C2** is **ON**. When **X001** is **OFF**, **C2** is **OFF**. **X002** represents the **Clock** bit. Each time **X002** transitions from **OFF-to-ON**, the current value (**OFF or ON**) in bits **C2** through **C7** shifts to the right by one memory location. **X003** represents the **Reset** bit. When **X003** is **ON** all bits in the **ShiftRegister** (**C2** through **C7** in the example) are turned **OFF**.



Timing Diagram



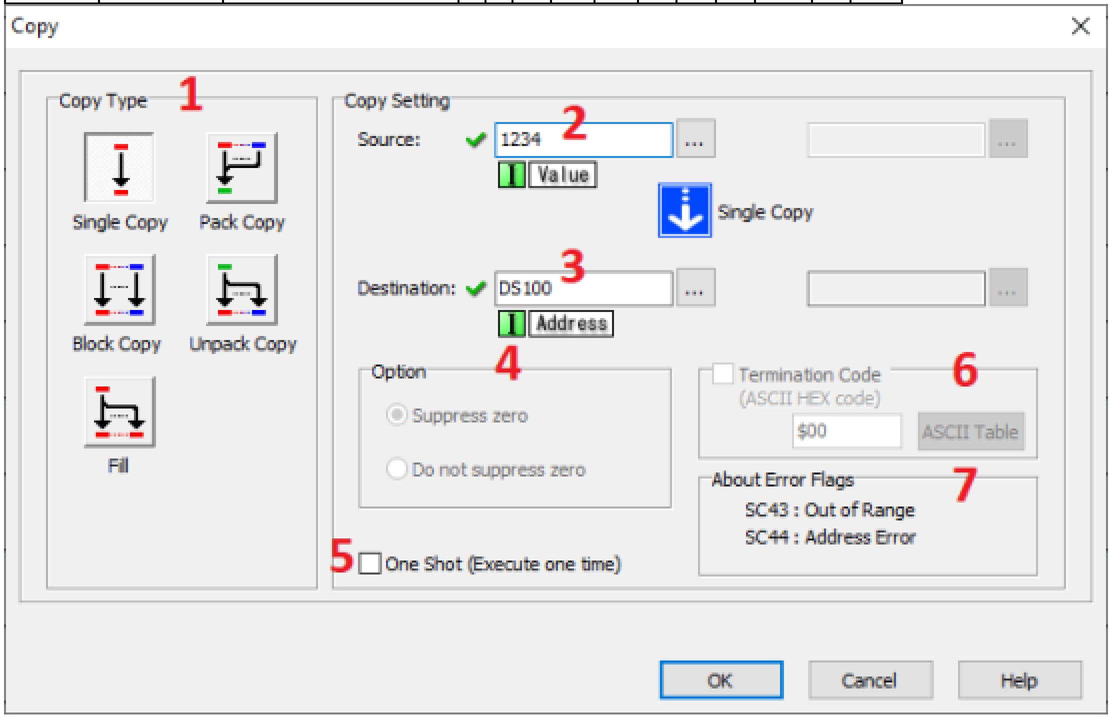
Copy Instruction: Single Copy

Definition

The **SingleCopy** instruction is used to copy a data or text value from its **Source** location to a specified **Destination** register. The **Source** location of the data or text can be another register, identified by its **MemoryAddress**, or it can be a constant value typed directly into the **Source** field on the dialog. The **Single Copy** instruction allows you to copy numerical text values to one or more data registers as integer values.

			Destination	
			Bits	**Registers**

			Y	C	DS	DD	DH	DF	YD	TD	CTD	SD	TXT
Source	**Bits**	X, Y, T, CT, SC	●	●									
		C	●	●									
	Registers	DS			●	●	●	●	●	●	●	●	●
		DD			●	●	●	●	●	●	●	●	●
		DH			●	●	●	●	●	●	●	●	●
		DF			●	●	●	●	●	●	●	●	●
		XD, YD, TD, CTD			●	●	●	●	●	●	●	●	●
		SD			●	●	●	●	●	●	●	●	●
		TXT			●	●	●	●	●	●	●	●	●
	Constant	Decimal, Hex			●	●	●	●	●	●	●	●	●
		String, ASCII Code											●


Copy

Copy Type 1

Single Copy | Pack Copy | Block Copy | Unpack Copy | Fill

Copy Setting

Source: 1234 2 Value

Single Copy

Destination: DS100 3 Address

Option 4
Suppress zero
Do not suppress zero

5 One Shot (Execute one time)

Termination Code (ASCII HEX code) 6
$00 ASCII Table

About Error Flags 7
SC43 : Out of Range
SC44 : Address Error

OK | Cancel | Help

Setup

1Copy Type: Select **Single Copy** to copy one **Source** value to a **Destination** location.

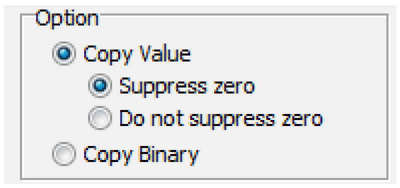
Note: In the case of text **Source** values, a single text string is copied to a specific number of consecutive registers. The number of registers is equal to the number of characters, including spaces, in the **Source** text string. (See example below.)

2 Source: Identify the **Source Memory Address** or type a constant data or text value directly into the **Source** field. (See Data Types for the required typing conventions.)

3 Destination: Identify the **Memory Address** to which the data or text is to be copied. The **Destination** address must accommodate the **Data Type** of the **Source** entry.

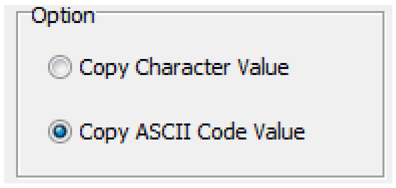
4 Option: This **Option** is used with certain combinations of **Source** and **Destination Data Types**.

4a When the **Source** is Numeric and the **Destination** is Text:


Option
[✓ Copy Value]
[✓ Suppress zero]
Do not suppress zero
Copy Binary

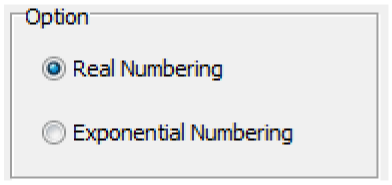
Source = DS1 (The value is 123.)
Destination = TXT1
The option '**Suppress zero**' is selected: TXT1-TXT3="123"
The option '**Do not Suppress zero**' is selected: TXT1-TXT5="00123"
The option '**Copy Binary**' is selected: TXT1="{" (123d or 7Bh)

4b When the **Source** is Text and the **Destination** is Numeric:


Option
Copy Character Value
[✓ Copy ASCII Code Value]

Source: TXT1 (The value is '5')
Destination: DS1
The option '**Copy Character Value**' is selected: DS1=5
The option '**Copy ASCII Code Value**' is selected: DS1=53 (35h)

4c When the **Source** is Float and the **Destination** is Text:


Option

[✓ Real Numbering]

Exponential Numbering

Source = DF1 (The value is 10000)
Destination = TXT1
The option '**Real Numbering**' is selected: TXT1-TXT13="10000.0000000"
The option '**Exponential Numbering**' is selected: TXT1-TXT13="1.0000000E+04"

5 One Shot: Choose **One Shot** to execute the **Single Copy** instruction one time when the enabling rung makes an **OFF-to-ON** transition. Otherwise the instruction will execute every scan. If **One Shot** is selected, the **One Shot** symbol will appear adjacent to the **Coil** in the **Ladder Editor**.

6 Termination Code: This option is supported by **C0-1x** and **C2-x** CPUs. When the Destination is Text Registers the Termination Code option becomes available. A single termination character can be added to the length of the Destination. Click on the checkbox to select and enter the ASCII Code in the field. Click on the ASCII Table button to open the ASCII Table shown below. Use this table to quickly select the desired ASCII Code.

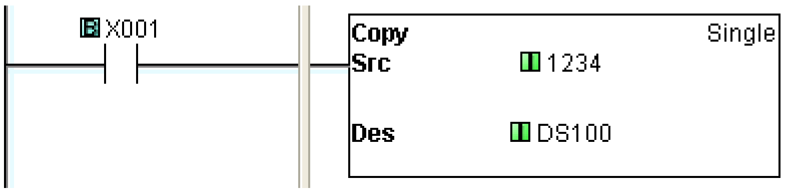
7About Error Flag: **SC43** and **SC44** are **Error Flags** available for use in your program. **SC43 Out of Range** is valid for Single, Block, and Pack Copy modes. **SC44 Address Error** is valid for Single Copy mode when using a Pointer Address. The **Source** or **Destination** Pointer Address is out of range of the memory type.

Note: When the **Source** is a **DH** address type and the option **'Do not Suppress zero'** is selected, the value in the **DH** address is copied into **4 TXT** addresses.
Example:
Source: DH1 (The value is 12h.)
Destination: TXT1
(1) The option **'Suppress zero' is selected: TXT1 = '1', TXT2 = '2'**
(2) The option **'Do not Suppress zero'' is selected: TXT1 = '0', TXT2 = '0', TXT3 = '1', TXT4 = '2'**

Example Program

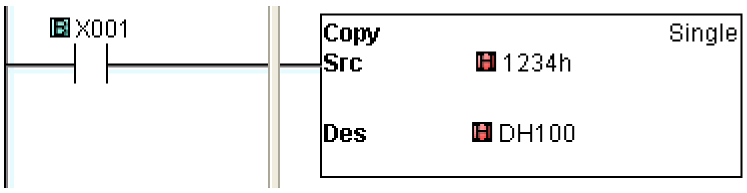
Example Program 1: Copy Decimal Data Constant

In the following example, when **X001** is **ON**, the decimal constant, **1234**, is copied to the **Memory Address: DS100**.



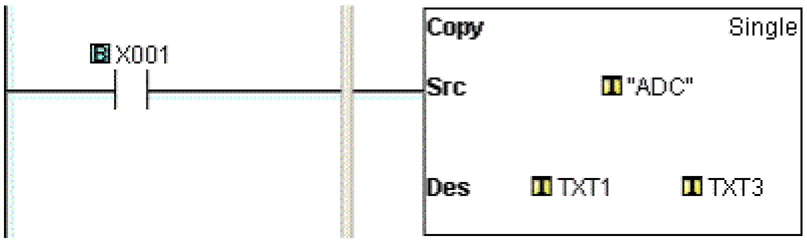
Example Program 2: Copy Hex Constant

In the following example, when **X001** is **ON**, the hexadecimal constant, **1234h**, is copied to the **Memory Address: DH100**.



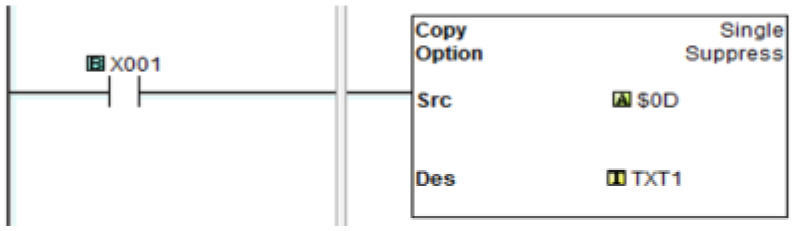
Example Program 3: Copy Text

In the following example, when **X001** is **ON**, the text value **"ADC"** is copied to the **Memory Addresses: TXT 1 through TXT3**.



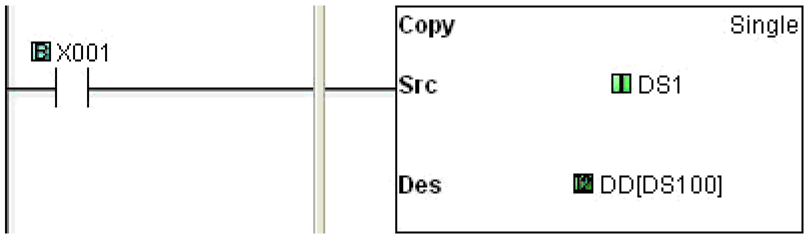
Example Program 4: Copy an ASCII Code

In the following example, when **X001** is **ON**, **ASCII** character **$0D (CR)** is copied to **TXT1**. See **ASCII Table** for the entire **ASCII** codes.



Example Program 5: Using Pointer Addressing

In the following example, **DD[DS100]** is the **Pointer Addressing**. When **DS100 = 10**, **DD[DS100]** is identical to **DD10**. When **X001** is **ON**, the data in **DS1** is copied to **DD10**.



Related Topics:

Block Copy
Fill
Pack Copy
Unpack Copy

Casting Datatypes

Pointer Addressing

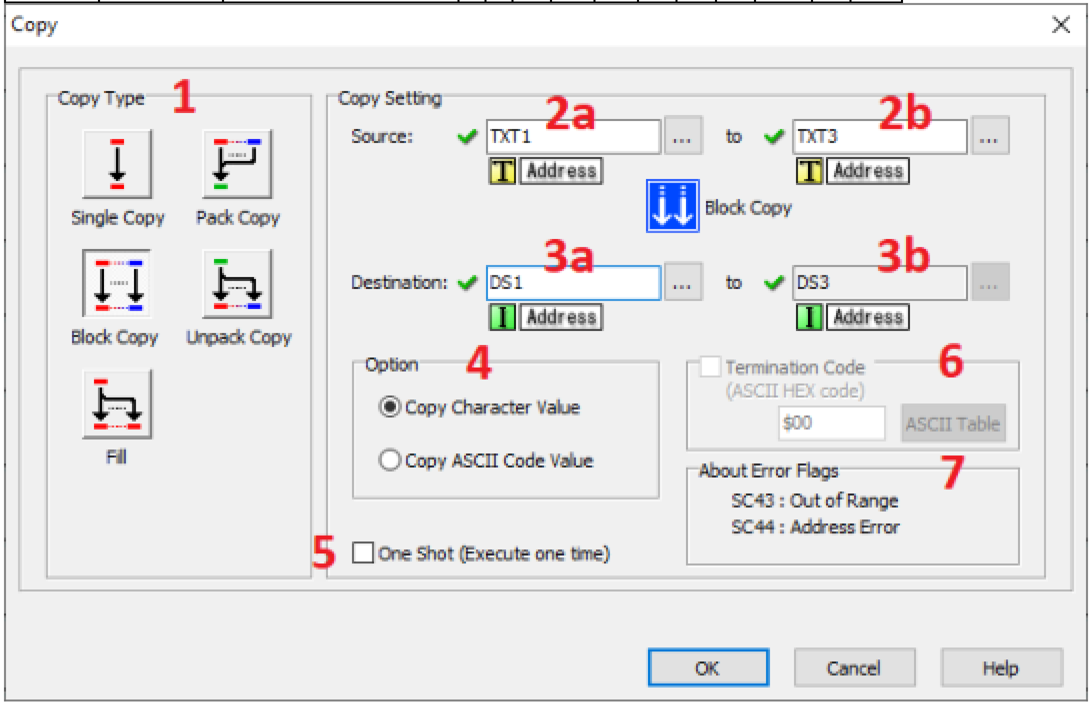
Copy Instruction: Block Copy

Block Copy Definition

The **BlockCopy** instruction is used to copy data or text from its **Source** registers to specified sequential **Destination** registers. The **Source** location of the data or text is identified by its beginning and ending **MemoryAddresses**, and the **Destination** registers are identified by the **Memory Address** of the first register in the sequence.

			Destination										
			Bits		**Registers**								
			Y	C	DS	DD	DH	DF	YD	TD	CTD	SD	TXT
Source	**Bits**	X, Y, T, CT, SC	●	●									
		C	●	●									
	Registers	DS			●	●	●	●		●	●	●	
		DD			●	●	●	●		●	●	●	
		DH			●	●	●	●		●	●	●	
		DF			●	●	●	●		●	●	●	

		XD, YD, TD, CTD											
		SD			●	●	●	●		●	●	●	
		TXT			●	●	●						●
	Constant	Decimal, Hex											
		String, ASCII Code											



Source: ✓ TXT1 to ✓ TXT3

Destination: ✓ DS1 to ✓ DS3

Option: [✓ Copy Character Value] / Copy ASCII Code Value

One Shot (Execute one time)

Termination Code (ASCII HEX code) $00 ASCII Table

About Error Flags
SC43 : Out of Range
SC44 : Address Error

✓ Entry required or invalid entry ✓ Valid entry

Setup

1 Copy Type: Select **Block Copy** to copy multiple registers from their **Source** locations to **Destination** locations.

2a Source: Identify the beginning **Source Memory Address**.

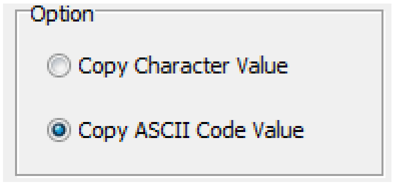
2b Source: Identify the ending **Source Memory Address**. The ending **Source Memory Address** must be the same type as the beginning **Source Memory Address**.

3a Destination: Identify the beginning **Destination Memory Address**. The **Destination** address must accommodate the **Data Type** of the **Source** entry.

3b Destination: The identity of the ending **Destination Memory Address** will be calculated by the **CLICK** programming software and will be visible in this box.

4 One: This **Option** field is used with certain combinations of **Source** and **Destination Data Types**. There is only a single Option available in **Block Copy** mode.

When the Source is Text and the Destination is Numeric:



Copy Character Value [✓ Copy ASCII Code Value]

Source: TXT1-TXT3 (The values are ‘1’, ‘2’, ‘3’)
Destination: DS1-DS3
The option ‘**Copy Character Value**’ is selected: DS1=1, DS2=2, DS3=3
The option ‘**Copy ASCII Code Value**’ is selected: DS1=49 (31h), DS2=50 (32h), DS3=51 (33h)

5 One Shot: Choose One Shot to execute the Block Copy instruction one time when the enabling rung makes an OFF-to-ON transition. If One Shot is selected, the One Shot symbol will appear adjacent to the Coil in the Ladder Editor.

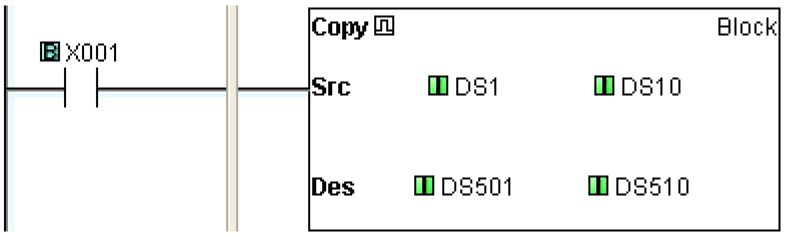
6 Termination Code: This option is supported by **C0-1x** and **C2-x** CPUs. When the Destination is Text Registers the Termination Code option becomes available. A single termination character can be added to the length of the Destination. Click on the checkbox to select and enter the ASCII Code in the field. Click on the ASCII Table button to open the ASCII Table shown below. Use this table to quickly select the desired ASCII Code.

7About Error Flag: SC43 and **SC44 are Error Flags** are available for use in your program. **SC43 Out of Range** is valid for Single, Block, and Pack Copy modes. **SC44 Address Error** is not used in Block Copy mode.

Example Program

Example Program: Block Copy Data Registers

In the following example, when **X001** transitions from **OFF-to-ON** (**One Shot** is selected), the values in **Memory Addresses DS1** through **DS10** are copied to **Memory Addresses DS501** through **DS510**.



Related Topics:

[Single Copy](#)
[Fill](#)
[Pack Copy](#)
[Unpack Copy](#)

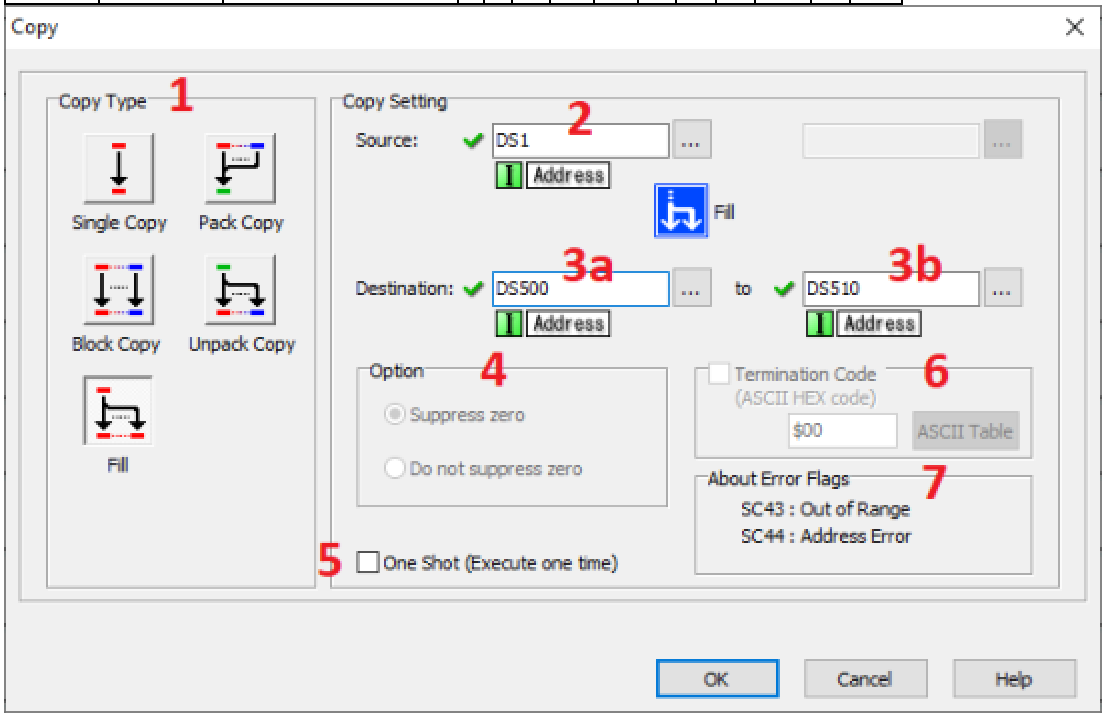
Copy Instruction: Fill

Fill Definition

The **Fill** mode of the **Copy Instruction** is used to copy a data or text value from its **Source** location to a specified range of **Destination** registers. The **Source** location of the data or text can be another register, identified by its **Memory Address**, or it can be a constant value typed directly into the **Source** field on the dialog. Constant data or text values must follow the typing conventions for that Data Type.

			Destination										
			Bits		**Registers**								
			Y	C	DS	DD	DH	DF	YD	TD	CTD	SD	TXT
Source	**Bits**	X, Y, T, CT, SC											
		C											
	Registers	DS			●	●	●	●	●	●	●	●	
		DD			●	●	●	●	●	●	●	●	
		DH			●	●	●	●	●	●	●	●	
		DF			●	●	●	●	●	●	●	●	
		XD, YD, TD, CTD			●	●	●	●	●	●	●	●	

		SD			●	●	●	●	●	●	●	●	
		TXT											●
	Constant	Decimal, Hex			●	●	●	●	●	●	●	●	●
		String, ASCII Code											●



Source: ✓ DS1

Destination: ✓ DS500 to ✓ DS510

Termination Code (ASCII HEX code) $00

One Shot (Execute one time)

About Error Flags
SC43 : Out of Range
SC44 : Address Error

✓ **Entry required or invalid entry** ✓ **Valid entry**

Setup

1 Copy Type: Select **Fill** to copy from a single data or text source to multiple **Memory Addresses**.

2 Source: Identify the **Source Memory Address** or type a constant data or text value in the **Source** field.

3a Destination: Identify the beginning **Destination Memory Address**. The **Destination** address must accommodate the **Data Type** of the **Source** entry.

3b Destination: Identify the ending **Destination Memory Address**.

4 Option: This **Option** field is not used in **Fill** mode.

5 One Shot: Choose **One Shot** to execute the **Fill** instruction one time when the enabling rung makes an **OFF-to-ON** transition. If **One Shot** is selected, the **One Shot symbol** will appear adjacent to the **Coil** in the **Ladder Editor**.

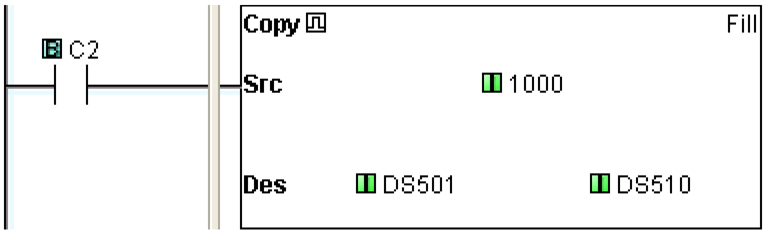
6 Termination Code: This option is supported by **C0-1x** and **C2-x** CPUs. When the Destination is Text Registers the Termination Code option becomes available. A single termination character can be added to the length of the Destination. Click on the checkbox to select and enter the ASCII Code in the field. Click on the ASCII Table button to open the ASCII Table shown below. Use this table to quickly select the desired ASCII Code.

7 About Error Flag:SC43 and **SC44** are **Error Flags** available for use in your program. They are not used in **Fill** mode.

Example Program

Example Program: Fill Destination with Data Constant

In the following example, when **C2** transitions from **OFF-to-ON** (One Shot is selected), the **Source** constant **1000** is loaded in **Memory Addresses DS501** through **DS510**.



Related Topics:

Single Copy
Block Copy
Pack Copy
Unpack Copy

Copy Instruction: Pack Copy

Description

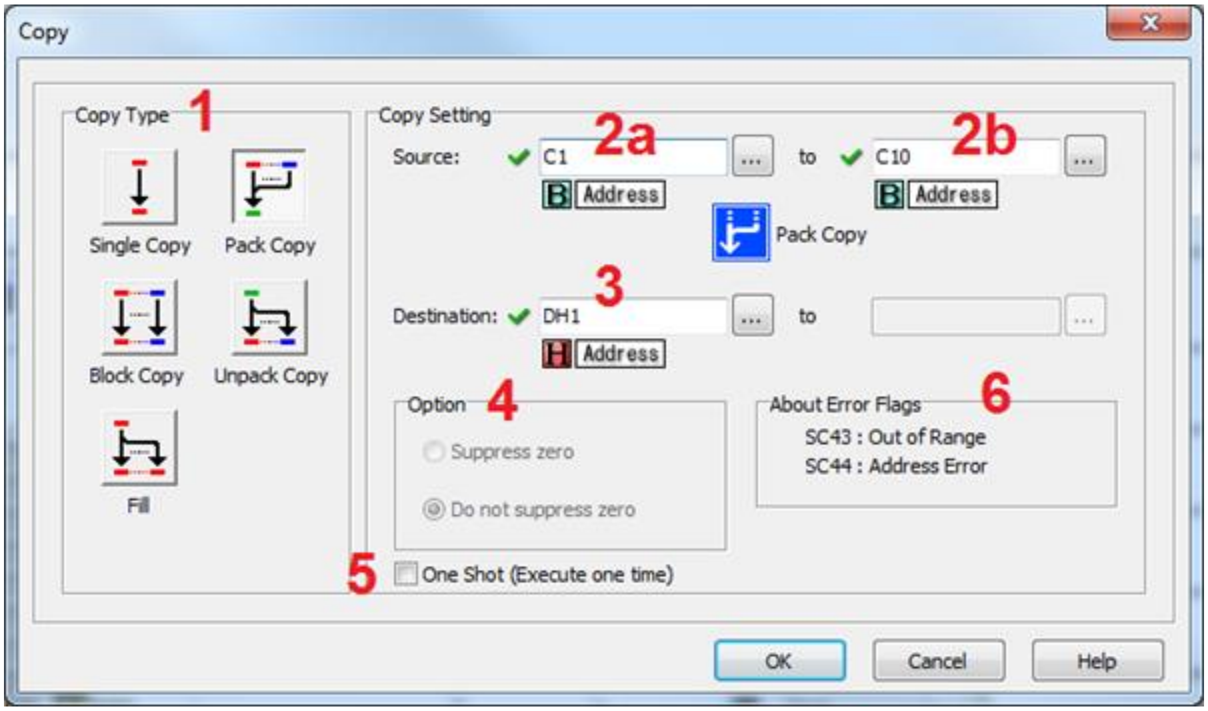
The **Pack Copy** mode of the **Copy Instruction** supports the following functions:

- Combine the status of up to **16 Source Bit Memory Addresses** (**X, Y, C, T, CT** or **SC**) and copy the combined status into a **Destination Data Register** (**DS** or **DH**).

- Combine the status of up to 32 Source Bit Memory Addresses (C) and copy the combined status into a Destination Data Register (DD or DF).
- Combine the status of two Single-Word Source Word Memory Addresses (DS or DH) and copy the combined status into a Double-Word Destination Data Register (DD or DF).
- Convert the numerical ASCII characters stored in a series of the **TXT Memory Addresses** to numerical data and store into a **Destination Data Register** (**DS, DD, DH, DF, TD** or **CTD**).

Please refer to the example programs on the bottom of this help topic.

			Destination										
			Bits		**Registers**								
			Y	C	DS	DD	DH	DF	YD	TD	CTD	SD	TXT
Source	**Bits**	X, Y, T, CT, SC					●						
		C			●	●	●	●					
	Registers	DS				●		●					
		DD											
		DH				●							
		DF											
		XD, YD, TD, CTD											
		SD											
		TXT			●	●	●	●		●	●	●	
	Constant	Decimal, Hex											
		String, ASCII Code											



Source: ✓ C1 to ✓ C10

Destination: ✓ DH1

About Error Flags
SC43 : Out of Range
SC44 : Address Error

☐ One Shot (Execute one time)

✓ **Entry required or invalid entry** ✓ **Valid entry**

1Copy Type: Select **Pack Copy** to copy from a range of data to a single **Memory Address**.

2a Source: Enter a **BitMemory Address** that represents the start of a range of a **Bit Memory Address**, Word Memory Address or **TXT Memory Address**.

2b Source: Identify the ending **Source Memory Address**. The ending **Source Memory Address** must be the same type as the beginning **Source Memory Address**.

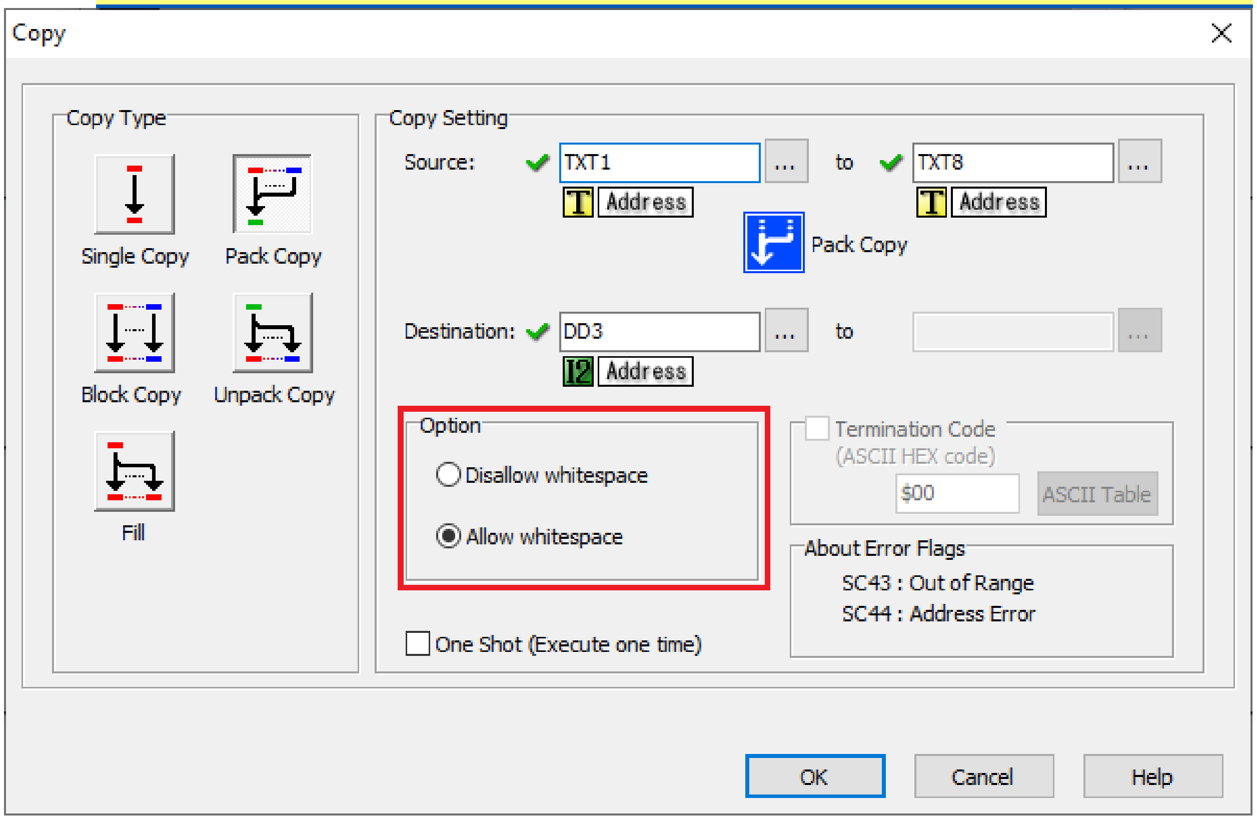
3 Destination: Identify the **Destination Memory Address**. If you selected a series of **X, Y, C, T CT** or **SC Memory Addresses** as the **Source**, select a **DS**, **DD**, **DH** or **DFMemory Address** as the **Destination**. If you selected a series of **TXT Memory Addresses** as the source, select a **DS**, **DD**, **DH**, **DF**, **TD** or **CTD Addresses** as the **Destination**.

4 This Option field is not used in this mode.

5 One Shot: Choose **One Shot** to execute the **Pack Copy** instruction one time when the enabling rung makes an **OFF-to-ON** transition. If **One Shot** is selected, the **One Shot symbol** will appear adjacent to the **Coil** in the **Ladder Editor**.

6About Error Flag: SC43 and **SC44** are **Error Flags** available for use in your program. **SC43 Out of Range** is valid for Single, Block, and Pack Copy modes. **SC44 Address Error** is valid for Single Copy mode when using a Pointer Address.

Note: If one of those **TXT Memory Addresses** has non-numerical ASCII character like 'A', this **Copy instruction** won't convert the ASCII characters to a numerical data. Instead, **System Control Bit SC43 'Out of Range'** will turn on to indicate the error condition.



Source: TXT1 to TXT8

Pack Copy

Destination: DD3 to

Option
- [ ] Disallow whitespace
- [✓] Allow whitespace

- [ ] Termination Code (ASCII HEX code) $00

About Error Flags
SC43 : Out of Range
SC44 : Address Error

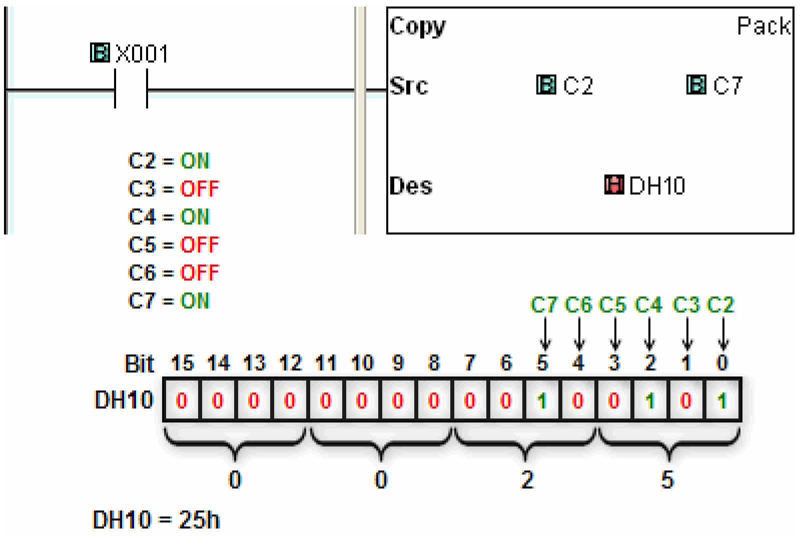
- [ ] One Shot (Execute one time)

Note: C0-1x and C2-x CPUs have an additional Option of **Allow whitespace** when converting ASCII TXT registers into a numeric register. Leading and Trailing spaces of the Source are ignored.

Example Programs

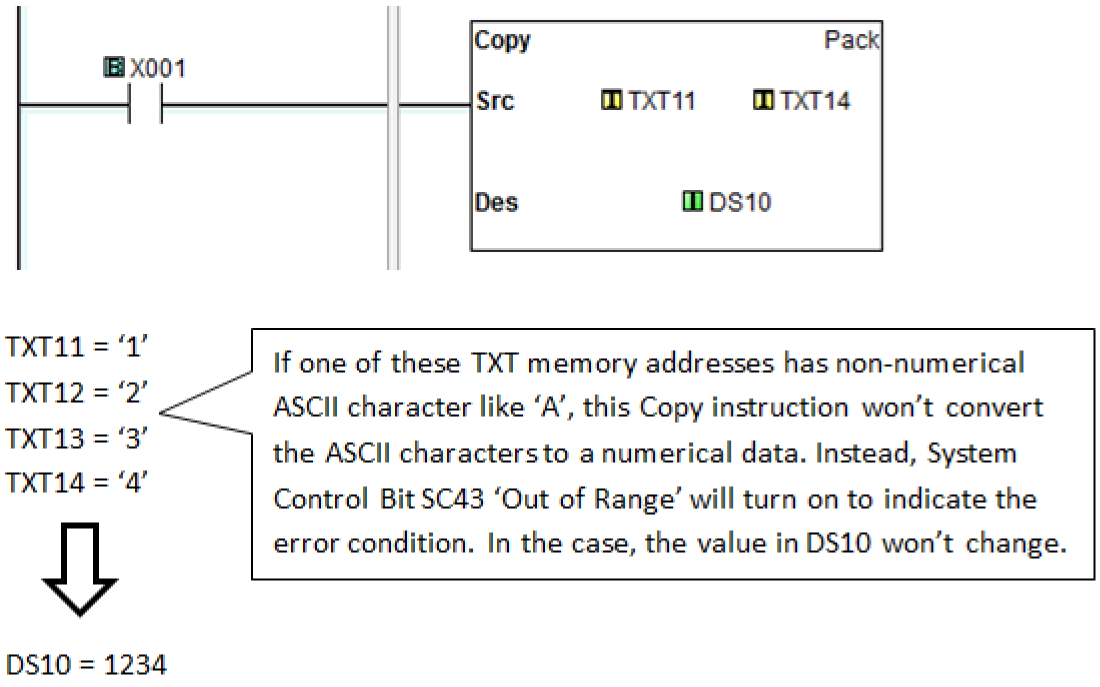
Example Program: Pack Copy a Range from C2 to C7

In the following example, when **X001** is **ON**, the **Source** range **C2** through **C7** is loaded in **Memory Address DH10**.



Example Program 2: Pack Copy numerical data (integer) from TXT11 to TXT14

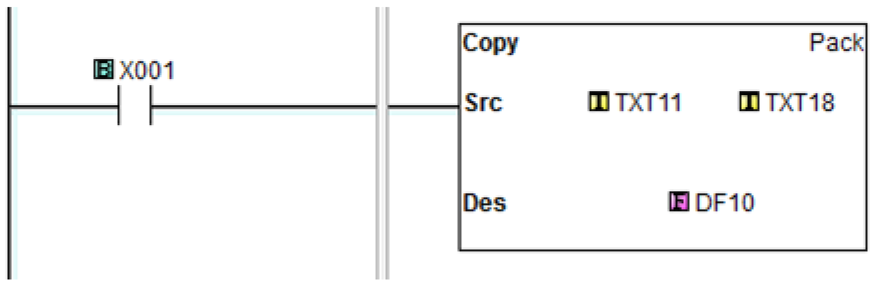
In the following example, when **X001** is **ON**, the numerical ASCII characters stored from **TXT11** to **TXT15** are converted to a numerical data and loaded in **Memory Address DS10**.

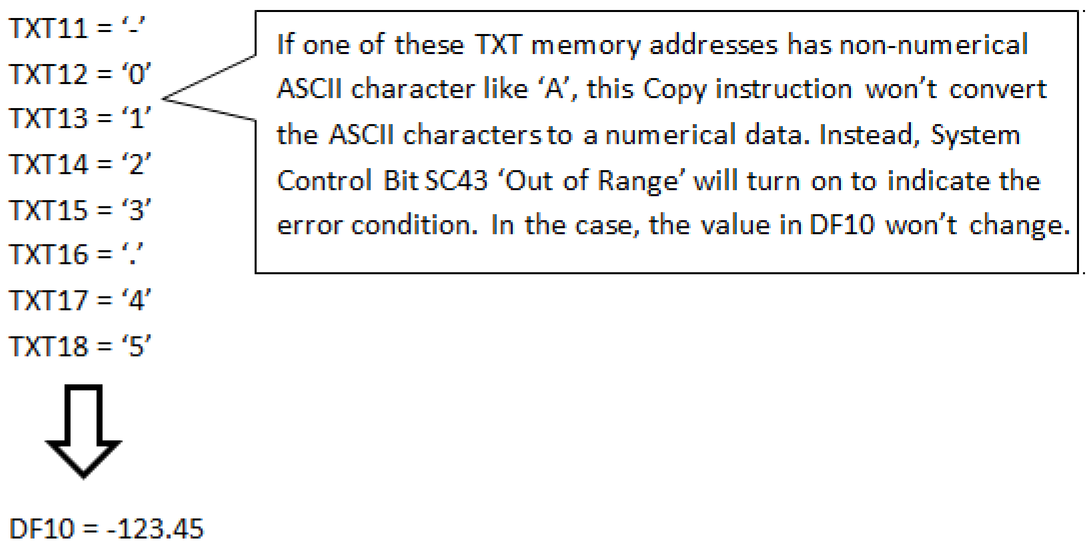


If one of these TXT memory addresses has non-numerical ASCII character like 'A', this Copy instruction won't convert the ASCII characters to a numerical data. Instead, System Control Bit SC43 'Out of Range' will turn on to indicate the error condition. In the case, the value in DS10 won't change.

Example Program 3: Pack Copy numerical data (floating point) from TXT11 to TXT18

In the following example, when **X001** is **ON**, the numerical ASCII characters stored from **TXT11** to **TXT18** are converted to a numerical data and loaded in **Memory Address DF10**.




TXT11 = '-'
TXT12 = '0'
TXT13 = '1'
TXT14 = '2'
TXT15 = '3'
TXT16 = '.'
TXT17 = '4'
TXT18 = '5'

If one of these TXT memory addresses has non-numerical ASCII character like 'A', this Copy instruction won't convert the ASCII characters to a numerical data. Instead, System Control Bit SC43 'Out of Range' will turn on to indicate the error condition. In the case, the value in DF10 won't change.

DF10 = -123.45

TXT to DF Examples

TXT1-TXT6 = "-12.34", DF=-12.34

TXT1-TXT6 = "+12.34", DF=12.34

TXT1-TXT6 = " 12.34", SC43 Out of Range = True, Leading Space not allowed. (Allow whitespace enabled, DF=12.34)

TXT1-TXT6 = "12.34 ", SC43 Out of Range = True, Trailing Space not allowed. (Allow whitespace enabled, DF=12.34)

TXT1-TXT6 = "12,345", SC43 Out of Range = True, Thousands separator not allowed.

TXT1-TXT6 = "-01234", DF=-1234

TXT1-TXT6 = "1234e2", DF=123400

TXT1-TXT6 = "1234E2", DF=123400

TXT1-TXT6 = "123e-2", DF=1.23

TXT1-TXT10 = "-01234e-02", DF=-12.34

TXT to DS, DD, TD, or CTD Examples

TXT1-TXT6 = “123456”, DD=123456

TXT1-TXT6 = “-12345”, DD=-12345

TXT1-TXT6 = “+12345”, DD=12345

TXT1-TXT6 = “000123”, DD=123

TXT1-TXT6 = “ 12345”, SC43 Out of Range = True, Leading Space not allowed. (Allow whitespace enabled, DD=12345)

TXT1-TXT6 = “12345 ”, SC43 Out of Range = True, Trailing Space not allowed. (Allow whitespace enabled, DD=12345)

TXT1-TXT6 = “12,345”, SC43 Out of Range = True, Comma separator not allowed.

TXT1-TXT6 = “12.345”, SC43 Out of Range = True, Period separator not allowed.

TXT1-TXT6 = “1234e2”, SC43 Out of Range = True, Exponential notation not allowed.

TXT to DH Examples

TXT1-TXT4 = “1234”, DH=0x1234

TXT1-TXT4 = “abcd”, DH=0xABCD

TXT1-TXT4 = “ABCD”, DH=0xABCD

TXT1-TXT4 = “ ABC”, SC43 Out of Range = True, Leading Space not allowed. (Allow whitespace enabled, DH=ABC)

TXT1-TXT4 = “ABC ”, SC43 Out of Range = True, Trailing Space not allowed. (Allow whitespace enabled, DH=ABC)

TXT1-TXT6 = “A,BC”, SC43 Out of Range = True, Comma separator not allowed.

TXT1-TXT6 = “A.BC”, SC43 Out of Range = True, Period separator not allowed.

Related Topics:

Single Copy
Block Copy

Fill
Unpack Copy

Copy Instruction: Unpack Copy

Definition

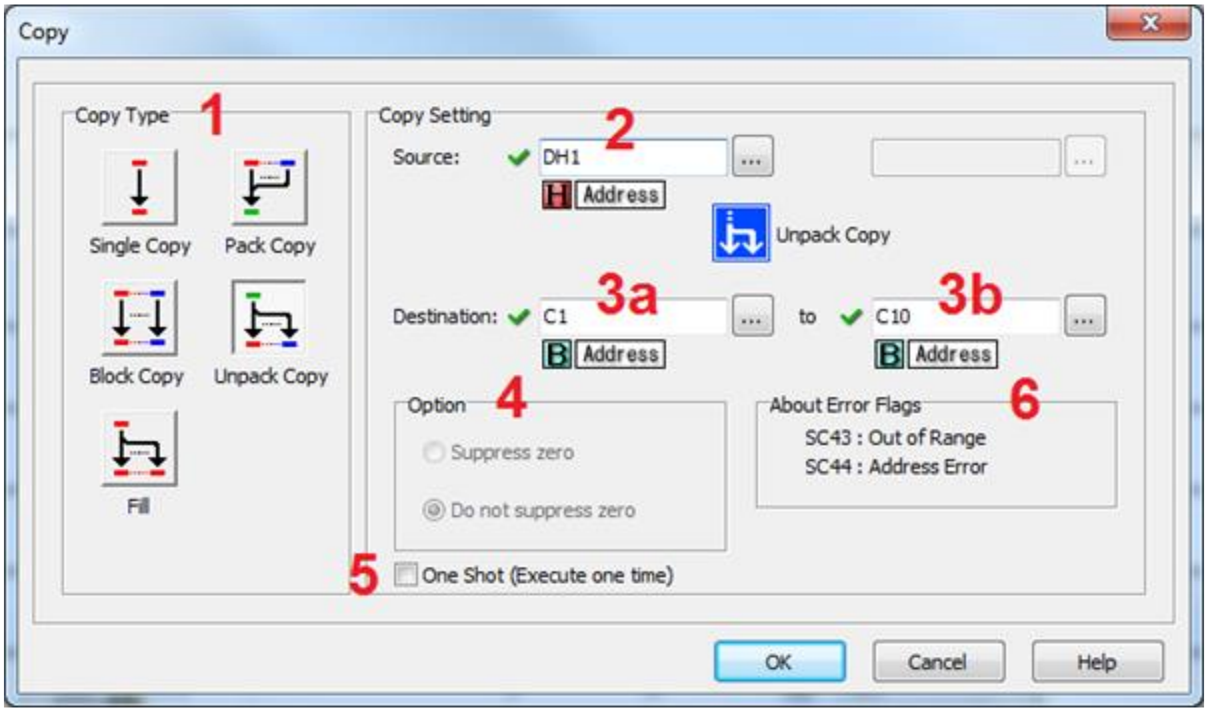
The **Unpack Copy** mode of the **Copy Instruction** supports the following functions:

- Separate the status of a **Source Data Register** (**DS** or **DH**) and copy into up to 16 **Destination Bit Memory Addresses** (**Y** or **C**).
- Separate the status of a **Source Data Register** (**DD** or **DF**) and copy into up to 32 **Destination Bit Memory Addresses** (**Y** or **C**).
- Separate the Double-Words **Source Data Register** (**DD** or **DF**) and store into two Single-Word **Destination Word Memory Addresses** (**DS** or **DH**).

Refer to the example programs at the end of this Help topic.

			Destination										
			Bits		**Registers**								
			Y	C	DS	DD	DH	DF	YD	TD	CTD	SD	TXT
Source	**Bits**	X, Y, T, CT, SC											
		C											
	Registers	DS		●									
		DD		●	●		●						
		DH	●	●									
		DF		●	●		●						
		XD, YD, TD, CTD											
		SD											
		TXT											
	Constant	Decimal, Hex											

		String, ASCII Code											


Copy

Copy Type 1

Single Copy, Pack Copy, Block Copy, Unpack Copy, Fill

Copy Setting

Source: DH1 2

Unpack Copy

Destination: C1 3a to C10 3b

Option 4

Suppress zero

Do not suppress zero

About Error Flags 6

SC43 : Out of Range

SC44 : Address Error

5 One Shot (Execute one time)

OK Cancel Help

Setup

1 Copy Type: Select **Unpack Copy** to copy from a single **Memory Address**.

2 Source: Enter a **DS** or **DH Memory Address (DD** or **DF Double-Word** that has the data to **Copy** to the destinations.

3a Destination: For Word Sources, enter the starting **C** or **Y Address**. For **Double-Word** sources, enter the starting **C**, **Y, DS**, or **DH Address**.

3b Destination: Enter the ending address (Max 32 consecutive **Bit** addresses or 2 consecutive **Word** addresses.).

4Option: The **Option** field is not used in this mode.

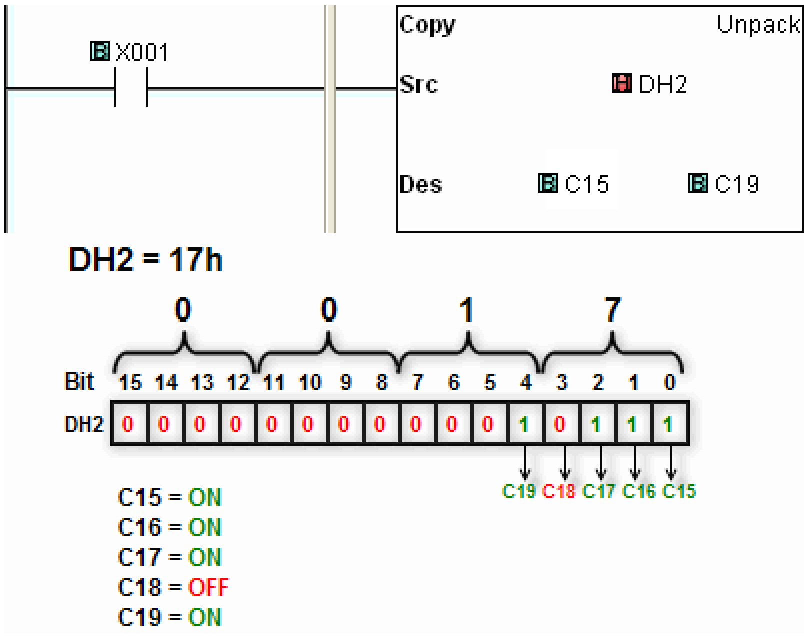
5 One Shot: Choose **One Shot** to execute the **Pack Copy** instruction one time when the enabling rung makes an **OFF-to-ON** transition. If **One Shot** is selected, the **One Shot symbol** will appear adjacent to the **Coil** in the **Ladder Editor**.

6About Error Flag:SC43 and **SC44** are **Error Flags** available for use in your program. **SC43 Out of Range** is valid for Single, Block, and Pack Copy modes. **SC44 Address Error** is valid for Single Copy mode when using a Pointer Address.

Example Program

Example Program: Unpack Copy a Range to C15 through C19

In the following example, when **X001** is **ON**, the **SourceDH2** is copied to **Destinations C15** through **C19**.



Related Topics:

Single Copy
Block Copy
Fill
Pack Copy
Casting Datatypes
Pointer Addressing

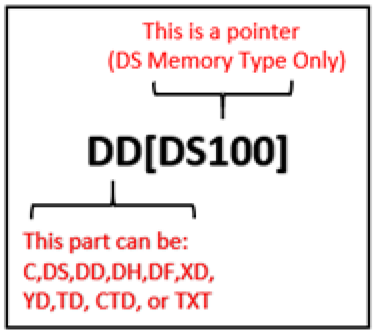
Pointer Addressing

Pointer

Addressing

In certain cases, the **CLICK PLC** allows the use of **Pointer Addressing** for flexibility in programming.

Only the **DS Memory Type** can be used as **Pointer**. **Pointer Addressing** uses the **Pointer's** data value to **Point** to a **Memory** location within the range of one of the eligible **Memory** types. **Pointer Addressing** can be used for the **C**, **DS**, **DD**, **DH**, **DF**, **XD**, **YD**, **TD**, **CTD** and **TXT Memory** types.



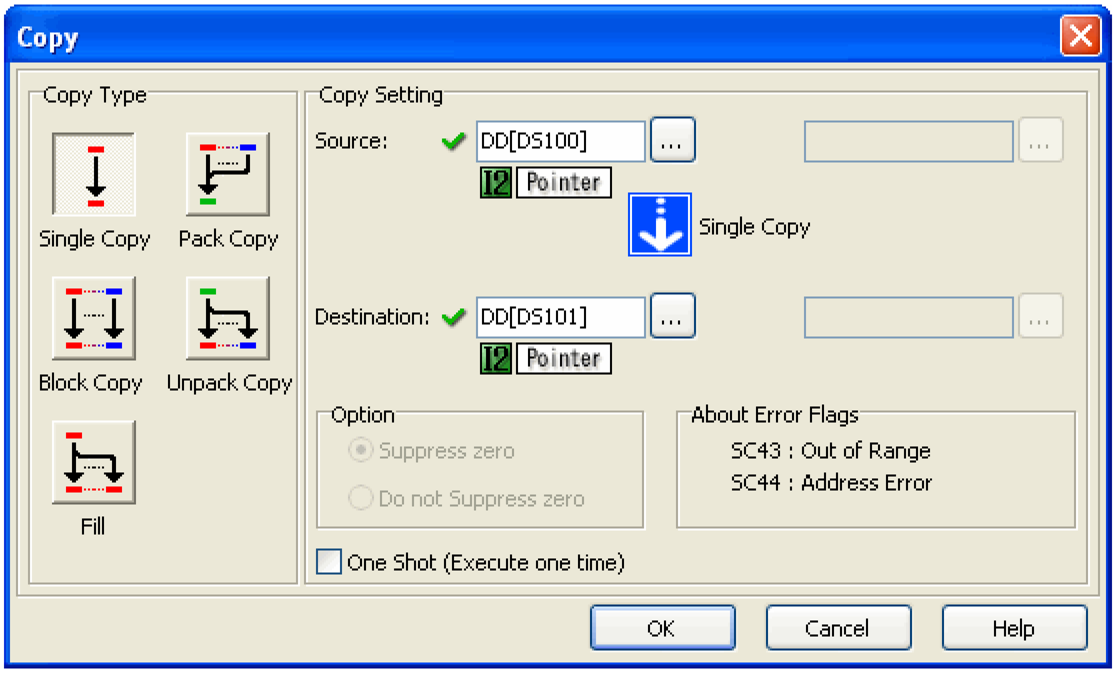
Pointer Addressing Example 1

DS1 = 100

DD[DS1] means **DD100**

In the above example, **DS1** is a **Pointer**. **DD[DS1]** is called **Pointer Addressing**. **DD[DS1]** is identical to **DD100** in this case.

Important: Currently, only the **Copy** instruction supports **Pointer Addressing** in the **Single Copy** mode. The **Pointer Addressing** can be used for the **Source** and/or **Destination**as shown below.



Source: DD[DS100] Pointer

Single Copy

Destination: DD[DS101] Pointer

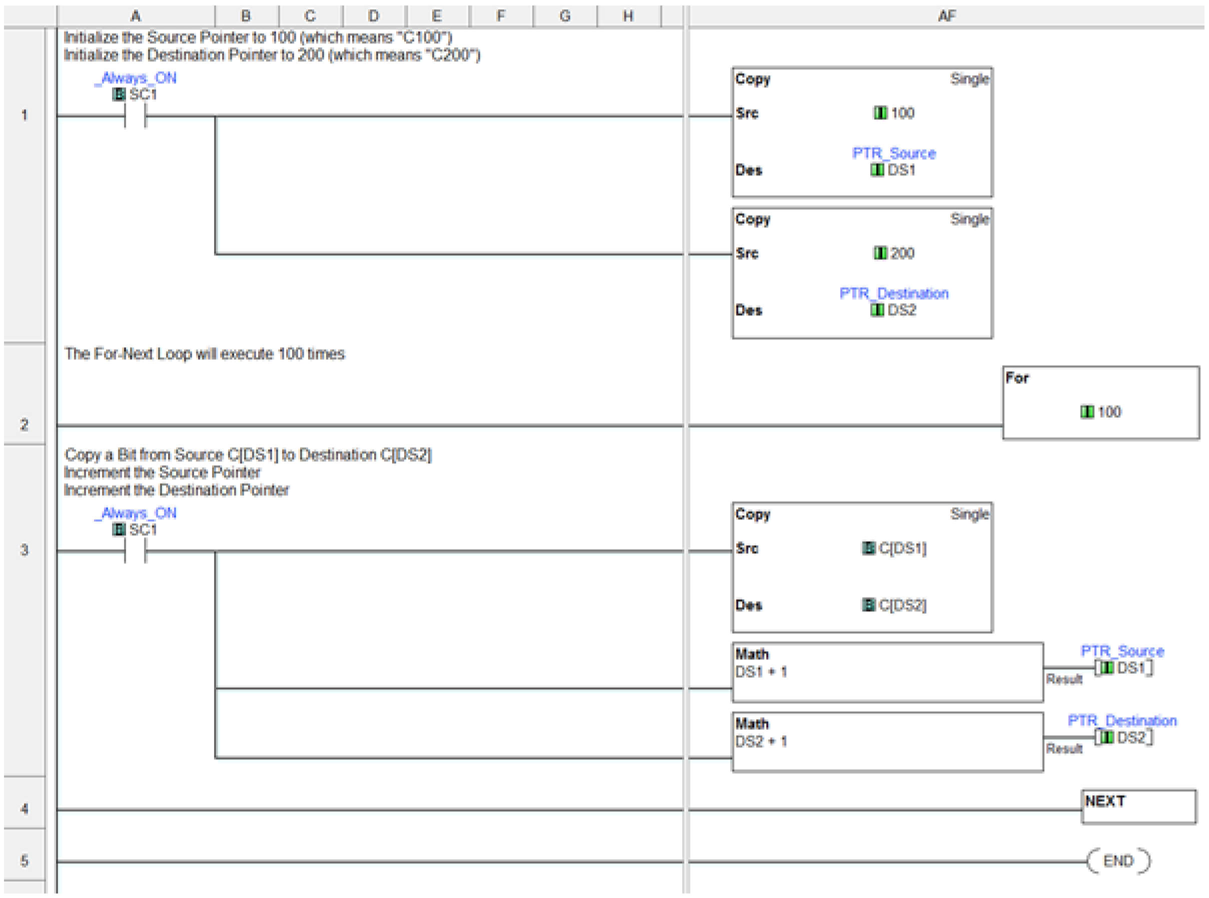
Option
- Suppress zero
- Do not Suppress zero

About Error Flags
SC43 : Out of Range
SC44 : Address Error

☐ One Shot (Execute one time)

Pointer Addressing Example 2

Using Pointer Addressing with a For-Next Loop to move a block of memory. Although Pointer Addressing is only available in the Single Copy mode, the For-Next Loop can be applied for more flexibility. This example will move 100 Bits from Source C100-C199 into Destination C200-C299.



Initialize the Source Pointer to 100 (which means "C100")
Initialize the Destination Pointer to 200 (which means "C200")

The For-Next Loop will execute 100 times

Copy a Bit from Source C[DS1] to Destination C[DS2]
Increment the Source Pointer
Increment the Destination Pointer

DS1 + 1

DS2 + 1

Related Topics:

Single Copy

Block Copy

Pack Copy

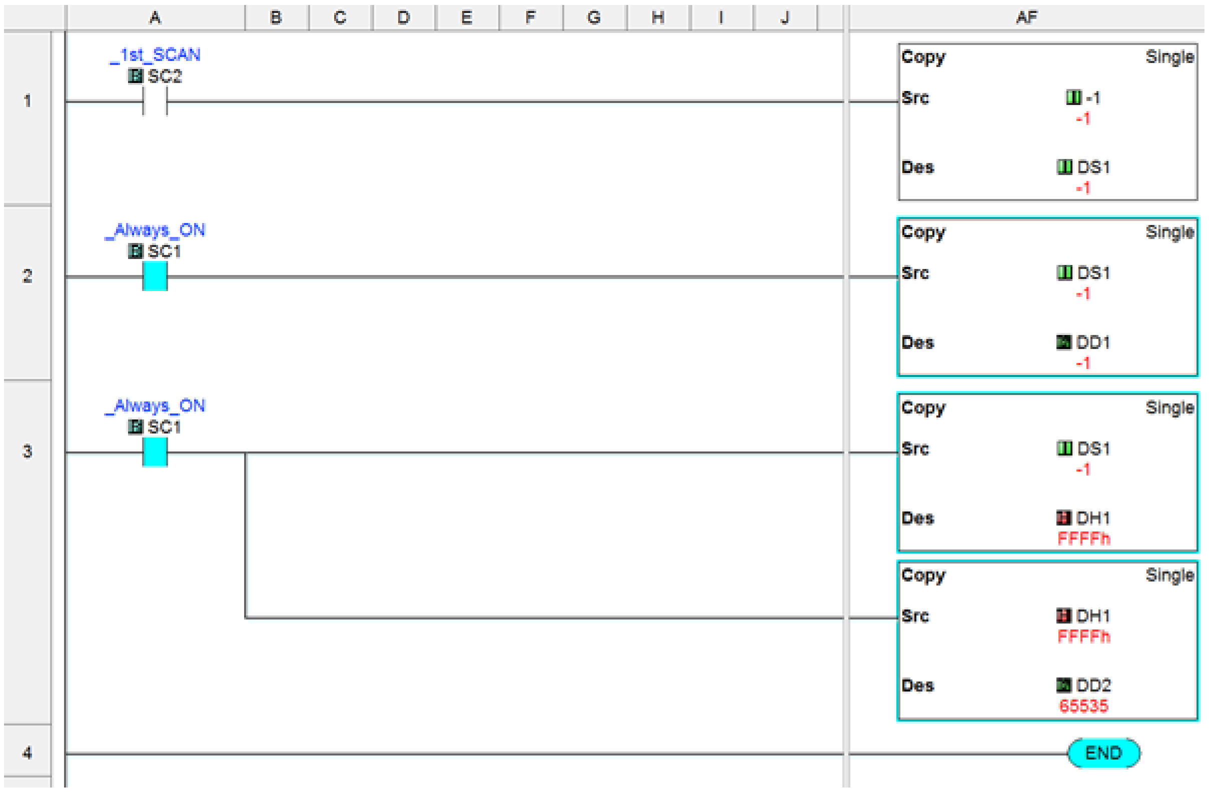
Unpack Copy

Casting Between Datatypes

All CLICK registers are stored as Signed values except for DH. For Single Word Integers this allows a range of -32,768 to 32,767. For Double Word Integers the range is -2,147,483,648 to 2,147,483,647. Negative values are stored as Two's-Complement. A value of -1 is stored in a DS register as 0xFFFF. The same -1 value stored in a DD registers is 0xFFFFFFFF.

Example 1: Bypassing the Sign Priority

The Copy Instruction preserves the sign when moving data between different datatype sizes. The following example shows how to bypass this sign priority. This is useful when a signed register actually contains an unsigned value.



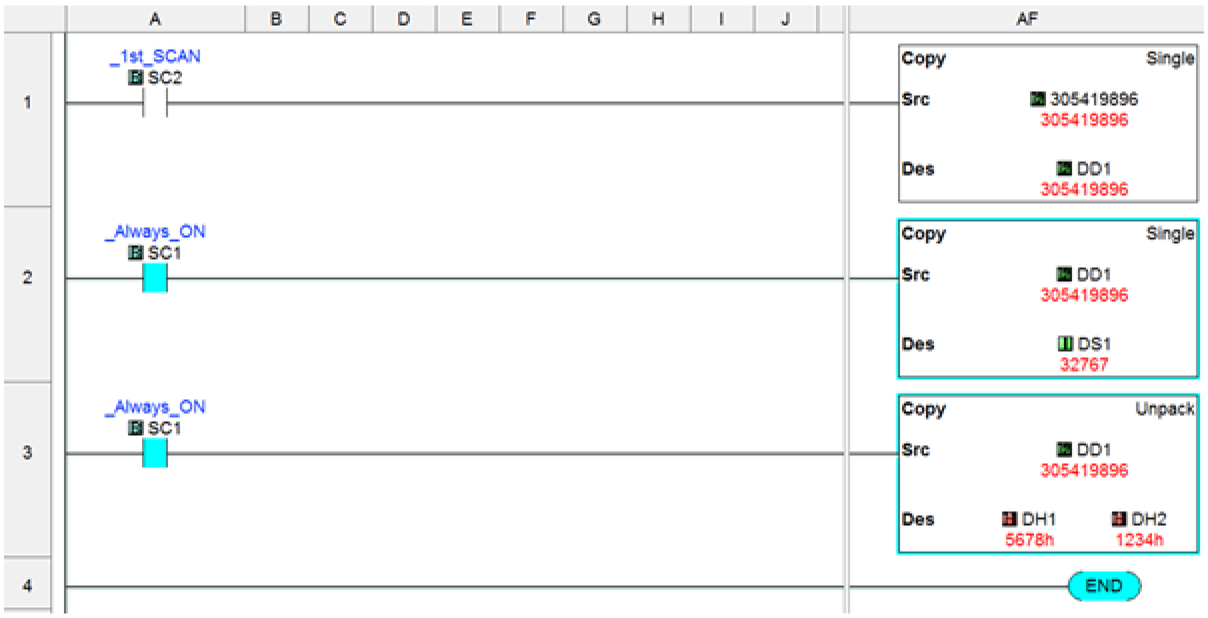
Rung1: Initialize DS1 to a value of -1.

Rung2: Move the signed DS1 register to the signed DD1 register. The sign is preserved and the decimal value of the 32-bit register DD1 is the same as the 16-bit register DS1. The result is -1 (0xFFFFFFFF).

Rung3: Move the signed DS1 register (which contains an unsigned value) to the signed DD1 register. First, Copy from DS to DH, the binary pattern is copied. Second, Copy from DH to DD, the binary pattern is copied. The result is 65535 (0x0000FFFF). This method works because the DH registers are unsigned.

Example 2: Preserving values when moving between 32-bit and 16 bit registers

When moving values from a 32-bit to a 16-bit register care must be taken to avoid going Out of Range of the Destination datatype. The Copy instruction will range limit the value to the datatype of the Destination. Trying to copy a value of 1,000,000 into a DS will result in 32,767. If the binary data pattern needs to be preserved the 32-bit register can be Unpacked into two 16-bit registers.



Rung1: Initialize DD to a value of 305,419,896 (0x12345678).

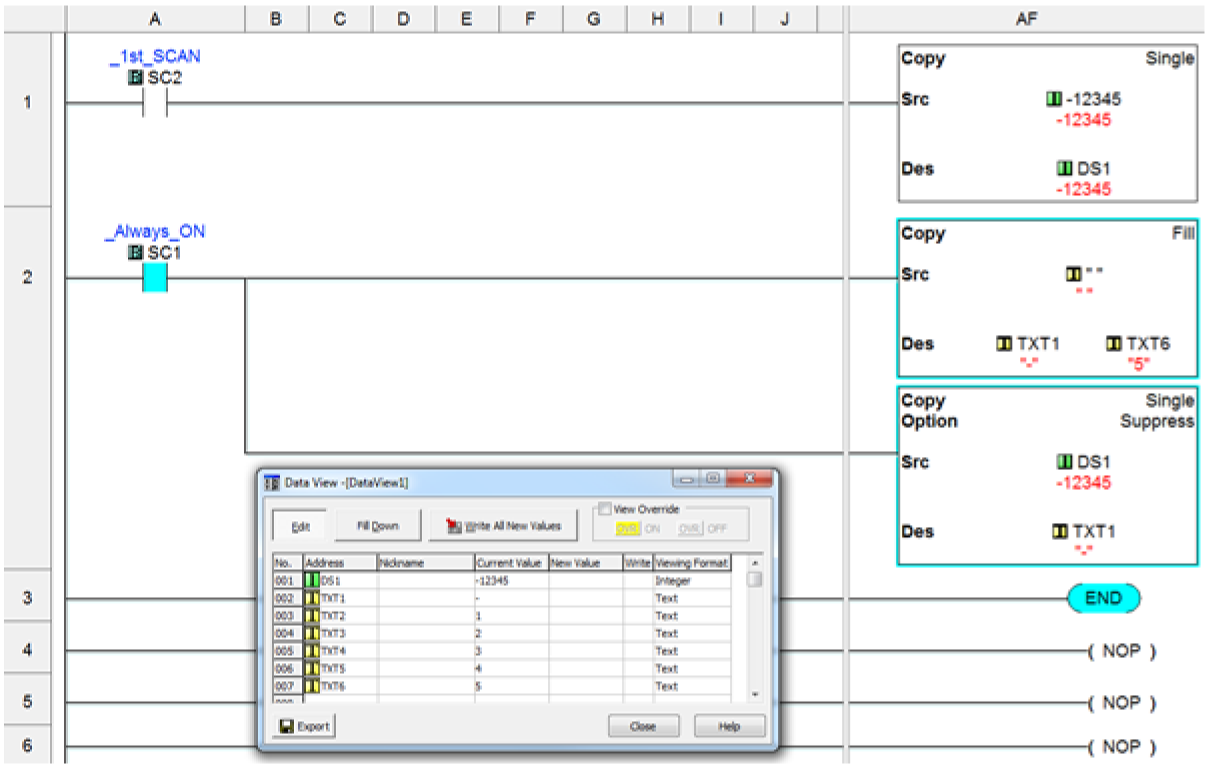
Rung2: Move the signed DD1 register to the signed DS1 register. The Source value is too large for the Destination register and becomes limited to the maximum range 32,767 (0x7FFF).

Rung3: Unpack signed DD1 register to DH registers. The low word (0x5678) is copied into DH1, the high word (0x1234) is copied into DH2.

Example 3: Converting a Value to Text

Number Values can be converted into Strings using the Copy instruction. The Source will be a numeric register; the destination is a Text register. It may be useful to clear the destination registers (Fill with space characters " ") before performing the conversion. The maximum output length depends on the Source datatype;

DH, 4 Characters DS, 6 Characters DD, 11 Characters DF, 14 Characters



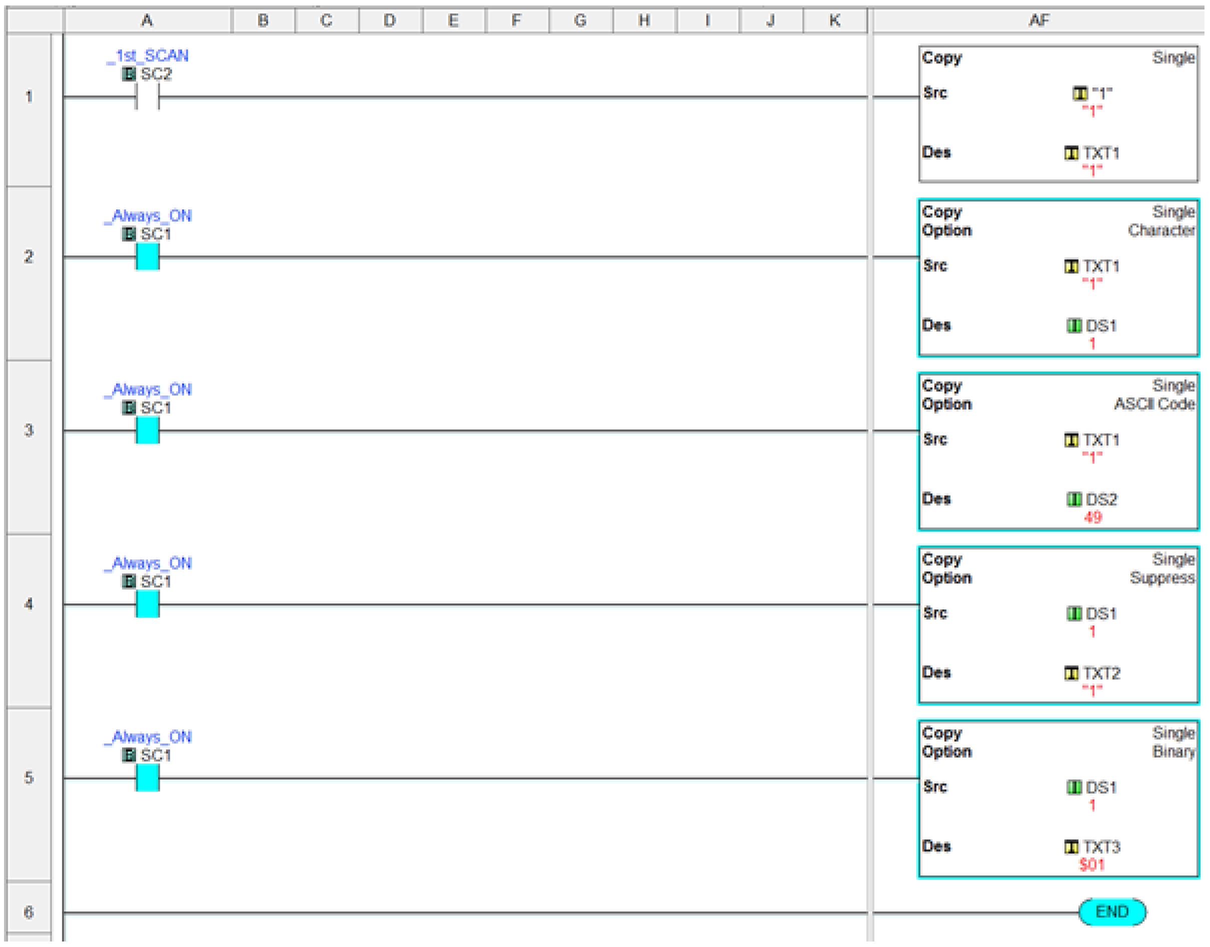
No.	Address	Nickname	Current Value	New Value	Write	Viewing Format
001	DS1		-12345			Integer
002	TXT1		-			Text
003	TXT2		1			Text
004	TXT3		2			Text
005	TXT4		3			Text
006	TXT5		4			Text
007	TXT6		5			Text

Rung1: Initialize DS1 to a value of -12345.

Rung2: Clear the Destination TXT area. Copy the value from DS1 and convert to ASCII character string. As few as 1, and as many as 6 Destination registers will be updated depending on the number of digits in the Source (including minus sign for negative values).

Example 4: Converting Text to Numbers

Converting a numeric register to Text is fairly straightforward. But converting from Text to a numeric register can cause some confusion. The following example explores the various methods.



Rung1: Initialize TXT to the character "1".

Rung2: Copy from a TXT to DS "Copy Character Value". When the Source characters are numeric (0-9) then they will be translated into the proper numeric value to the Destination.

Rung3: Copy from a TXT to DS "Copy ASCII Code Value". Copy the ASCII Code of the Source Character into the Destination.

Rung4: Copy from a DS to TXT "Copy Value". Copy the value of the Source into as many Destination addresses as needed for the Source datatype.

Rung5: Copy from a DS to TXT "Copy Binary". Copy the binary pattern of the Source into the Destination.

Example 5: More examples of casting strings into numeric registers.

TXT to DF:

TXT1-TXT6 = "-12.34", DF=-12.34

TXT1-TXT6 = "+12.34", DF=12.34

TXT1-TXT6 = “ 12.34”, SC43 Out of Range = True, Leading Space not allowed.

TXT1-TXT6 = “12.34 ”, SC43 Out of Range = True, Trailing Space not allowed.

TXT1-TXT6 = “12,345”, SC43 Out of Range = True, Thousands separator not allowed.

TXT1-TXT6 = “-01234”, DF=-1234

TXT1-TXT6 = “1234e2”, DF=123400

TXT1-TXT6 = “1234E2”, DF=123400

TXT1-TXT6 = “123e-2”, DF=1.23

TXT1-TXT10 = “-01234e-02”, DF=-12.34

TXT to DS, DD, TD, or CTD:

TXT1-TXT6 = “123456”, DD=123456

TXT1-TXT6 = “-12345”, DD=-12345

TXT1-TXT6 = “+12345”, DD=12345

TXT1-TXT6 = “000123”, DD=123

TXT1-TXT6 = “ 12345”, SC43 Out of Range = True, Leading Space not allowed.

TXT1-TXT6 = “12345 ”, SC43 Out of Range = True, Trailing Space not allowed.

TXT1-TXT6 = “12,345”, SC43 Out of Range = True, Comma separator not allowed.

TXT1-TXT6 = “12.345”, SC43 Out of Range = True, Period separator not allowed.

TXT1-TXT6 = “1234e2”, SC43 Out of Range = True, Exponential notation not allowed.

TXT to DH:

TXT1-TXT4 = “1234”, DH=0x1234

TXT1-TXT4 = “abcd”, DH=0xABCD

TXT1-TXT4 = “ABCD”, DH=0xABCD

TXT1-TXT4 = “ ABC”, SC43 Out of Range = True, Leading Space not allowed.

TXT1-TXT4 = “ABC ”, SC43 Out of Range = True, Trailing Space not allowed.

TXT1-TXT6 = “A,BC”, SC43 Out of Range = True, Comma separator not allowed.

TXT1-TXT6 = “A.BC”, SC43 Out of Range = True, Period separator not allowed.

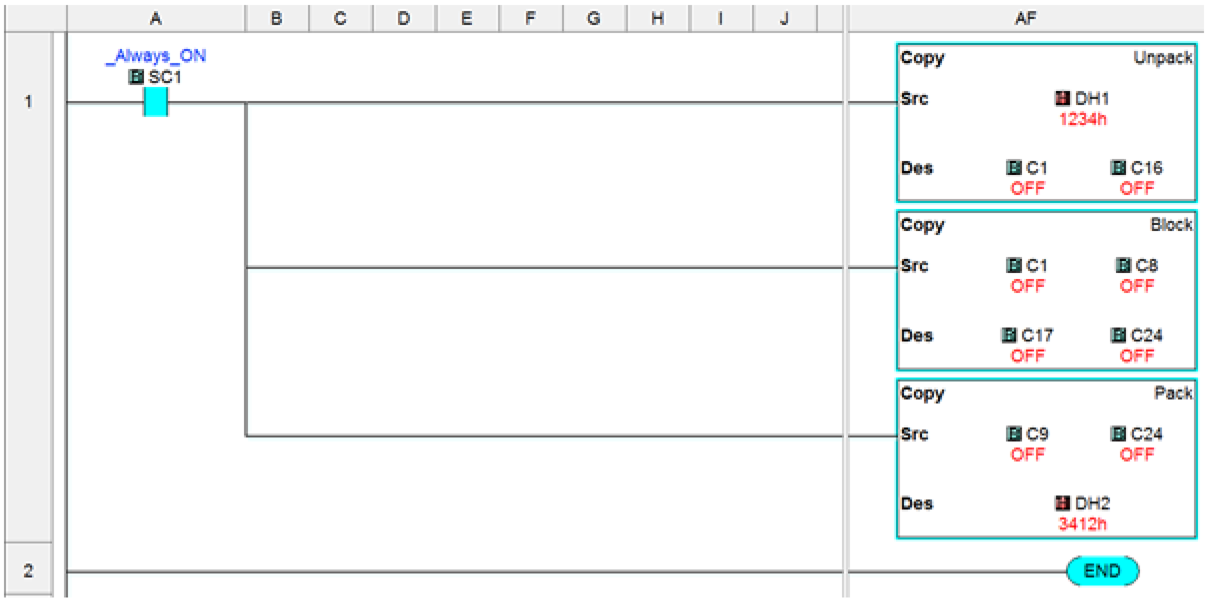
Related Topics:

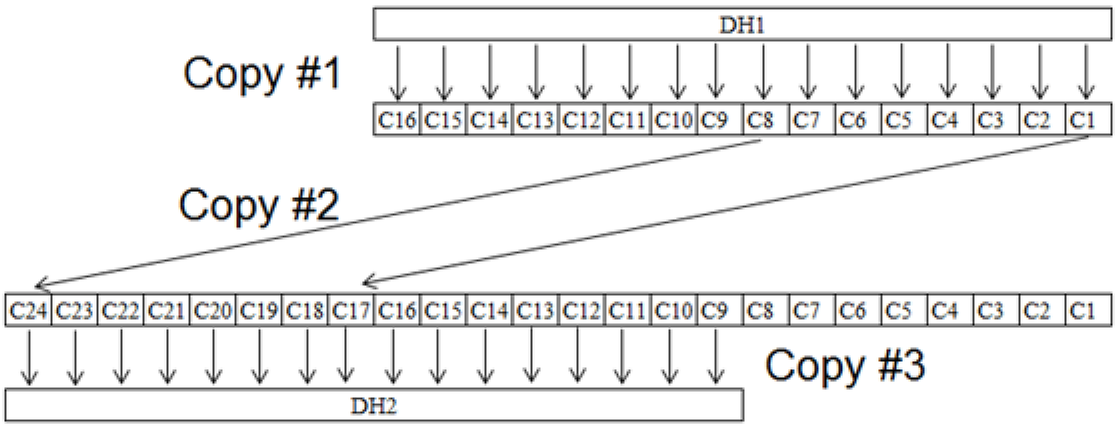
[Single Copy]
[Block Copy]
[Pack Copy]
[Unpack Copy]

Byte Swap and Word Swap

Byte Swap Example

A Byte Swap operation can be created by combining instructions to move the data as needed. The following example uses DH registers, but the same idea can apply to all 16-bit registers: DS, DH and SD.

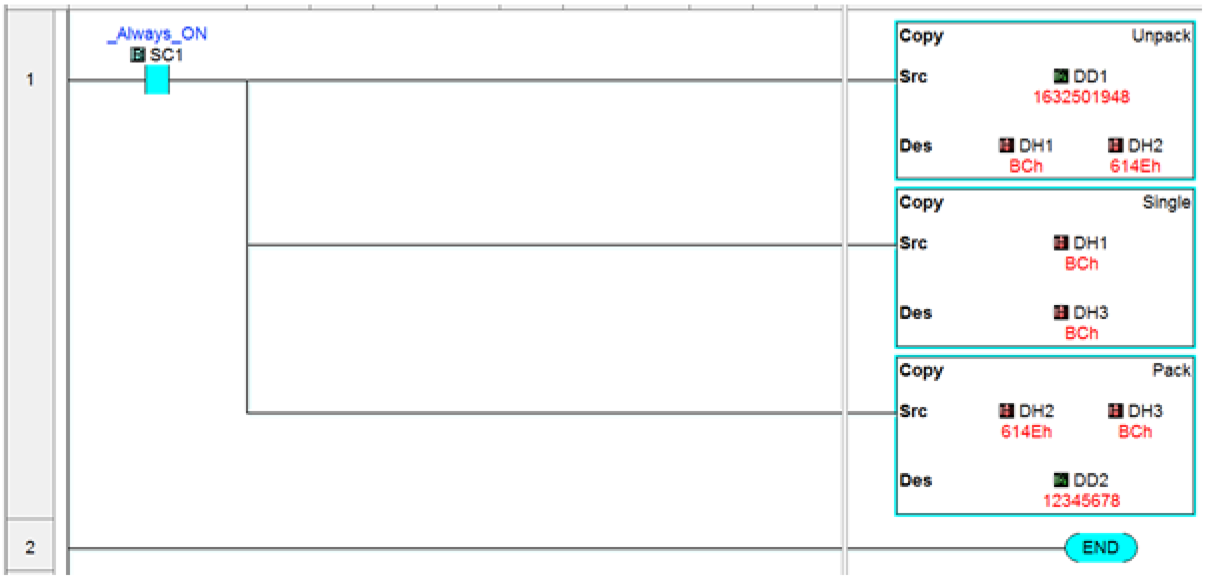


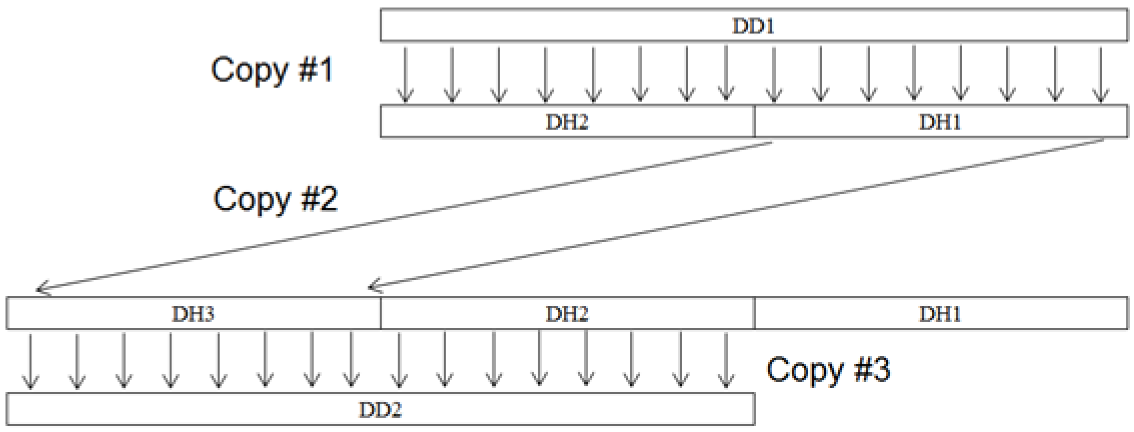

DH1
Copy #1
C16 C15 C14 C13 C12 C11 C10 C9 C8 C7 C6 C5 C4 C3 C2 C1
Copy #2
C24 C23 C22 C21 C20 C19 C18 C17 C16 C15 C14 C13 C12 C11 C10 C9 C8 C7 C6 C5 C4 C3 C2 C1
Copy #3
DH2

1. Copy #1 – Copy Unpack from the 16-bit register into 16 C-bits.
2. Copy #2 - Copy Single the Low Byte above the High Byte.
3. Copy #3 – Copy Pack 16 C-bits into a single 16-bit Register.

Word Swap Example

A Word Swap operation can be created by combining instructions to move the data as needed. The following example uses DD registers, but the same idea can apply to all 32-bit registers: DD, DF and CTD.





1. Copy #1 – Copy Unpack 32-bits into two 16-bit registers.
2. Copy #2 –Copy Single the Low Word above the High Word.
3. Copy #3 – Copy Pack two 16-bit registers into a single 32-bit register.

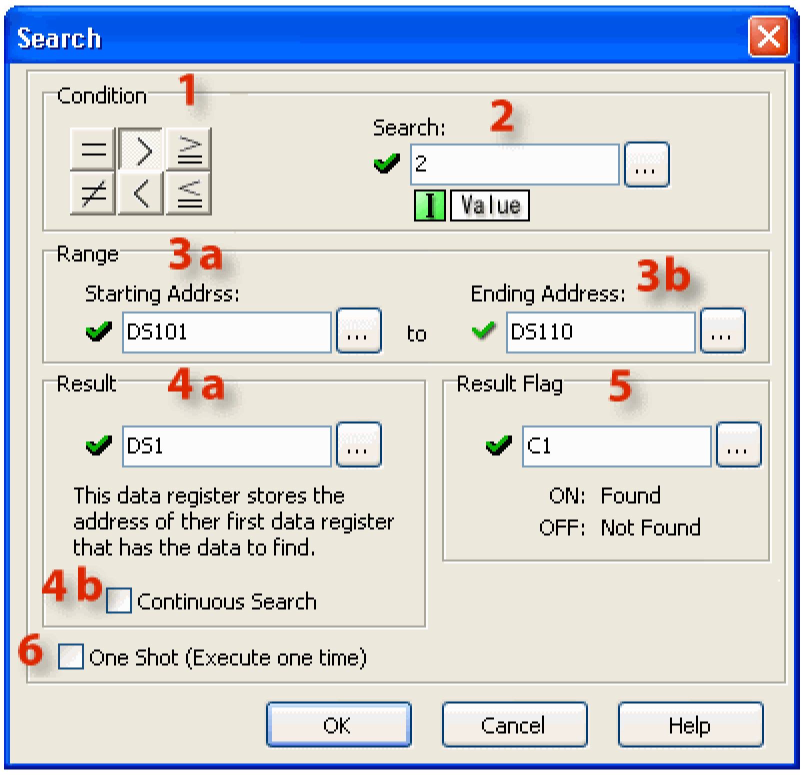
Related Topics:

Single Copy
Block Copy
Pack Copy
Unpack Copy

Search Instruction

Definition

The **Search** instruction is used to search for a data value that meets the specified condition and that is located within a specified range of data registers. A successful search returns the **Memory Address** of the value that satisfies the condition.



✓ **Entry required or invalid entry** ✓ **Valid entry**

Setup

1 Condition: Establish the condition you want the searched item to satisfy.

2Search: Set the value to accompany the **Condition**. This value can be a constant or a **Data Register**.

3a Starting Address: Set the **Starting Address** of the range you want to search within.

3b Ending Address: Set the **Ending Address** of the range you want to search within.

4aResult: Assign a **Data Register** to receive the **Search Result**. If the Search was not successful, "**-1** (minus one)" is put in the **Data Register**.

4bContinuous Search:

Continuous Search = OFF

The **Search** always starts from the **Starting Address**. If there is no data that matches the condition, "**-1**" will be put in **Memory Address** assigned to store the **Search** result.

Continuous Search = ON

The **Search** is continued from the next address after a successful **Search**. Once all instances that satisfy the **Search** criteria are found, "**-1**" will be put in memory address assigned to store the **Search** result. **Put Zero in the memory address storing the Search result to re-start the Search from the Starting Address of the Search Range.**

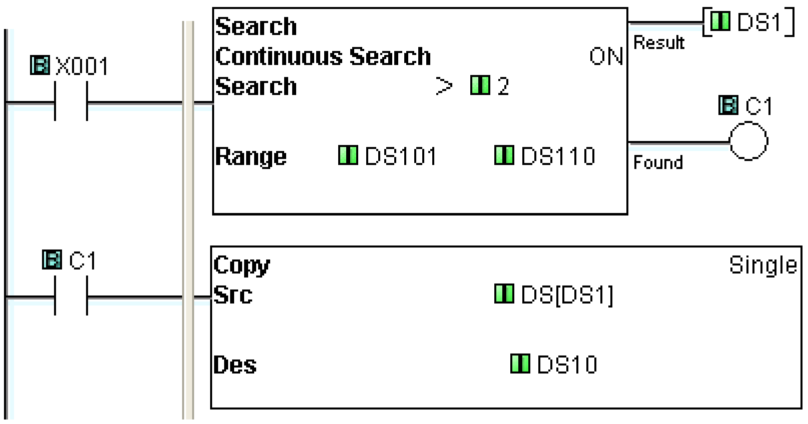
5 Result Flag: The **Result Flag** notifies the user that the **Search** criteria were met.

6One Shot: Checking **One Shot** causes the instruction to execute one time when the enabling rung transitions from **OFF** to **ON**.

Example Program

Example Program 1: Search and Copy Result

In the following example, when **X001** is **ON**, the memory range from **DS101** to **DS110** is searched for a value greater than **2**. On the first scan, the search begins at **DS101** and continues until a value greater than **2** is found or until the range of **Memory Addresses** is exhausted. If the search locates a value greater than **2**, the search is stopped until the next scan and the **Memory Address** of the qualifying value (first value greater than 2) is written to **DS1**. **Flag** bit, **C1**, turns **ON** to indicate a successful search and turns **ON** the **Copy** instruction in the next rung. Using **Pointer Addressing** the value contained in the **Memory AddressDS1** is Copied to **DS10**.

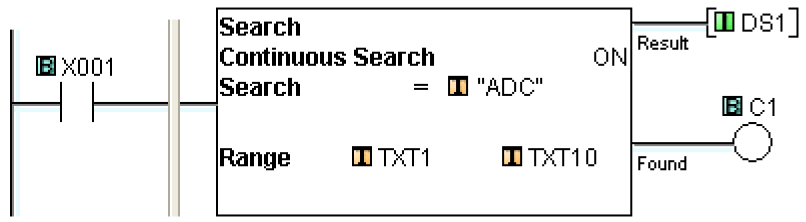


Scan Sequence

Data Values	Scan	DS1 =	C1 =
DS101 = 1	n	102	ON
DS102 = 3	n + 1	105	ON
DS103 = 1	n + 2	-1	OFF
DS104 = 1			
DS105 = 4			
DS106 = 1			
DS107 = 1			
DS108 = 1			
DS109 = 1			
DS110 = 1			

Example Program 2: Search for Text

In this example, when **X001** is **ON**, the memory range from **TXT1** to **TXT10** is searched for a text value equal to "**ADC**." Each letter is contained in a single 8-bit register, so the search is looking for a text string comprised of three consecutive registers which contain the letters: **A**, **D**, and **C**. A successful search returns the beginning memory address (the location of "A") to **DS1** and turns **ON** the **Flag** bit, **C1**.



__Scan Sequence__

Data Values	Scan	DS1 =	C1 =
TXT1 = A	n	3	ON
TXT2 = D			
TXT3 = A			
TXT4 = D			
TXT5 = C			
TXT6 = A			
TXT7 = D			
TXT8 = A			
TXT9 = D			
TXT10 = A			

__**Related Topics:**__

__Single Copy__

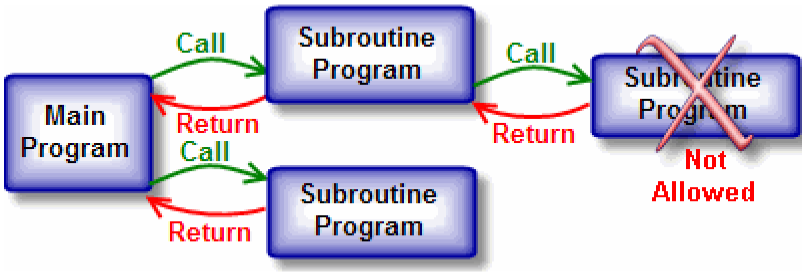
Call Instruction

Definition

The **Call** instruction is required to call a **SubroutineProgram** from the **Main Program**. The **Call** instruction resides in the **Coil Area** of the **Main Program**. A **Subroutine** program must have a Return instruction to return to the **Main Program**.

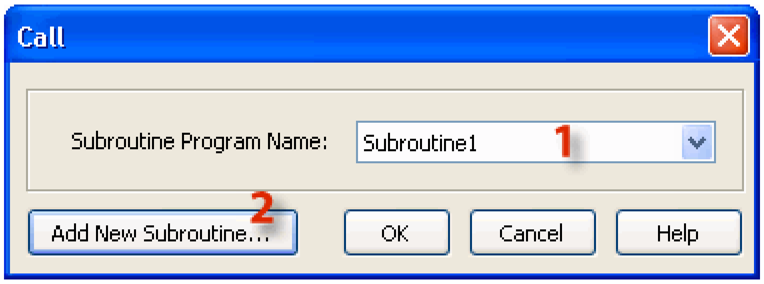
Note: You can have up to **986 Subroutine Programs** in a **CLICK** project.

The nesting level of the **Subroutine** is **1**. This means a **Subroutine Program** cannot **Call** another **Subroutine Program**. It always needs to return to the **Main Program**.



Setup

Position the **Box Cursor** on the **Coil Area** of the rung you have selected. This will be the enabling rung for the **Call** instruction. Select **Call** from the Instruction Menu > Program Control or the **Instruction List**. The **Call** instruction is only visible on the **Instruction List** when viewing the **Main Program**.



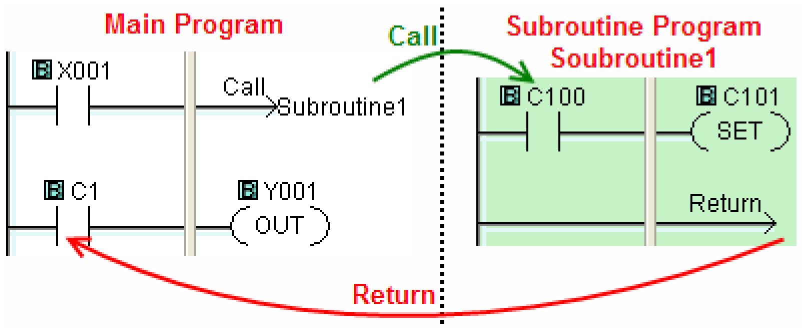
1Subroutine Program Name: Select the desired **Subroutine** program from the drop-down list, and click **OK**.

2 Add New Subroutine: If you have not already created a **Subroutine** program, you can do so by clicking the **Add New Subroutine** button. The Add New Subroutine dialog will appear, and you can create a **Subroutine Program**.

Example Program

Example Program: Call and Return

In this **Example Program**, when **X001** is **ON**, the **Call** instruction causes the **Subroutine Program** to be processed immediately. The **Subroutine Program** must contain at least a **Return** instruction. Once a **Return** instruction is executed, the **CLICK CPU** continues executing the **Main Program** after the **Call** instruction.



Related Topics:

[Add New Subroutine]()
[Quick Guide Subroutine]()
[Return]()

Return Instruction

Definition

The **Return** instruction is used to return to the **Main Program** from a **Subroutine Program** or **Interrupt Program**. The **Return** instruction can be **Conditional** or **Unconditional**. **Conditional** means the rung including a **Return** instruction has **Contacts**. **Unconditional means that there is no Contact on the rung including the Return instruction**.

Note: The **Return** instruction is available from the **Instruction Menu** or the **Instruction List**, but it is only visible on the **Instruction List** when viewing a **Subroutine Program** or **Interrupt program**.

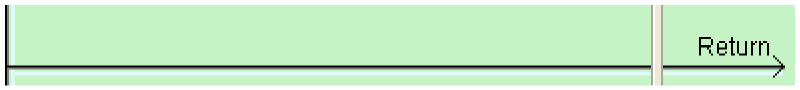
Setup

Position the **Box Cursor** on the **Coil Area** of the rung you have selected. This will be the enabling rung for the **Return** instruction. Select **Return** from the [Instruction Menu > Program Control]() or the **Instruction List**. The **Return** instruction is only visible on the **Instruction List** when viewing the **Main Program**.

Return Instruction Examples

Example Program 1: Unconditional Return

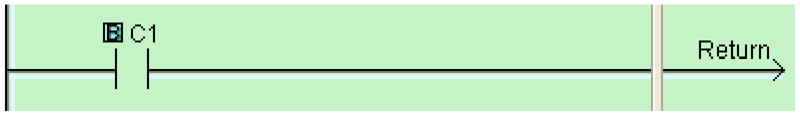
When this rung is executed, the **Program** execution goes back to the **Main Program**.



Caution: Any **Subroutine Program** and **Interrupt Program** must have an **Unconditional Return**.

Example Program 2: Conditional Return

When **C1** is **ON**, the **Program** execution goes back to the **Main Program**.



Related Topics:

Call Instruction
Quick Guide Subroutine
Quick Guide Interrupt

T

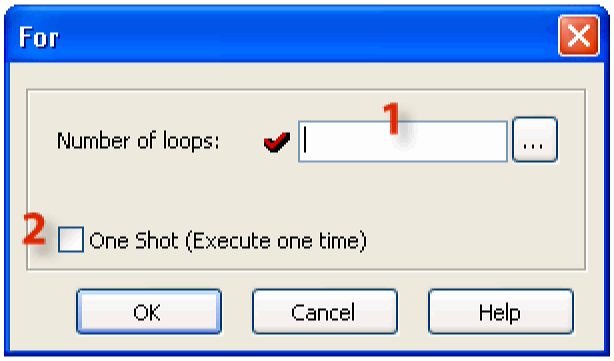
For Instruction

Definition

The **For** instruction indicates the starting point of a **For - Next loop**, and based on its user setting, determines how many times the **For - Next loop** will be executed in one program scan. Between the **For** instruction rung and the **Next** instruction rung, place the rungs of logic that should be repeated multiple times.

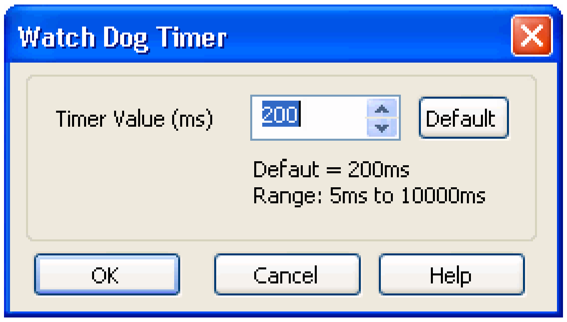
Caution: Placing one **For-Next loop** inside another **For-Next loop** is NOT permitted.

Setup



1 Number of loops: The **Number of loops** can be a constant typed directly or it can be any eligible **Data Memory Address**.

Note: If the **Number of Loops** setup is large, it will increase the program scan time. When the program scan time exceeds the **Watchdog Timer** setup (Default = **200ms**), the **CLICK PLC** drops from the **Run** mode. You may increase the **Watch Dog Timer** setup.



Timer Value (ms) 200

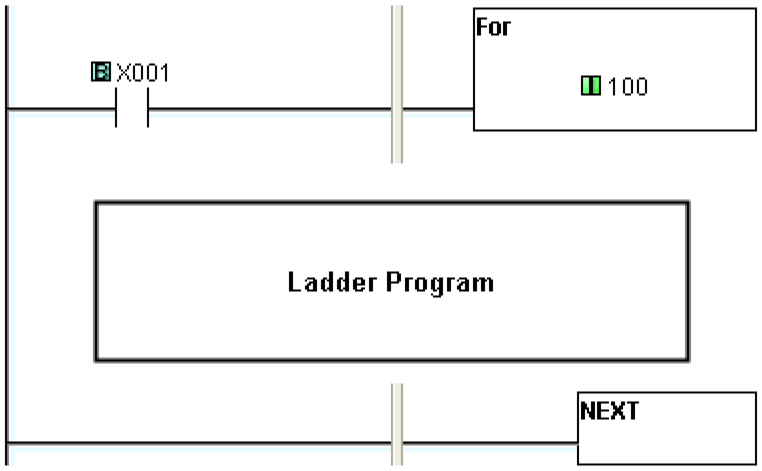
Defaut = 200ms
Range: 5ms to 10000ms

2 One Shot: Select the **One Shot** by clicking the check box. If **One Shot** is selected, the **For-Next Loop** will be processed (all loops) one time after being enabled. The **For-Next Loop** will not be executed again until there is another **OFF-to-ON** transition by the enabling contact.

Example Program

Example Program: For-Next Loop

In the following example, when **X001** is **ON**, the **Ladder Program** is executed **100** times in one same program scan.



Related Topics:

[Next Instruction]()

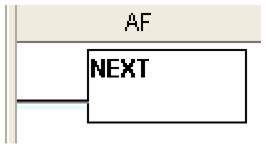
Next Instruction

Definition

The **Next** instruction indicates the end of a **For Next** loop. The **Next** instruction must be the only instruction on its rung. No enabling contacts are allowed on the same rung with the **Next** instruction.

Setup

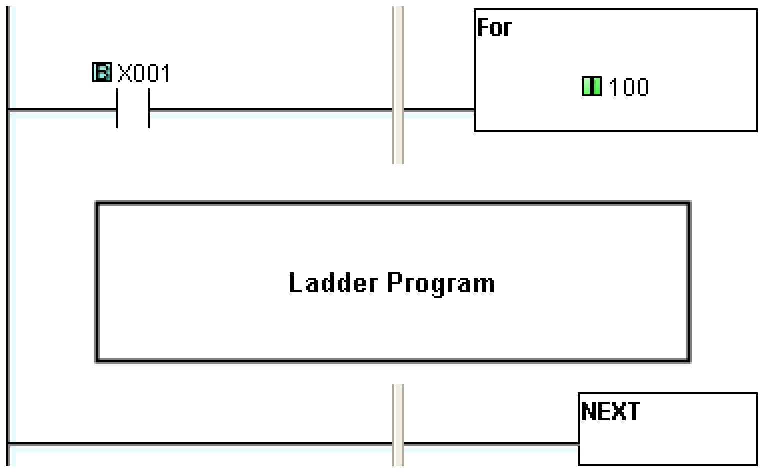
When the user selects the **Next** instruction from the **Instruction List** and drops it into the **Ladder Editor** at the specified position, the symbol below appears in the **Ladder Editor** at the specified position.



Example Program

Example Program: For-Next Loop

In the following example, when **X001** is **ON**, the **Ladder Program** is executed **100** times.



The **Next** instruction does not require a dialog for setup. The placement of the **Next** instruction in the appropriate **Coil Area** location is achieved by one of several simple methods:

1. **D**rag and Drop the instruction from the**Instruction List**
2. Placing the**Box Cursor**in the rung position where you want the**Next** instruction to appear and:
    1. Type the letter**"n"**or **"N"**
    2. Select[Instruction > Program Control > Next]()

Related Topics:

[For Instruction]()

End Instruction

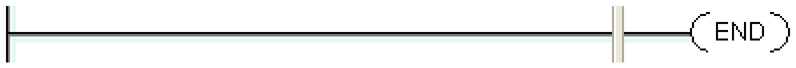
Definition

The **End** instruction marks the termination point of the normal program scan.
The **Main** program must have an **End** instruction to tell the **CPU** that there are no more rungs to be processed. The **End** instruction cannot reside in a **Subroutine** or **Interrupt** program.

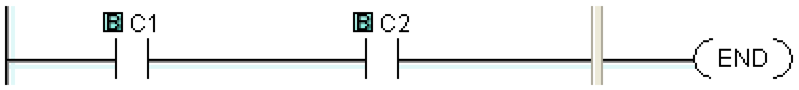
Specifications

The **End** instruction can be **Unconditional** or **Conditional**.

1. **Unconditional End:Unconditional End** means that there is no **Contact** on the rung including the **End** instruction. So the **End** instruction is always executed.



1. **Conditional End: Conditional End** means the rung including an **End** instruction has **Contacts**. These **Contacts** decide when the **End** instruction is executed.



Caution: The **Main Program** needs to have at least one **Unconditional End** instruction. There is **No Limit** on the number of **Conditional** or **Unconditional End** instructions.

The **End** instruction does not require a dialog for setup. The placement of the **End** instruction in the appropriate **Coil Area** location is achieved by one of several simple methods:

1. Drag and Dropthe instruction from the**Instruction List**
2. Placing the**Box Cursor** in the rung position where you want the **End Instruction** to appear and:
    1. Type the letter **"e"** or **"E"**
    2. Select**Instruction > Program Control > End**

Send and Receive ASCII Instructions

Description

To send and receive ASCII Instructions to and from external devices you must have at least one scan between the Success and the Enable signals. If less that one scan between signals data may be lost. Please refer to **Timing Chart** below.

Note: Port 1 on the **CLICK Basic** and **Analog** CPUs supports only the programming software and operator interfaces/**HMI's**. It does not support the **Receive** and **Send** instructions in a running program. **Port 1** on the **CLICK Ethernet** and **CLICK PLUS** CPUs is an Ethernet port.

Note: For the **Com Port** wiring info, please refer to the CLICK PLC Hardware Manual or CLICK PLUS PLC Hardware Manual.

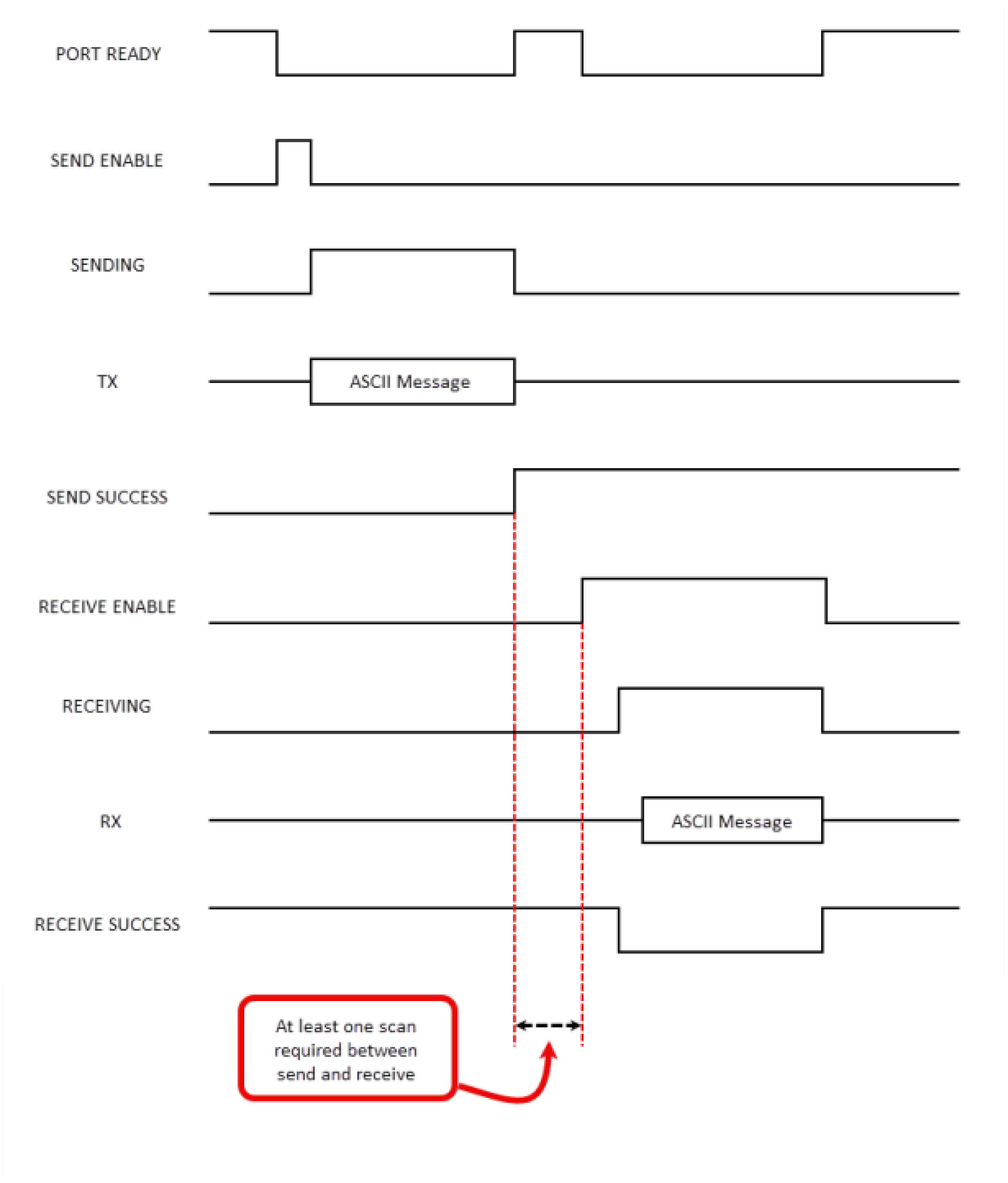
Setup

To set up the Send and Receive instructions see links below:

- [Send Instruction: ASCII]
- [Receive Instruction: ASCII]

Timing Chart

Send and Receive an ASCII Message to an External Device



Related Topics:

Send Instruction: ASCII

Receive Instruction: ASCII

Receive Instruction: ASCII

Description

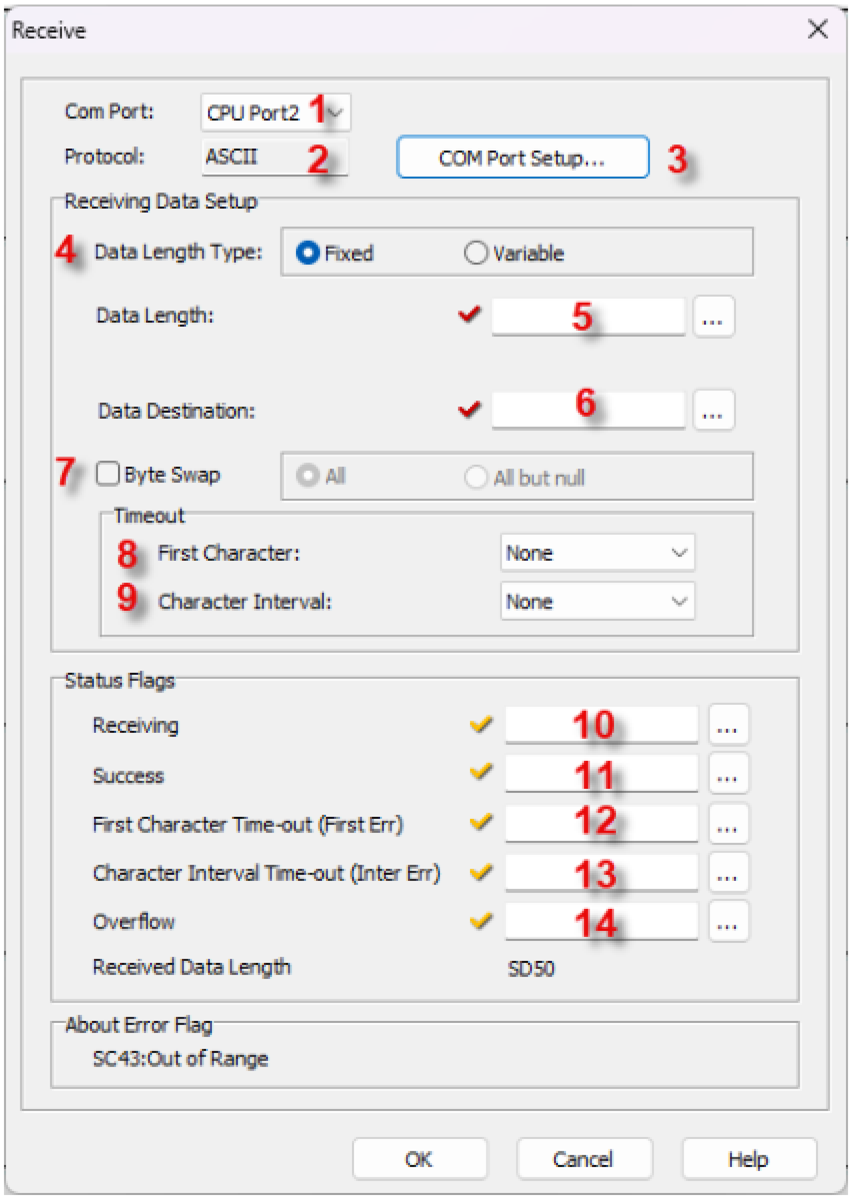
The **Receive** instruction allows you to use **Com Port 2 or 3** (if available) on the **CLICK CPU** modules, or **Com Port 1 or 2** of a **C2-DCM module**, to receive **ASCII** text messages from external devices. The **CLICK CPU** modules support the **MODBUS (RTU)** and **ASCII** protocols. This help topic covers the **ASCII** protocol. Please refer to the help topic Receive Instruction: MODBUS for the **MODBUS** protocol.

Note: Port 1 on the **CLICK Basic** and **Analog** CPUs supports only the programming software and operator interfaces/**HMI's**. It does not support the **Receive** and **Send** instructions in a running program. **Port 1** on the **CLICK Ethernet** and **CLICK PLUS** CPUs is an Ethernet port.

Note: For the **Com Port** wiring info, please refer to the CLICK PLC Hardware Manual or CLICK PLUS PLC Hardware Manual.

The serial ports default to **MODBUS** protocol, but can be quickly changed to **ASCII** protocol. Click the **COM Port Setup** button to open the COM Port Setup dialog. When the dialog opens, select the desired **COM Port** and select **ASCII** in the protocol box.

Setup


Receive

Com Port: CPU Port2 1

Protocol: ASCII 2

Receiving Data Setup

4 Data Length Type: [✓ Fixed] Variable

Data Length: 5

Data Destination: 6

7 Byte Swap — All / All but null

Timeout

8 First Character: None

9 Character Interval: None

Status Flags

Receiving 10

Success 11

First Character Time-out (First Err) 12

Character Interval Time-out (Inter Err) 13

Overflow 14

Received Data Length SD50

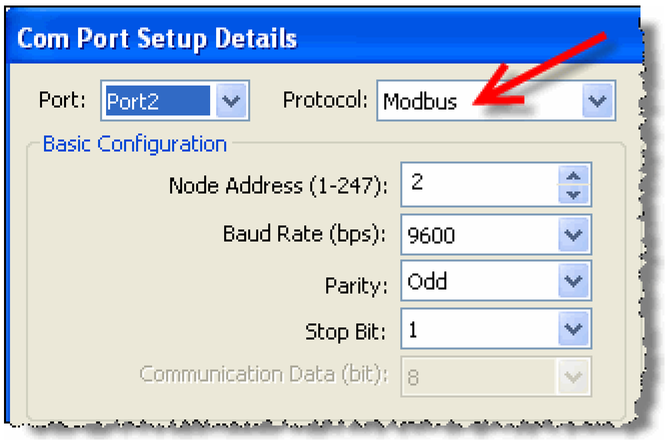
About Error Flag

SC43:Out of Range

1 Port ID: Click the down arrow to select a **Port** from the available list. **Port 2** is default.

2 Protocol Type: This field displays the **Protocol Type** that is currently selected. The two choices are **MODBUS** and **ASCII**. To change from **MODBUS** to **ASCII**, see **Item 3**.

3 COM Port Setup: Click on this button to open the **Com Port Setup** window shown below. From this window click on the down arrow for the **Protocol** field and select **ASCII** or **MODBUS**.



Com Port Setup Details

Port: Port2 | Protocol: Modbus

Basic Configuration

Node Address (1-247): 2

Baud Rate (bps): 9600

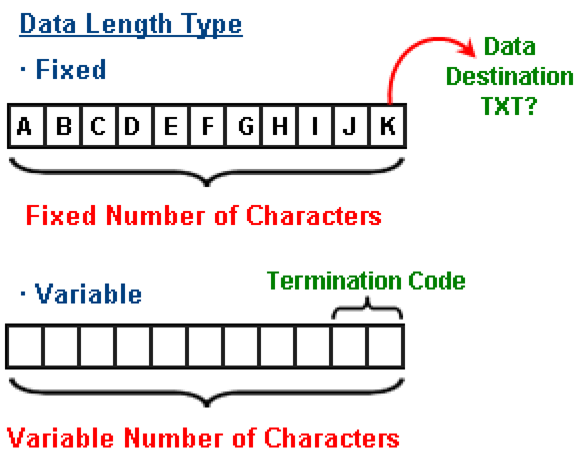
Parity: Odd

Stop Bit: 1

Communication Data (bit): 8

Note: If the **Protocol** parameter is grayed out, it means that there is at least one **Receive** or **Send** instruction assigned to the **Com Port** already. If you still want to change the **Protocol**, please delete those **Receive/Send** instructions first.

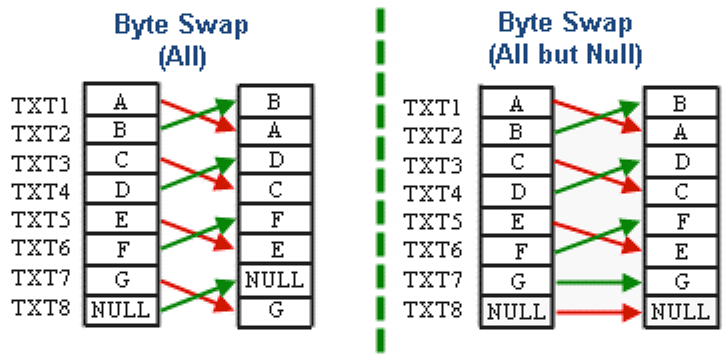
4 Data Length Type: Select **Fixed** or **Variable Data Length Types**. **Fixed** will have equal **Data Length Types** while **Variable** will have a **Variable** number of characters.



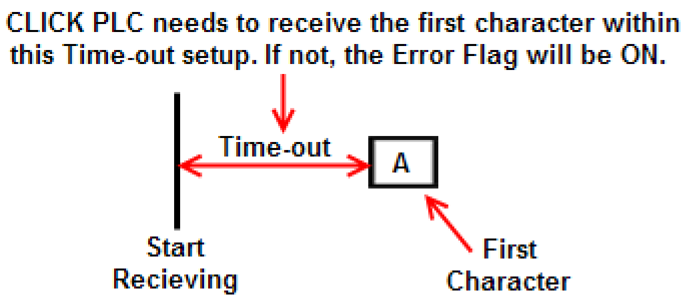
5 Data Length: Enter or select using the **Browse Icon** ⋯, the **Bit** to send the **Data Length**.

6 Data Destination: Enter or select using the **Browse Icon** ⋯, the **Bit** the **Data** will be stored.

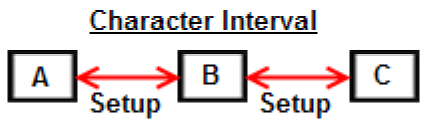
7 Byte Swap: Select the checkbox if **Byte Swap** will be used. If selected, the **ALL** and **All But Null** selections will become active. Click one of these methods to select it.



8 First Character: Click on the down arrow and select a **Time-out** value from the drop down list for the **First Character**.


CLICK PLC needs to receive the first character within this Time-out setup. If not, the Error Flag will be ON.
Time-out
A
Start Recieving
First Character

9 Character Interval: Click on the down arrow and select a **Time-out** value from the drop down list for the **Character Interval**.



10 Receiving: Assign a **CBit** as the **Receiving** flag. This flag turns **ON** when the com port is **Receiving** an **ASCII** message from the external device. Please refer to the **Timing Chart** shown below.

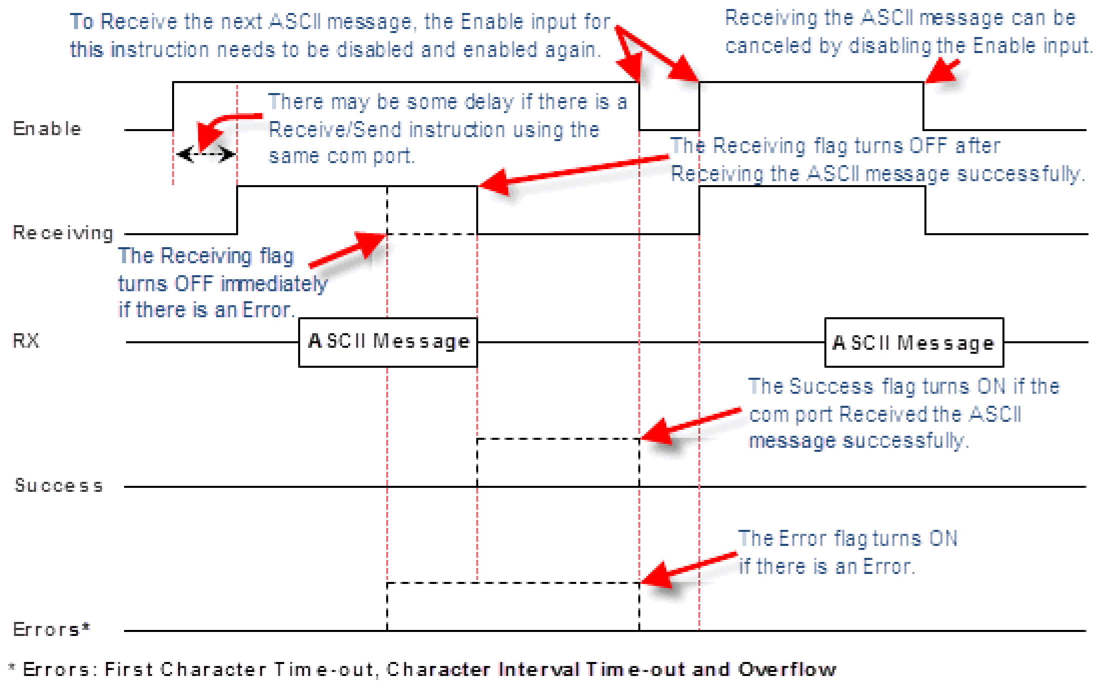
11Success: Assign a **C Bit** as the **Success** flag. This flag turns **ON** when the com port **Received** an **ASCII** message from the external device successfully. This flag turns **OFF** when the **Enable** input for this instruction is disabled. Please refer to the **Timing Chart** shown below.

12 First Character Time-out (First Err): Assign a **CBit** as the **First Character Time-out** flag. This flag turns **ON** if the com port could not **Receive** the **First Character** within the time period defined as the **First Character Interval Timeout** (see parameter #8). This flag turns **OFF** when the **Enable** input for this instruction is disabled. Please refer to the **Timing Chart** shown below.

13 Character Interval Time-out (Inter Err): Assign a **CBit** as the **Character Interval Time-out** flag. This flag turns **ON** if the com port could not **Receive** the next **Character** within the time period defined as the **Character Interval Timeout** (see parameter #9). This flag turns **OFF** when the **Enable** input for this instruction is disabled. Please refer to the **Timing Chart** shown below.

14 Overflow: Assign a **C Bit** as the **Overflow** flag. This flag turns **ON** if the com port **Received** more than 128 characters. This flag works only when the **Data Length Type** is variable (see parameter #4). This flag turns **OFF** when the **Enable** input for this instruction is disabled. Please refer to the **Timing Chart** shown below.

Timing Chart



To Receive the next ASCII message, the Enable input for this instruction needs to be disabled and enabled again.
Receiving the ASCII message can be canceled by disabling the Enable input.
Enable
There may be some delay if there is a Receive/Send instruction using the same com port.
The Receiving flag turns OFF after Receiving the ASCII message successfully.
Receiving
The Receiving flag turns OFF immediately if there is an Error.
RX
ASCII Message
ASCII Message
The Success flag turns ON if the com port Received the ASCII message successfully.
Success
The Error flag turns ON if there is an Error.
Errors*
* Errors: First Character Time-out, Character Interval Time-out and Overflow

<u>Related Topics:</u>

[Send and Receive ASCII Instructions]

[Receive Instruction: MODBUS]

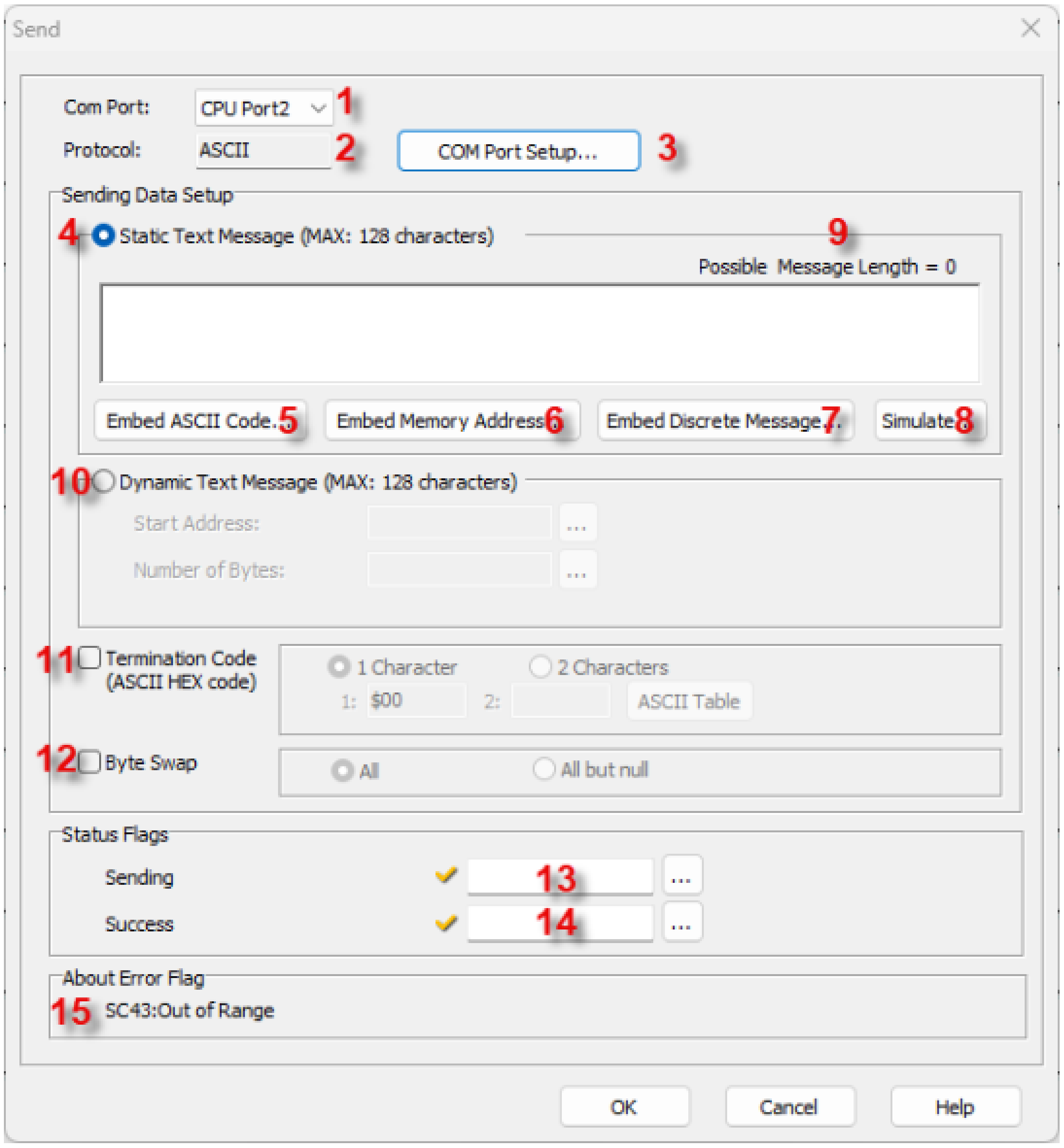
Send Instruction: ASCII

Description

The **Send** instruction allows you to send **ASCII** text messages to external devices such as a serial printer or a text message display. The **CLICK CPU** modules support the **MODBUS (RTU)** and **ASCII** protocols. This help topic covers the **ASCII** protocol. To use the **MODBUS (RTU)** protocol, please refer to the help topic Send Instruction: MODBUS.

Note: For the **Com Port** wiring info, please refer to the CLICK PLC Hardware Manual or CLICK PLUS PLC Hardware Manual.

Setup



Send

Com Port: CPU Port2 1

Protocol: ASCII 2

COM Port Setup... 3

Sending Data Setup

4 [✓ Static Text Message (MAX: 128 characters)] 9

Possible Message Length = 0

Embed ASCII Code... 5 Embed Memory Address 6 Embed Discrete Message 7 Simulate 8

10 Dynamic Text Message (MAX: 128 characters)

Start Address: ...

Number of Bytes: ...

11 Termination Code (ASCII HEX code) — 1 Character / 2 Characters; 1: $00 2: ; ASCII Table

12 Byte Swap — All / All but null

Status Flags

Sending 13 ...

Success 14 ...

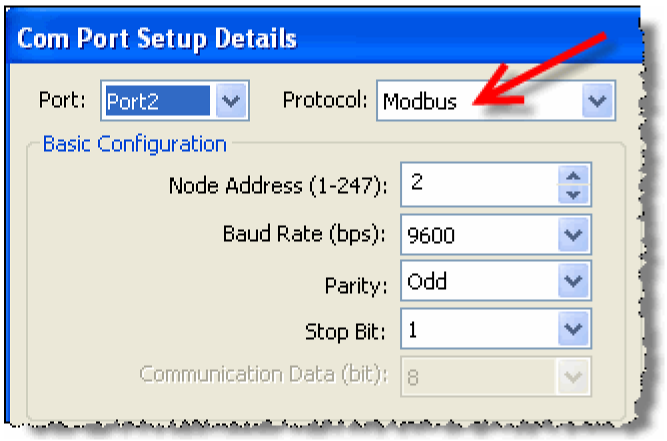
About Error Flag

15 SC43:Out of Range

1 Com Port: Click the down arrow to select a **Port** from the available list.

2 Protocol: This field displays the **Protocol** currently selected. If it is not **ASCII**, see **Item 3** to change the **Protocol Type**.

3 COM Port Setup: Click on this button to open the **Com Port Setup** window shown below. From this window click on the down arrow for the **Protocol** field and select **ASCII** or **MODBUS**.


Com Port Setup Details

Port: Port2 Protocol: Modbus

Basic Configuration

Node Address (1-247): 2

Baud Rate (bps): 9600

Parity: Odd

Stop Bit: 1

Communication Data (bit): 8

Note: If the **Protocol** parameter is grayed out, it means there is at least one **Receive** or **Send** instruction assigned to the **Com Port** already. If you still want to change the **Protocol**, please delete those **Receive/Send** instructions first.

4 Static Text Message (Max: 128 Characters): From this field, you can compose the **ASCII Text Message** to be sent to the external device. You can mix **Text Messages**, **ASCII Codes** and **Memory Addresses** in the **ASCII Text Message**. This field allows up to a **128** character message.

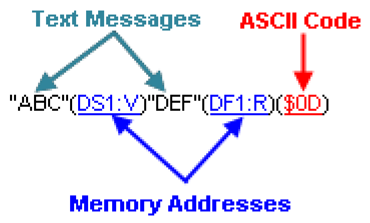
1. **Text Message:** Please surround by double quotations. (e.g., **"ABC"**)

2. **ASCII Code:** Click the **Embedded ASCII Code** button under the field to pick an **ASCII Code**. **ASCII Codes** are displayed in the format of "**$**" **+ 2 digit Hex Code** in this field.

3. **Memory Address:** You can embed the following **Memory Addresses** in the **ASCII Text Message**. What will be included in the **ASCII Text Message** when it is output from the **Com Port** is the data in the **Memory Address**. Click the **Embedded Memory Address** button under the field to enter a data register **Memory Address** and select the format of the data.

<u>**Supported Memory Addresses: DS, DD, DH, DF, XD, YD, TD, CTD and SD**</u>

Example:



5 Embedded ASCII Code: Click this button to open a dialog used to select an **ASCII Code (00h to FFh)**. You can select an **ASCII Code** to embed in the **ASCII Text Message**. Please refer to the help topic **ASCII Table** for details.

6 Embedded Memory Address: Click this button to open a dialog used to enter a **Data Register Memory Address** and select the format. Please refer to the help topic **Embedded Address** for details.

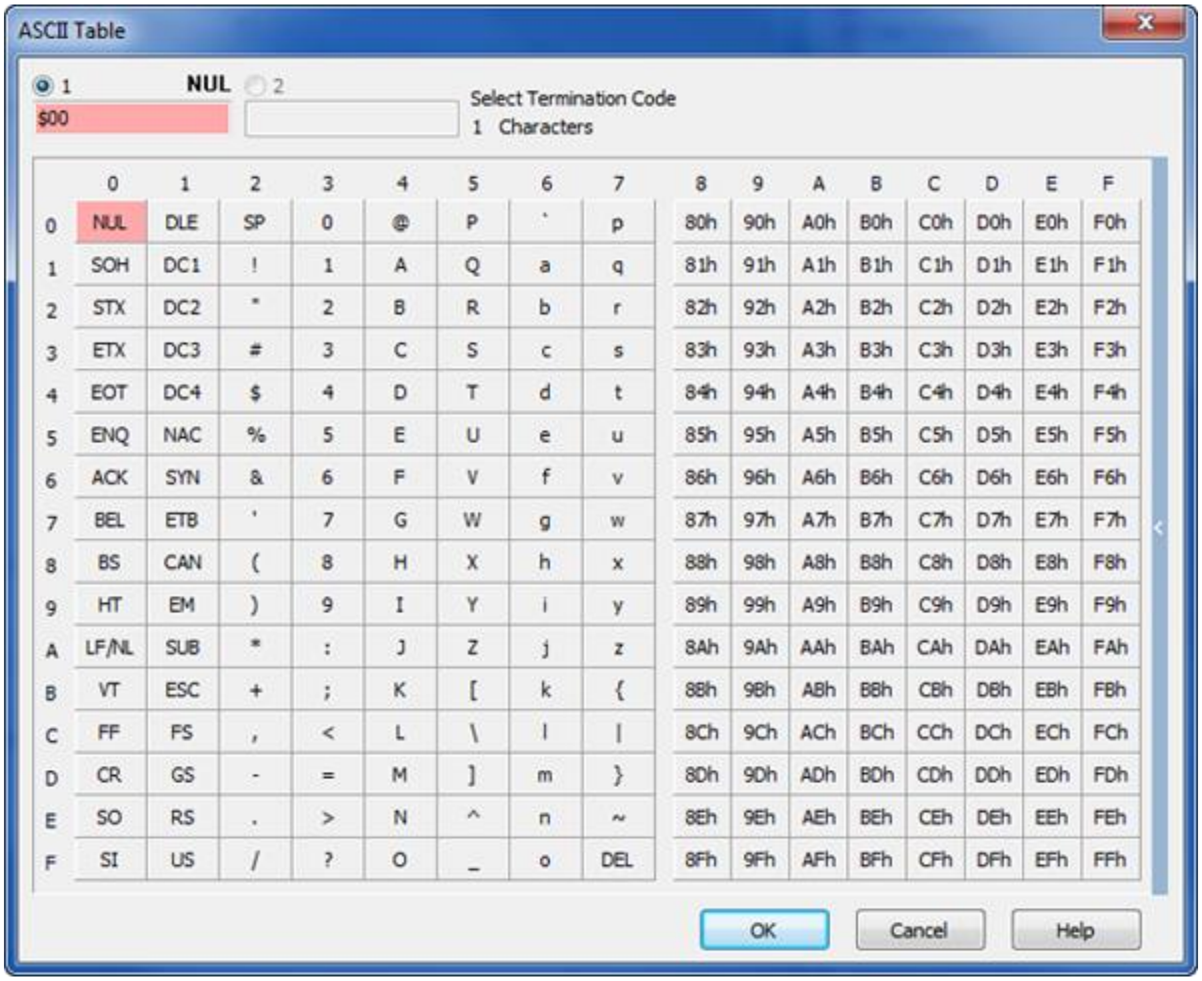
7 Embedded Discrete Address: Click this button to open a dialog used to enter a **Discrete Memory Address** and select the format. Please refer to the help topic **Embedded Address** for details.

8 Simulate: Click this button to open a dialog used to enter values to **Simulate** the **Data** in the **Memory Addresses** you **Embedded** in the **ASCII Text Message**. Please refer to the help topic **Static Message Simulate** for details.

9 Possible Message Length: The **CLICK Software** automatically calculates the **Possible Message Length** and displays it here.

10 Dynamic Text Message (Max: 128 Characters): Select **Dynamic Text Message** to **Send** a **Message** stored in a series of **Text Addresses**. Select the **Text Address** and the **Number of Bytes** using the **Start Address** and the **Number of Bytes Browse Icon** ... respectively.

11 Termination Code: Click on the checkbox to select. The **Termination Code** fields for **1 Character** and **2 Character** become active. Select the number of **Characters** to use and enter the **ASCII Code** in the field(s). Click on the **ASCII Table** button to open the **ASCII Table** shown below. Use this table to quickly select the desired **ASCII Code**.



ASCII Table

(● 1) NUL (○ 2)

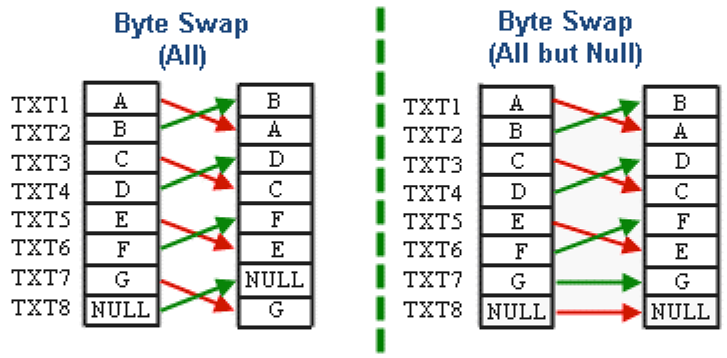
$00

Select Termination Code

1 Characters

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	DLE	SP	0	@	P	`	p	80h	90h	A0h	B0h	C0h	D0h	E0h	F0h
1	SOH	DC1	!	1	A	Q	a	q	81h	91h	A1h	B1h	C1h	D1h	E1h	F1h
2	STX	DC2	"	2	B	R	b	r	82h	92h	A2h	B2h	C2h	D2h	E2h	F2h
3	ETX	DC3	#	3	C	S	c	s	83h	93h	A3h	B3h	C3h	D3h	E3h	F3h
4	EOT	DC4	$	4	D	T	d	t	84h	94h	A4h	B4h	C4h	D4h	E4h	F4h
5	ENQ	NAC	%	5	E	U	e	u	85h	95h	A5h	B5h	C5h	D5h	E5h	F5h
6	ACK	SYN	&	6	F	V	f	v	86h	96h	A6h	B6h	C6h	D6h	E6h	F6h
7	BEL	ETB	'	7	G	W	g	w	87h	97h	A7h	B7h	C7h	D7h	E7h	F7h
8	BS	CAN	(	8	H	X	h	x	88h	98h	A8h	B8h	C8h	D8h	E8h	F8h
9	HT	EM	)	9	I	Y	i	y	89h	99h	A9h	B9h	C9h	D9h	E9h	F9h
A	LF/NL	SUB	*	:	J	Z	j	z	8Ah	9Ah	AAh	BAh	CAh	DAh	EAh	FAh
B	VT	ESC	+	;	K	[	k	{	8Bh	9Bh	ABh	BBh	CBh	DBh	EBh	FBh
C	FF	FS	,	<	L	\	l	|	8Ch	9Ch	ACh	BCh	CCh	DCh	ECh	FCh
D	CR	GS	-	=	M	]	m	}	8Dh	9Dh	ADh	BDh	CDh	DDh	EDh	FDh
E	SO	RS	.	>	N	^	n	~	8Eh	9Eh	AEh	BEh	CEh	DEh	EEh	FEh
F	SI	US	/	?	O	_	o	DEL	8Fh	9Fh	AFh	BFh	CFh	DFh	EFh	FFh

12 Byte Swap: Select the checkbox if **Byte Swap** will be used. If selected, the **ALL** and **All But Null** selections will become active. Click one of these methods to select it.



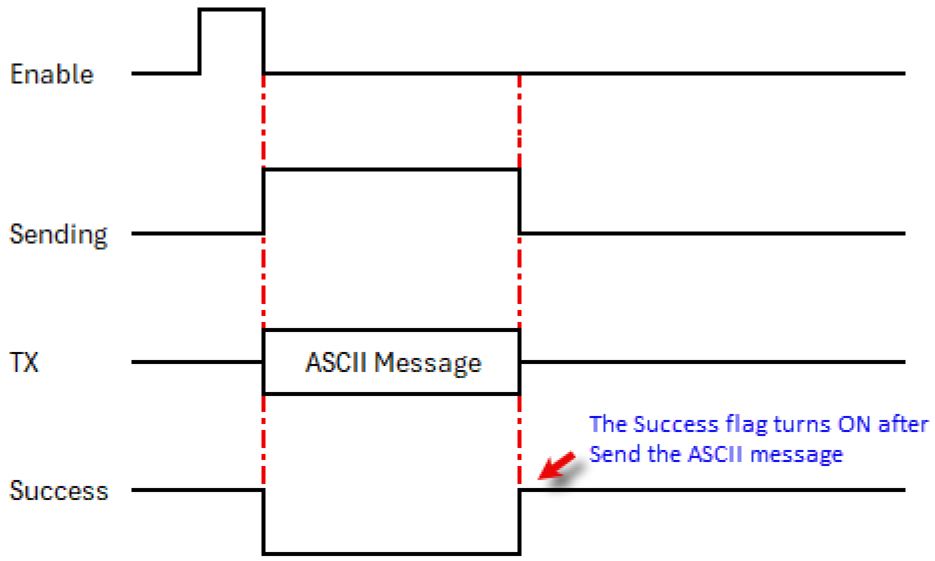
13 Sending: Assign a **C Bit** as the Sending flag. This flag turns **O**N when the com port is **Sending** an **ASCII** message to the external device. Please refer to the **Timing Chart** below.

14 Success: Assign a **C Bit** as the Success flag. This flag turns **O**N after the com port **Sent** an **ASCII** message to the external device. The **Send** instruction just **Sends** the **ASCII** message to the external device and there is no way for the **CLICK PLC** to check if the external device **Received** the **ASCII** message successfully. This flag always turns **ON** after **Sending** and **ASCII** message. This flag turns **OFF** when the **Enable** input for this instruction is **Enabled** again. Please refer to the **Timing Chart** below.

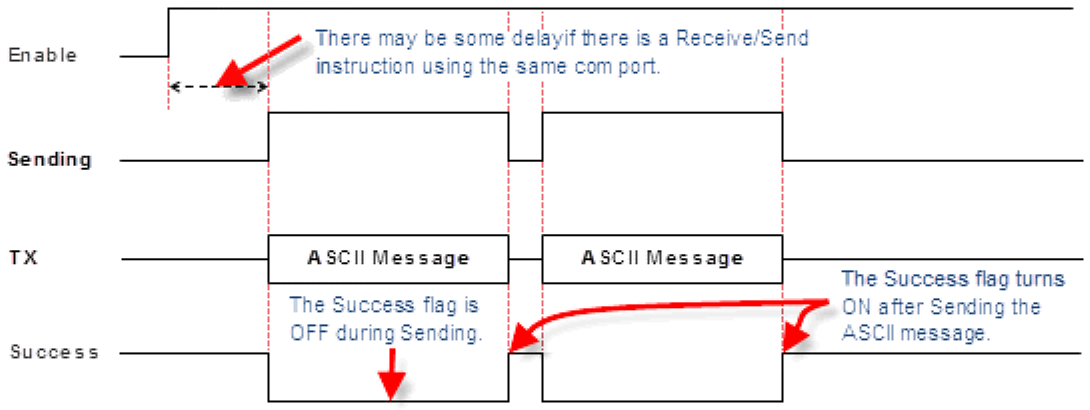
15 Error Flag: SC43:Out of Range is the PLC error code if any numeric value is out of the corresponding data types range.

Timing Charts

Send an ASCII Message to the External Device Once



Send an ASCII Message to the External Device Continuously


Enable
There may be some delayif there is a Receive/Send instruction using the same com port.
Sending
TX
ASCII Message
ASCII Message
The Success flag turns ON after Sending the ASCII message.
The Success flag is OFF during Sending.
Success

Related Topics:

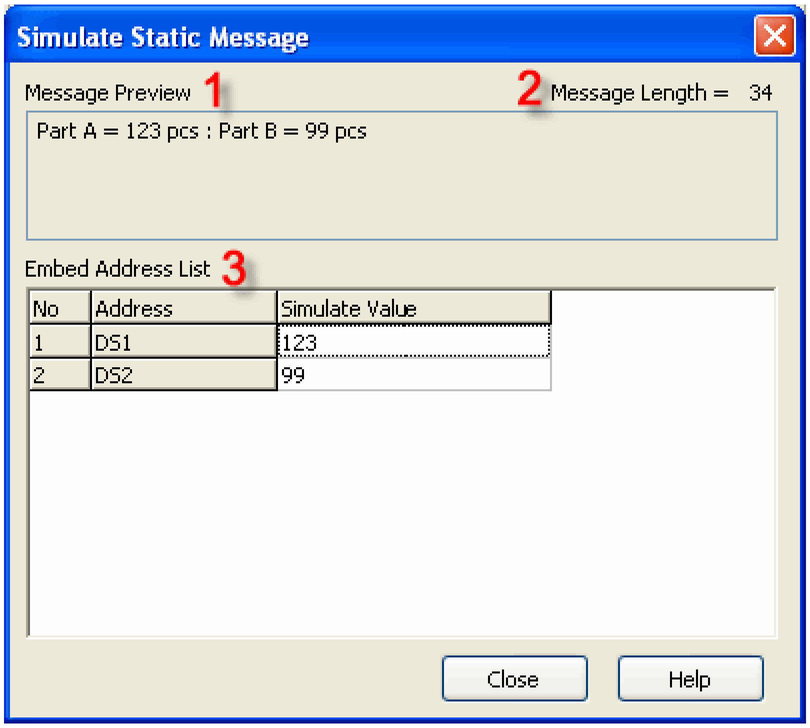
Send and Receive ASCII Instructions

Static Message Simulate

Description

With this dialog, you can assign test values to the **Memory Addresses** embedded in the **ASCII Message** and check the result.

Setup



Part A = 123 pcs : Part B = 99 pcs

No	Address	Simulate Value
1	DS1	123
2	DS2	99

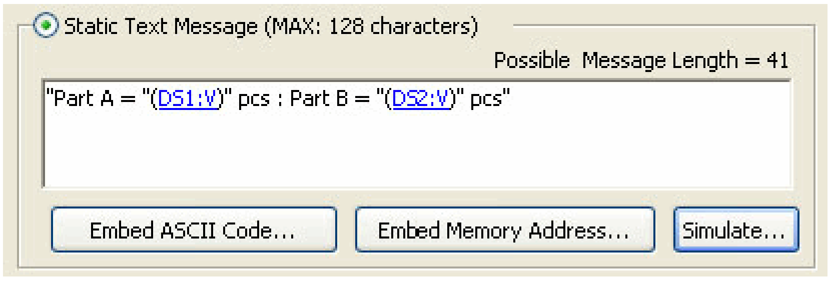
1 Message Preview: This field displays the **ASCII Message** that will be sent through the **Com Port**.

2 Message Length: This is the number of **Characters** used in the **ASCII Message** displayed in the **Message Preview** field.

3 Embed Address List: All **Memory Addresses Embedded** in the **ASCII Message** are displayed in this list. You can assign a **Simulation** value to each **Memory Address**. The **ASCII Message** in the **Message Preview** field is updated automatically.

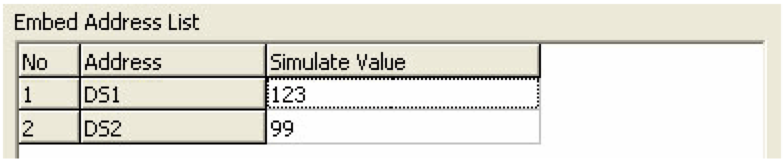
Example

This is an **Example** of the **Static Text Message**. **Memory Addresses DS1** and **DS2** are embedded in the **Message**.


Static Text Message (MAX: 128 characters)
Possible Message Length = 41
Embed ASCII Code...
Embed Memory Address...
Simulate...

"Part A = "(DS1:V)" pcs : Part B = "(DS2:V)" pcs"

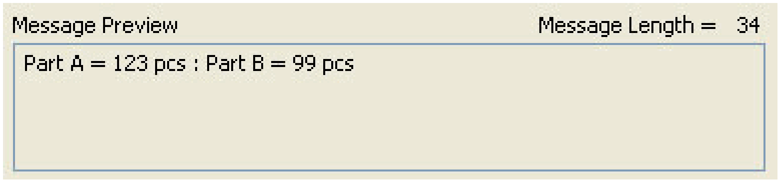
Click the **Simulate** button to open the **Simulate Static Message** dialog. Assign **123** to **DS1** and **99** to **DS2** as the **Simulation** value in the **Embed Address List** as shown below.



Embed Address List

No	Address	Simulate Value
1	DS1	123
2	DS2	99

The **ASCII Message** including the **Simulation** values is displayed in the **Message Preview** field.



Message Preview Message Length = 34

Part A = 123 pcs : Part B = 99 pcs

Embed Memory Address

Description

This dialog is used to **Embed** a **Memory Address** in the text message with several formatting options. This dialog has two **GUI** designs. One design is for the **DS**, **DD**, **DH**, **XD**, **YD**, **TD**, **CTD** and **SD Memory Addresses** and the other design is for the **DF Memory Addresses**.

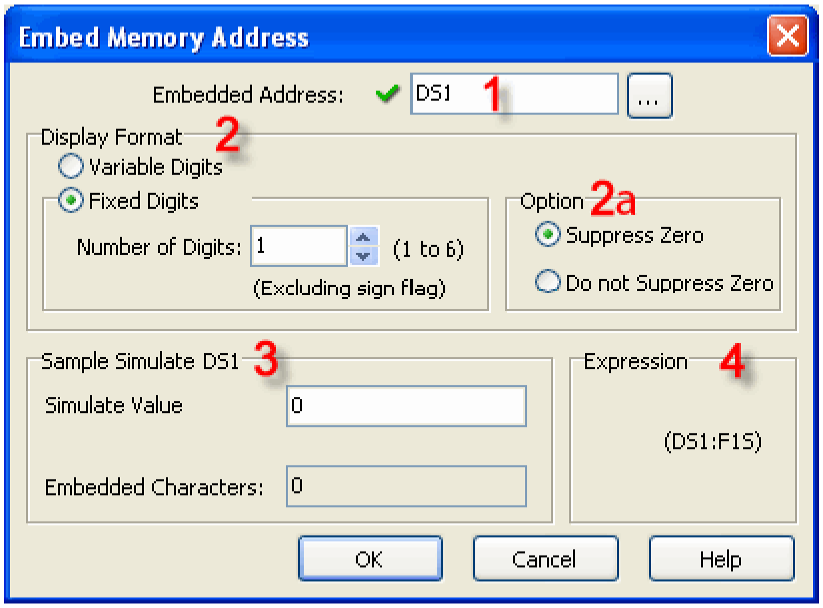
1. **Type A: DS**, **DD**, **DH**, **XD**, **YD**, **TD**, **CTD** and **SD**
2. **Type B: DF**

Note: You can not **Embedded Bit memory addresses** (**X**, **Y**, **C**, **T**, **CT** and **SC**) in the text message.

Setup

(Type A)

This **Setup** applies only for the **DS**, **DD**, **DH**, **XD**, **YD**, **TD**, **CTD** and **SD Memory Addresses**.



1 Embedded Address: Enter a **Memory Address** to **Embed** in the text message or click the **Browse Button** to open the Address Picker to select a **Memory Address** from the list.

2 Display Format: Allows you to **Set up** how the data in the **Memory Address** is **Embedded** in the text message.

1. **Variable Digits:** The data in the **Memory Address** is **Embedded** in the text message as it is. The number of characters **Embedded** in the text message varies according to the data.
2. **Fixed Digits - Number of Digits:** This is the minimum **Number** of characters to occupy in the text message. If the actual data is longer than this setup, this setup is ignored and the entire data will be copied into the text message.

   The range of this option varies according to the memory type you select.

Memory Type	**Least Digits Range**
DS, TD, SD	1 to 5
DD, CTD	1 to 10

DH, XD, YD	1 to 4

2a Option: Allows you to select to **Suppress** the **Zeroes** or **Not** to **Suppress** the **Zeroes**. Click to select the desired **Option**.

Example:

DS1 = 123

Display Format	**Characters in the Text Message** **(See Note Below)**	**Expression**
Variable Digits	"123"	DS1:V
Fixed Digits, Least Digits = 1, Suppress Zero	"123"	DS1:F1S
Fixed Digits, Least Digits = 1, Do not Suppress Zero	"123"	DS1:F1
Fixed Digits, Least Digits = 5, Suppress Zero	" 123" (Two Space Characters + "123")	DS1:F5S
Fixed Digits, Least Digits = 5, Do not Suppress Zero	"00123"	DS1:F5

Note: Double quotations are not included in the actual text message.

3 Sample Simulate: Enter a test value in the **Simulate Value** field. The **Embedded Characters** field shows what **Characters** will be **Embedded** in the text message.

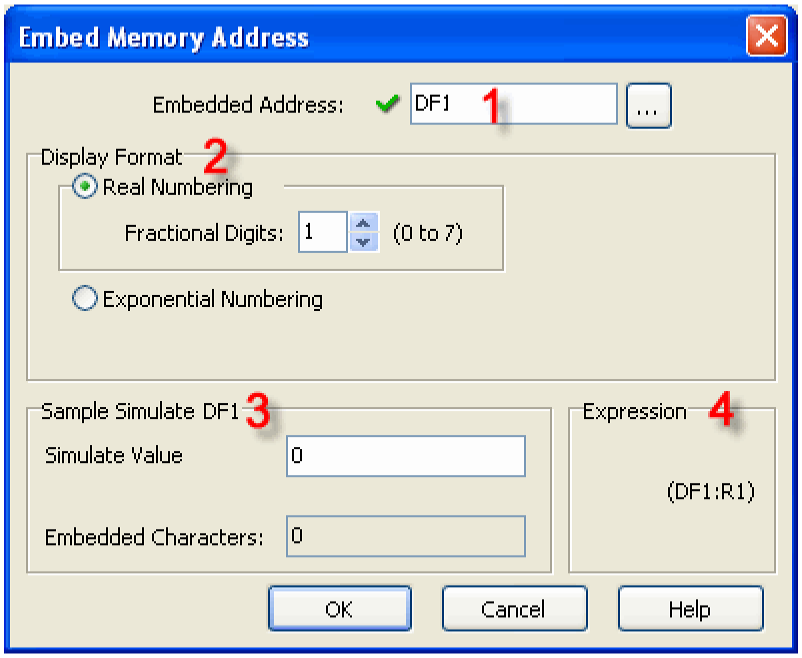
4 Expression: When the **Memory Address** is **Embedded** in the text message, one of the following **Expression Symbols** is attached to the **Memory Address** with a colon (e.g., **DS1:V**, **DH100:L4.S**). This will help when checking the **Format Setting** in the ladder program.

Expression	Meaning
V	Variable Digits
FnS	Fixed Digits, Number of Digits = n, Zero Suppression
Fn	Fixed Digits, Number of Digits = n, No Zero Suppression

Setup

(Type B)

This **Setup** applies only for the **DFMemory Address**



1 Embed Address: Enter a **DF Memory Address** to **Embed** in the text message or click the **Browse Button** to open the Address Picker to select a **Memory Address** from the list.

2 Display Format: Allows you to **Set up** the **Format** between the **Real Numbering** and the **Exponential Numbering**. When the **Real Numbering** is selected, you can specify the number of digits after the decimal point with the **Fractional Digits** parameter.

Example:

DF1 = 1.234

Digits Setting	Characters in the Text Message (See Note Below)	Expression
Real Numbering, Fractional Digits = 0	"1"	DF1:R0
Real Numbering, Fractional Digits = 4	"1.2340"	DF1:R4
Real Numbering, Fractional Digits = 7	" 1.2340000"	DF1:R7
Exponential Numbering	"1.2340000E+00"	DF1:E

Note: Double quotations are not included in the actual text message.

3 **Sample Simulate:** Enter a test value in the **Simulate Value** field. The **Embedded Characters** field shows what **Characters** will be **Embedded** in the text message.

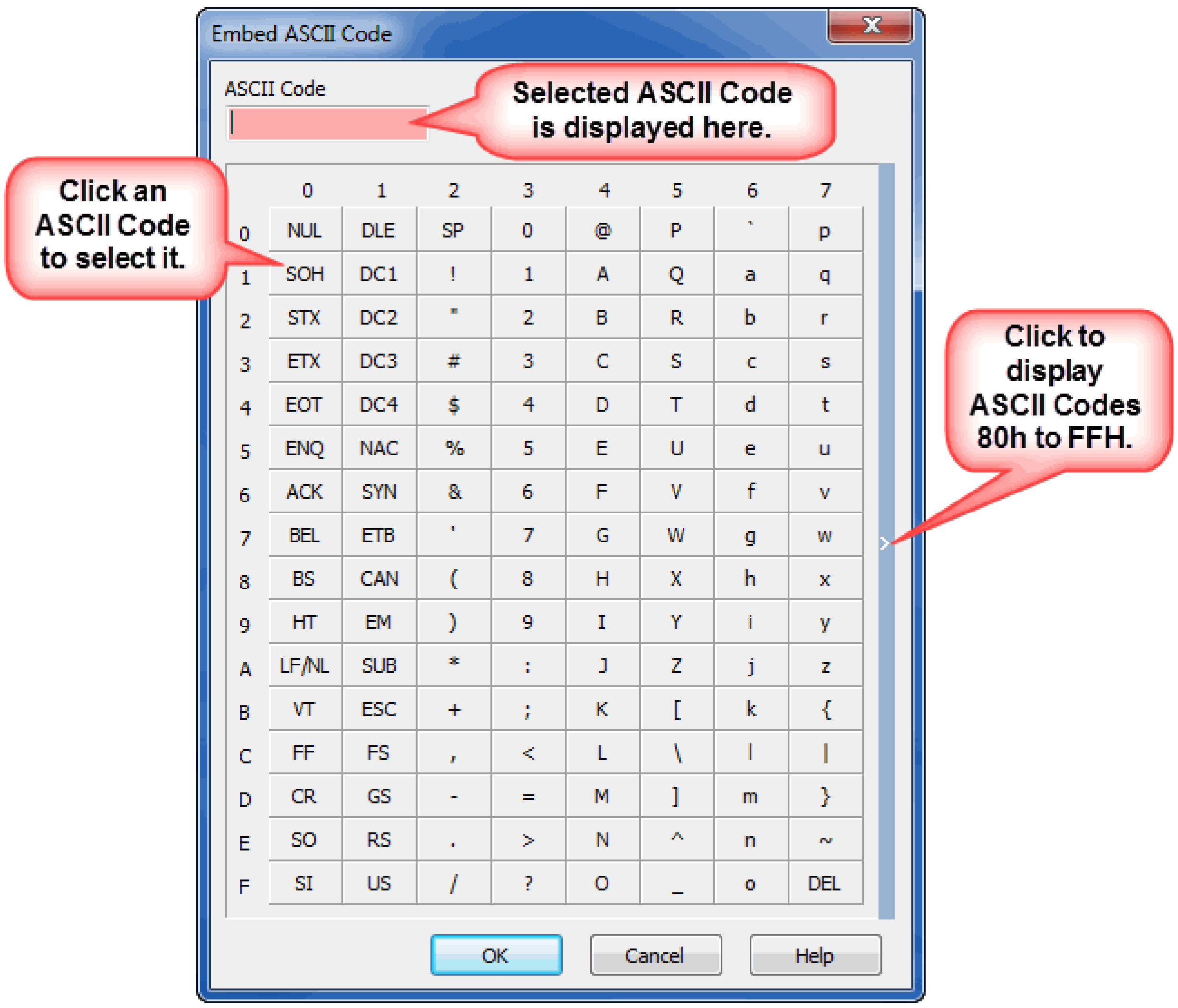
4 **Expression:** When the **Memory Address** is **Embedded** in the text message, one of the following **Expression Symbols** is attached to the **Memory Address** with a colon (e.g., **DF1:R4**, **DF100:E**). This will help when checking the **Format Setting** in the ladder program.

Expression	Meaning
Rn	Real Numbering, Fractional Digits = n
E	Exponential Numbering

Send Instruction: ASCII: Embed ASCII Code

Description

The dialog is used to select an **ASCII Code** (**00h to FFh**) to **Embed** in the **ASCII Text Message** for the Send Instruction: ASCII.


Embed ASCII Code
ASCII Code
Selected ASCII Code is displayed here.
Click an ASCII Code to select it.
Click to display ASCII Codes 80h to FFH.
OK
Cancel
Help

	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAC	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF/NL	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

Receive Instruction: MODBUS RTU

Description

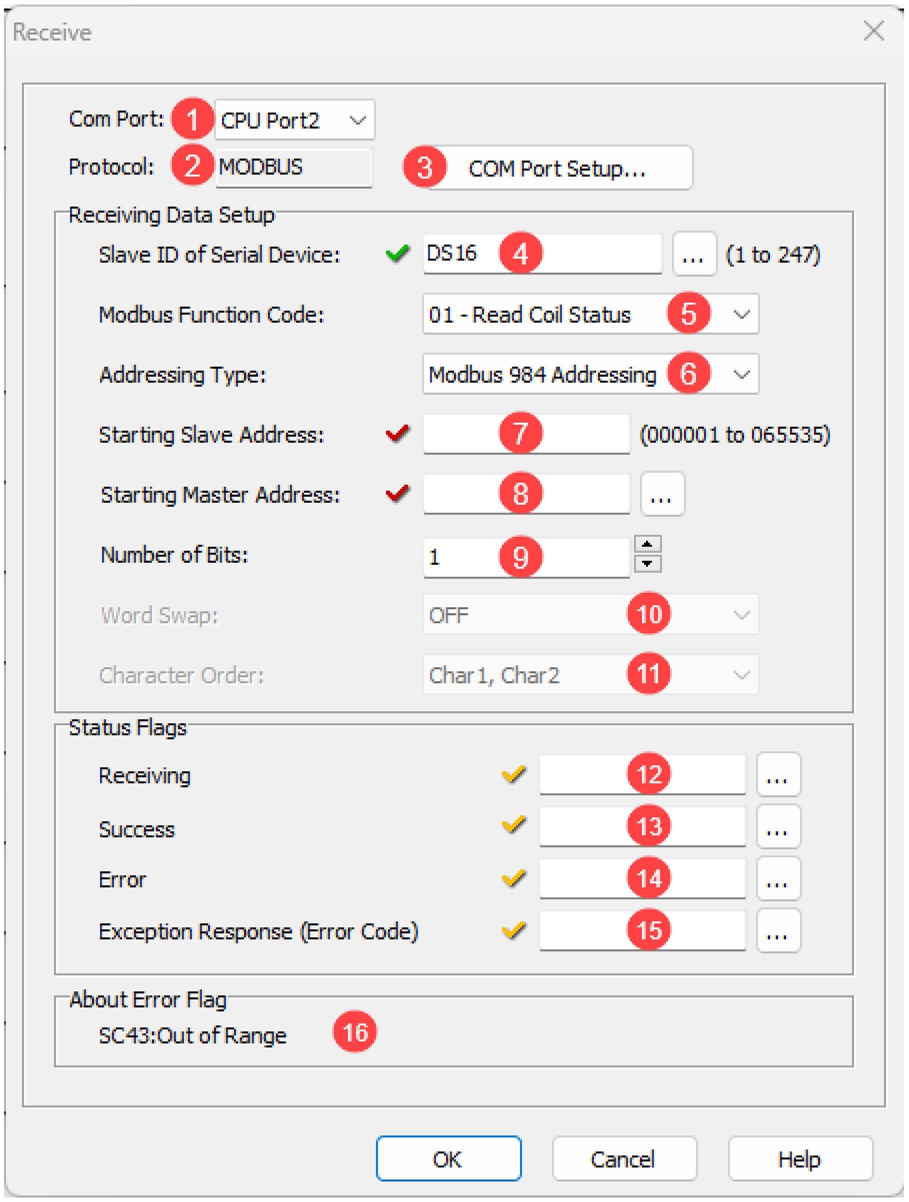
The **Receive** instruction allows you to use **Com Port 2 or 3** (if available) on the **CLICK CPU** modules, or **Com Port 1 or 2** of a **C2-DCM module**, as a network master and read data from external devices. The **CLICK CPU** modules support the **MODBUS (RTU)** and **ASCII** protocols. This help topic covers the **MODBUS** protocol. Please refer to the help topic Receive Instruction: ASCII for the **ASCII** protocol.

If you use a **Com Port** on the **CLICK CPU** modules as a **MODBUS** slave, you don't need to use this instruction. Setup the **Com Port** to match the configuration of the **MODBUS** network and assign a unique node address.

Note: For the **Com Port** wiring info, please refer to the CLICK PLC Hardware Manual or CLICK PLUS PLC Hardware Manual.

Note: If you need more detailed information about the **MODBUS** protocol, we recommend to visit the following web site: www.MODBUS.org

Setup



Receive

Com Port: 1 CPU Port2

Protocol: 2 MODBUS

3 COM Port Setup...

Receiving Data Setup

Slave ID of Serial Device:	DS16 4	(1 to 247)
Modbus Function Code:	01 - Read Coil Status 5	
Addressing Type:	Modbus 984 Addressing 6	
Starting Slave Address:	7	(000001 to 065535)
Starting Master Address:	8	
Number of Bits:	1 9	
Word Swap:	OFF 10	
Character Order:	Char1, Char2 11	

Status Flags

Receiving	12
Success	13
Error	14
Exception Response (Error Code)	15

About Error Flag

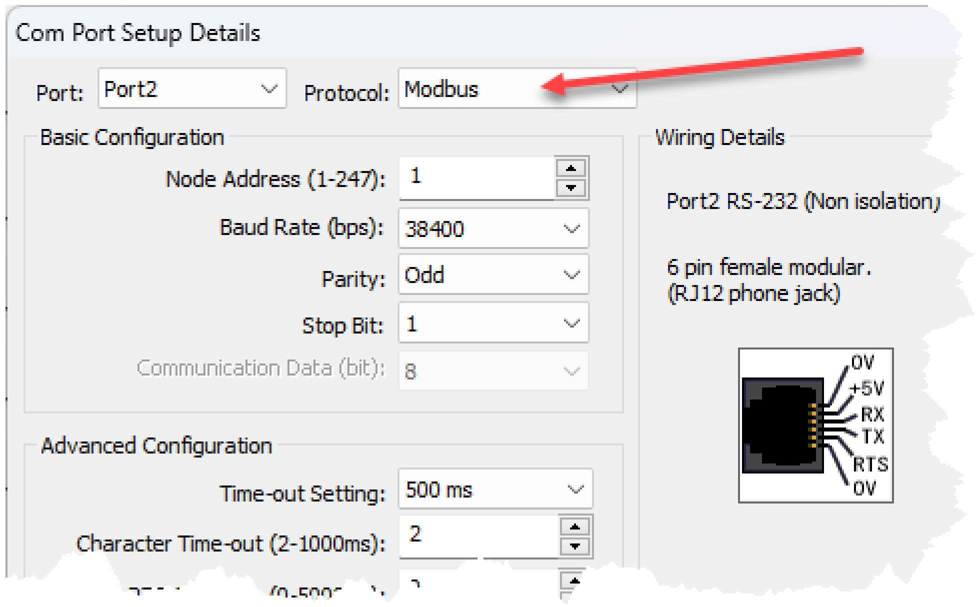
SC43:Out of Range 16

OK | Cancel | Help

1 Com Port: Click the down arrow to select a **Port** from the available list. **Port2** is default.

2 Protocol: This field displays the **Protocol Type** that is currently selected. The two choices are **MODBUS** and **ASCII**. To change from **MODBUS** to **ASCII**, see **Item 3**.

3 COM Port Setup: Click on this button to open the **Com Port Setup** window shown below. From this window click on the down arrow for the **Protocol** field and select **ASCII** or **MODBUS**


Com Port Setup Details

Port: Port2 Protocol: Modbus

Basic Configuration

Node Address (1-247): 1

Baud Rate (bps): 38400

Parity: Odd

Stop Bit: 1

Communication Data (bit): 8

Advanced Configuration

Time-out Setting: 500 ms

Character Time-out (2-1000ms): 2

Wiring Details

Port2 RS-232 (Non isolation)

6 pin female modular. (RJ12 phone jack)

.

Note: If the **Protocol** parameter is grayed out, it means there is at least one **Receive** or **Send** instruction assigned to the **Com Port** already. If you still want to change the **Protocol**, please delete all **Receive/Send** instructions first.

4 Slave ID (1-247): Select the **Node ID** of the **MODBUS** slave that you want to read data from. You can click the address picker [...] or enter a DS register to make this value a variable to enable communications with up to 247 different Modbus devices. If the value in the DS register is outside the range of 1-247, SC43 "Out of Range" will be set to TRUE.

5 MODBUS Function Code: Select one of the following **MODBUS** function codes:

- **01 – Read Coil Status**
- **02 – Read Input Status**
- **03 – Read Holding Registers**
- **04 – Read Input Registers**

6 Addressing Type: Select one of the following **MODBUS Addressing Types**:

Addressing Type	Description
MODBUS 984 Addressing	This Addressing is patterned after the Modicon PLC addressing, which many devices support. 0***** are Coils (Read/Write). 1***** are Input Bits (Read only).

	3***** are Input Registers (Read Only) 4***** are Holding Registers (Read/Write)
MODBUS Hex Addressing	This Addressing is patterned after what the MODBUS protocol actually requests, which is sim
CLICK Addressing	If the MODBUS slave is the CLICK PLC, we recommend to use this CLICK Addressing because y Address.

7 Starting Slave Address: Enter the **Slave Address** to start reading data from. The table below shows the valid **Slave Addresses**. If **CLICK Addressing**is selected, you can use the **Address Picker** by clicking the ⋯ icon.

MODBUS Addressing	**Function Code**			
	01	**02**	**03**	**04**
MODBUS 984	1 to 65535	100001 to 165535	400001 to 465535	300001 to 365535
MODBUS Hex	0h to FFFEh	0h to FFFEh	0h to FFFEh	0h to FFFEh
CLICK	X, Y, C, T, CT and SC	X, Y, C, T, CT and SC	CTD, DS, DD, DH, DF, SD, TD, YD and TXT	CTD, DS, DD, DH, DF, SD, TD, YD and TXT

8 Starting Master Address: Enter the **Starting Memory Address** of the **Master CLICK CPU** module to save the read data from the **MODBUS** slave. You can use the **Address Picker** by clicking the ⋯ icon.

Function Code			
01	**02**	**03**	**04**
Y and C	Y and C	CTD, DS, DD, DH, DF, SD, TD, YD and TXT	CTD, DS, DD, DH, DF, SD, TD, YD and TXT

9 Number of Bits: Enter the size of the data to read from the **MODBUS** slave.

10 Word Swap: This option is only available when the **Starting Master Address** is a **DD** or **DFMemory Address**. These **Memory Addresses** have **32 bit** data length, so they can store the data read from two registers in the **MODBUS** slave. You can swap the order of the data to be stored in the **DD** or **DF Memory Address**.

11 Character Order: This option is not available with the current **Firmware Version**.

12 Receiving: Assign a **C bit** as the **Receiving** flag. This **C bit** turns **ON** when the **Com Port** is sending a read request to a **MODBUS** slave with this instruction. Please refer to the **Timing Chart** shown below.

Note: This **Receiving** flag is **OFF** when another **Receive** instruction is requesting data from the **MODBUS** slave.

13 Success: Assign a **C bit** as the **Success** flag. This **C bit** turns **ON** after the **Com Port** received data from the **MODBUS** slave successfully. It stays **ON** until the instruction is re-enabled. Please refer to the **Timing Chart** shown below.

14 E rror: Assign a **C bit** as the **Error** flag. This **C bit** turns **ON** after the **Com Port** could not receive data from the **MODBUS** slave successfully. It stays **ON** until the instruction is re-enabled. Please refer to the **Timing Chart** shown below.

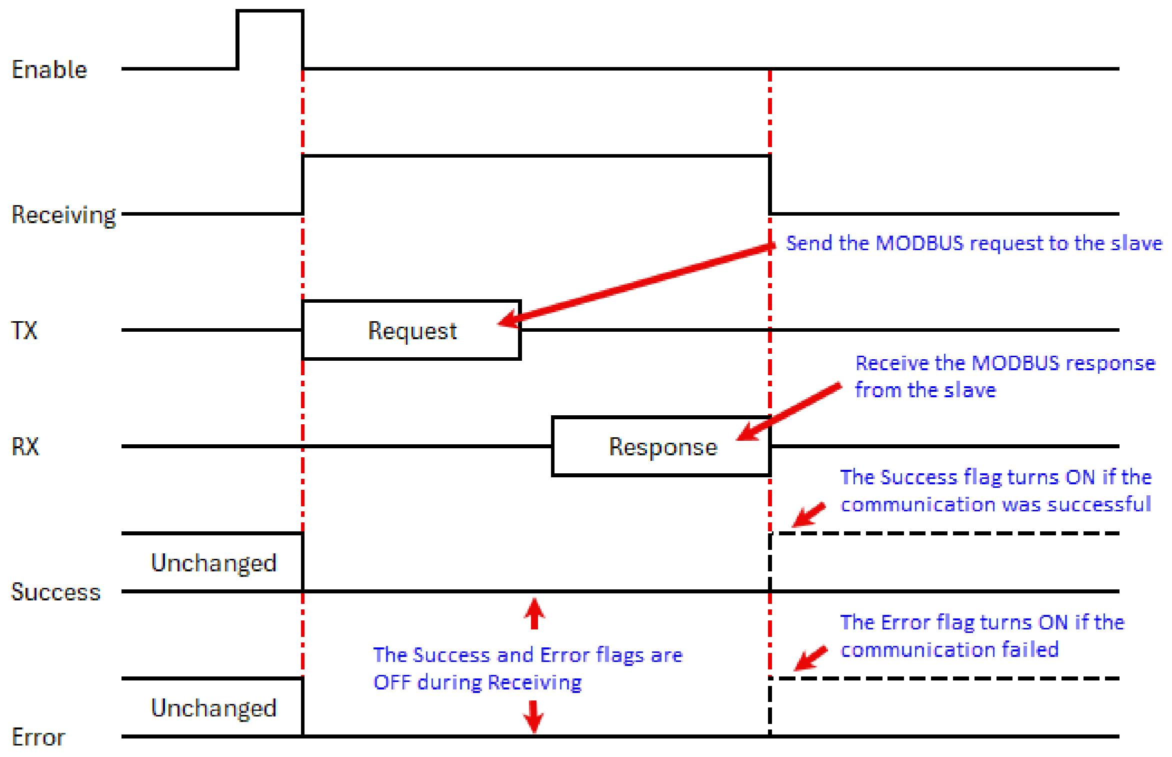
15 Exception Response (Error Code): Assign a **DS** or **DD Memory Address** to store the **Exception Response** from the **MODBUS** slave.

16 About Error Flag If you have defined the SlaveID using a DS address, and that value is outside the range of 1-247, System Bit SC43 will be set TRUE.

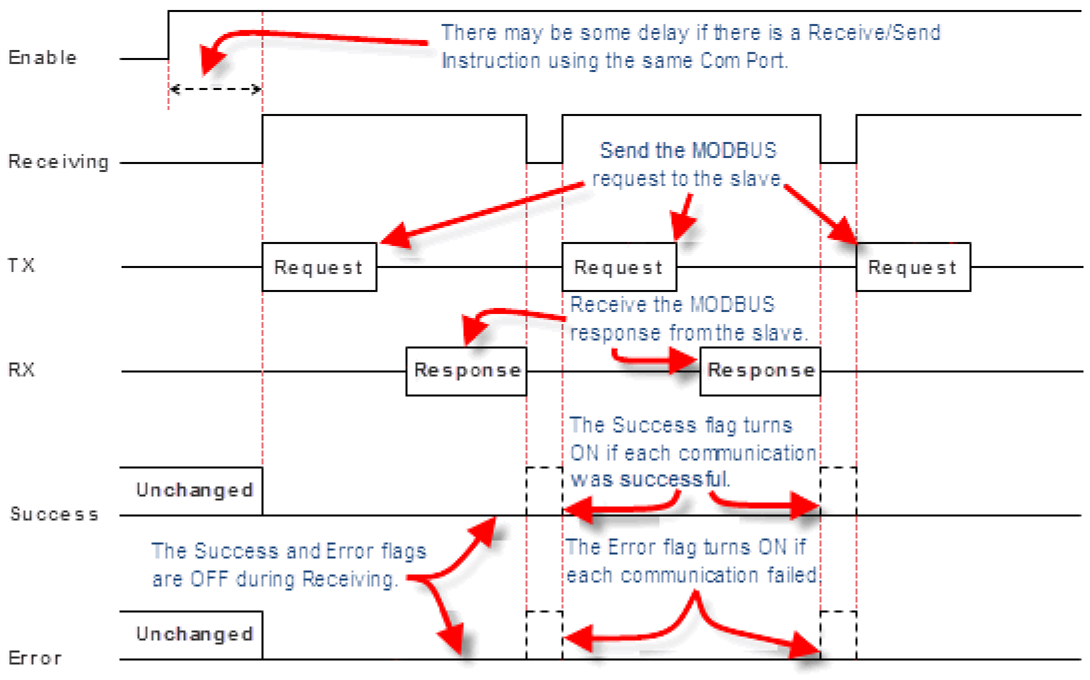
Note: Each **MODBUS** slave connected to the **CLICK PLC** would support a different set of the **Exception Response**. Please refer to the **User Manual/Specifications** of each **MODBUS Slave**. If you are using a **CLICK PLC** as a **MODBUS Slave**, please refer to this topic for the Exception Response that the **CLICK PLC** supports.

Timing Charts

Read Data from a MODBUS Slave Once


Enable
Receiving
TX
RX
Success
Error
Request
Response
Unchanged
Unchanged
Send the MODBUS request to the slave
Receive the MODBUS response from the slave
The Success flag turns ON if the communication was successful
The Error flag turns ON if the communication failed
The Success and Error flags are OFF during Receiving

Read Data from a MODBUS Slave Continuously


Enable
Receiving
TX
RX
Success
Error
There may be some delay if there is a Receive/Send Instruction using the same Com Port.
Send the MODBUS request to the slave
Request
Request
Request
Receive the MODBUS response from the slave.
Response
Response
The Success flag turns ON if each communication was successful.
Unchanged
Unchanged
The Success and Error flags are OFF during Receiving.
The Error flag turns ON if each communication failed.

Related Topics:

Receive Instruction: ASCII

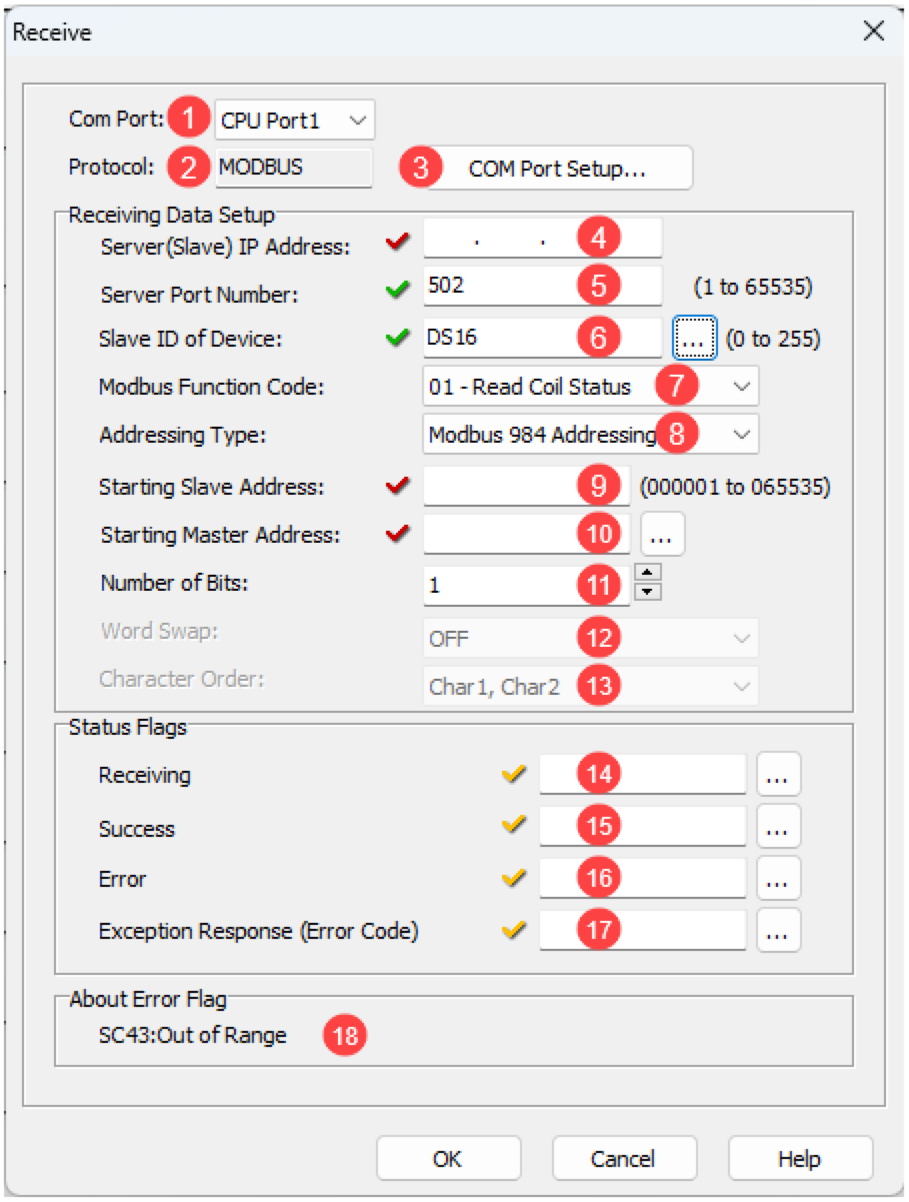
Receive Instruction: MODBUS TCP

Description

The **Receive** instruction allows you to use **Com Port 1** (**Ethernet**), **2** or **3** (if available) on the **CLICK CPU** modules as a network master and read data from external devices. The **CLICK CPU** modules support the **MODBUS RTU**, **MODBUS TCP** and **ASCII** protocols. This help topic covers the **MODBUS TCP** protocol. Please refer to the help topic Receive Instruction: MODBUS RTU for the **MODBUS RTU** protocol or Receive Instruction: ASCII for the **ASCII** protocol.

If you use **Com Port 1** (**Ethernet**) on the **CLICK CPU** modules as a **MODBUS TCP** server (slave), you don't need to use this instruction.

If you need more detailed information about the **MODBUS** protocol, we recommend to visit the following web site: www.MODBUS.org


Receive

Com Port: 1 CPU Port1

Protocol: 2 MODBUS 3 COM Port Setup...

Receiving Data Setup

Server(Slave) IP Address:	. . 4	
Server Port Number:	502 5	(1 to 65535)
Slave ID of Device:	DS16 6 ...	(0 to 255)
Modbus Function Code:	01 - Read Coil Status 7	
Addressing Type:	Modbus 984 Addressing 8	
Starting Slave Address:	9	(000001 to 065535)
Starting Master Address:	10 ...	
Number of Bits:	1 11	
Word Swap:	OFF 12	
Character Order:	Char1, Char2 13	

Status Flags

Receiving	14 ...
Success	15 ...
Error	16 ...
Exception Response (Error Code)	17 ...

About Error Flag

SC43:Out of Range 18

1 COM Port: Select **Port 1** to use the **Ethernet** port.

2 Protocol:COM Port 1 only supports the **MODBUS TCP** protocol.

3 COM Port Setup: Click on this button to open the Com Port Setup window.

4 Server (Slave) IP Address: Enter the **IP Address** of the target server.

5 Server Port Number: Enter the **Port Number** of the target server. ‘**502**’ is the default.

6 Slave ID of Serial Device: Typically used for **Modbus TCP** to Modbus **RTU** gateways but some server devices may require a certain value. This value may be either a number or a DS Address.

7 MODBUS Function Code: Select one of the following **MODBUS** function codes:

- **01 – Read Coil Status**
- **02 – Read Input Status**
- **03 – Read Holding Registers**
- **04 – Read Input Registers**

8 Addressing Type: Select one of the following **MODBUS Addressing Types**:

Addressing Type	Description
MODBUS 984 Addressing	This Addressing is patterned after the Modicon PLC addressing, which many devices support. 0***** are Coils (Read/Write). 1***** are Input Bits (Read only). 3***** are Input Registers (Read Only) 4***** are Holding Registers (Read/Write)
MODBUS Hex Addressing	This Addressing is patterned after what the MODBUS protocol actually requests, which is sim
CLICK Addressing	If the MODBUS slave is the CLICK PLC, we recommend to use this CLICK Addressing because y Address.

9 Starting Slave Address: Enter the **Slave Address** to start reading data from. The table below shows the valid **Slave Addresses**. If **CLICK Addressing**is selected, you can use the **Address Picker** by clicking the ⋯ icon.

MODBUS Addressing	Function Code			
	01	**02**	**03**	**04**
MODBUS 984	1 to 65535	100001 to 165535	400001 to 465535	300001 to 365535
MODBUS Hex	0h to FFFEh	0h to FFFEh	0h to FFFEh	0h to FFFEh
CLICK	X, Y, C, T, CT and SC	X, Y, C, T, CT and SC	CTD, DS, DD, DH, DF,	CTD, DS, DD, DH, DF,

			SD, TD, YD and TXT	SD, TD, YD and TXT

10 Starting Master Address: Enter the **Starting Memory Address** of the **Master CLICK CPU** module to save the read data from the **MODBUS** slave. You can use the **Address Picker** by clicking the ⋯ icon.

Function Code			
01	**02**	**03**	**04**
Y and C	Y and C	CTD, DS, DD, DH, DF, SD, TD, YD and TXT	CTD, DS, DD, DH, DF, TD, SD, YD and TXT

11 Number of Bits: Enter the size of the data to read from the **MODBUS** slave.

12 Word Swap: This option is only available when the **Starting Master Address** is a **DD** or **DFMemory Address**. These **Memory Addresses** have **32 bit** data length, so they can store the data read from two registers in the **MODBUS** slave. You can swap the order of the data to be stored in the **DD** or **DF Memory Address**.

13 Character Order: This option is not available with the current **Firmware Version**.

14 Receiving: Assign a **C bit** as the **Receiving** flag. This **C bit** turns **ON** when the **Com Port** is sending a read request to a **MODBUS** slave with this instruction. Please refer to the **Timing Chart** shown below.

Note: This **Receiving** flag is **OFF** when another **Receive** instruction is requesting data from the **MODBUS** slave.

15 Success: Assign a **C bit** as the **Success** flag. This **C bit** turns **ON** after the **Com Port** received data from the **MODBUS** slave successfully. It stays **ON** until the instruction is re-enabled. Please refer to the **Timing Chart** shown below.

16 Error: Assign a **C bit** as the **Error** flag. This **C bit** turns **ON** after the **Com Port** could not receive data from the **MODBUS** slave successfully. It stays **ON** until the instruction is re-enabled. Please refer to the **Timing Chart** shown below.

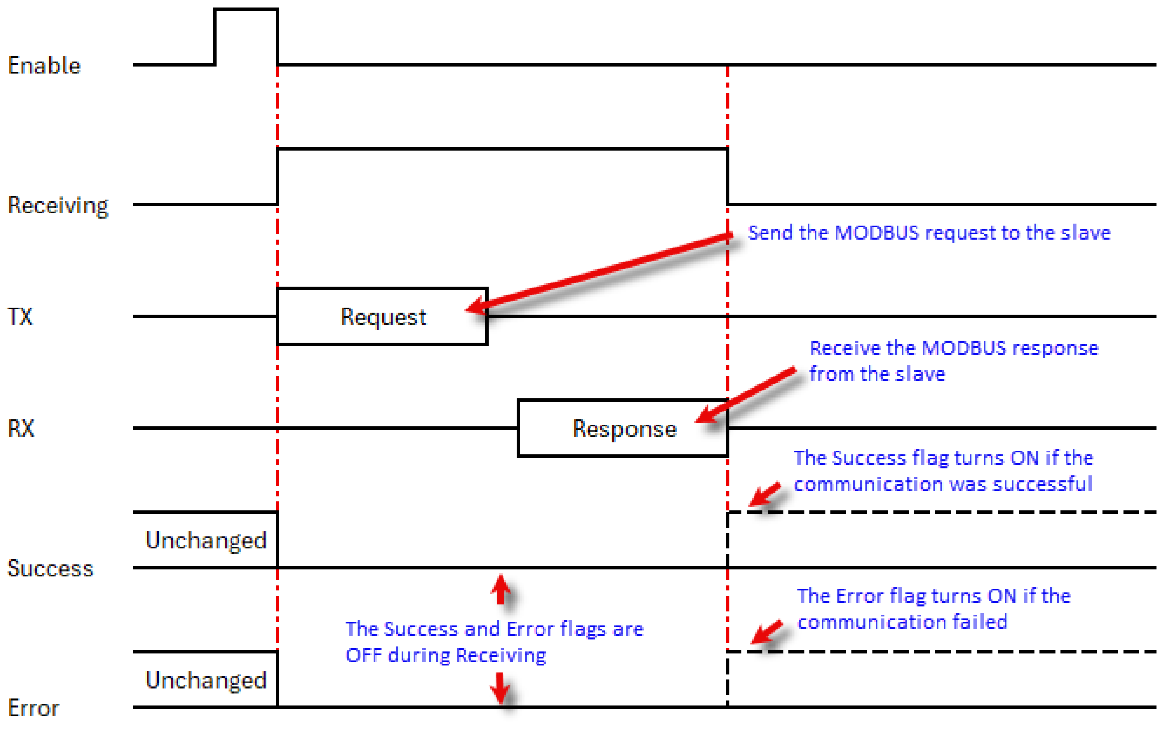
17 Exception Response (Error Code): Assign a **DS** or **DD Memory Address** to store the **Exception Response** from the **MODBUS** slave.

Note: Each **MODBUS** slave connected to the **CLICK PLC** would support a different set of the **Exception Response**. Please refer to the **User Manual/Specifications** of each **MODBUS Slave**. If you are using a **CLICK PLC** as a **MODBUS Slave**, please refer to this topic for the **Exception Response** that the **CLICK PLC** supports.

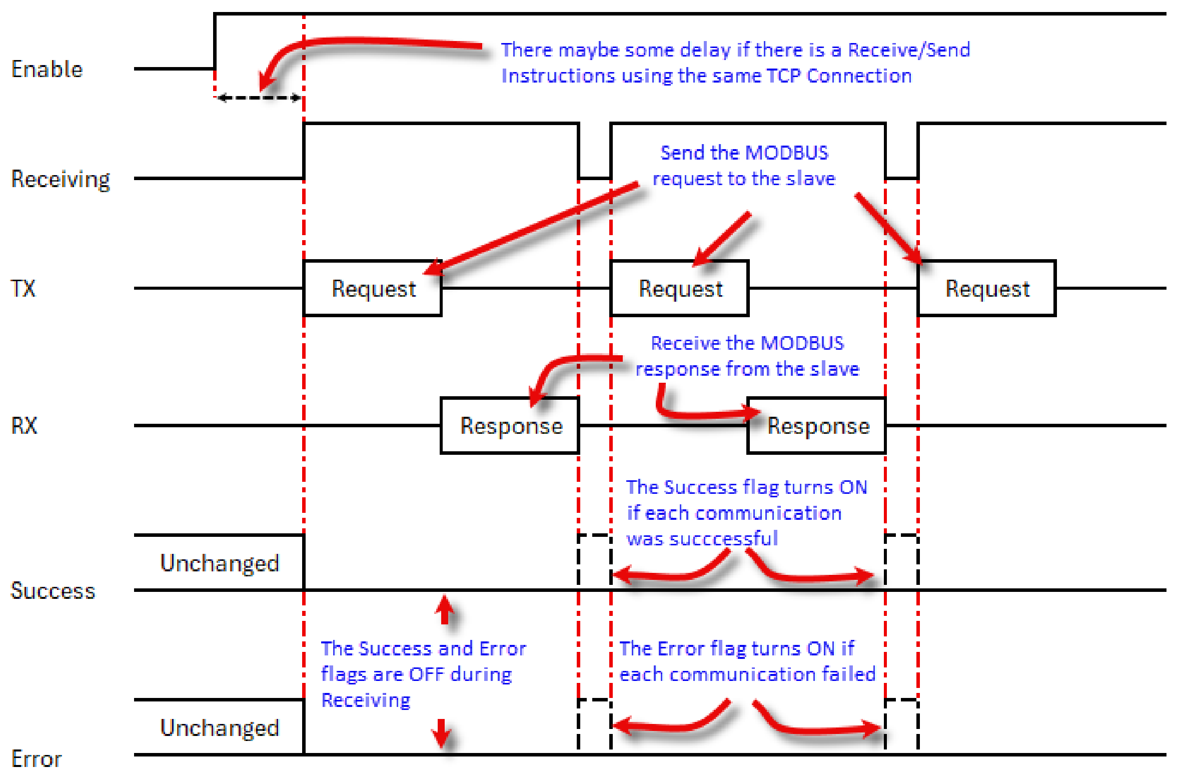
18 About Error Flag: If the Slave ID Is outside the range of 0 to 255, SC43: Out of Range will be set TRUE.

Timing Charts

Read Data from a MODBUS Slave Once


Enable
Receiving
TX
RX
Success
Error
Request
Response
Unchanged
Unchanged
Send the MODBUS request to the slave
Receive the MODBUS response from the slave
The Success flag turns ON if the communication was successful
The Error flag turns ON if the communication failed
The Success and Error flags are OFF during Receiving

Read Data from a MODBUS Slave Continuously


Enable
There maybe some delay if there is a Receive/Send Instructions using the same TCP Connection
Receiving
Send the MODBUS request to the slave
TX
Request
Request
Request
Receive the MODBUS response from the slave
RX
Response
Response
The Success flag turns ON if each communication was succcessful
Unchanged
Success
The Success and Error flags are OFF during Receiving
The Error flag turns ON if each communication failed
Unchanged
Error

Related Topics

Send Instruction: MODBUS RTU

Description

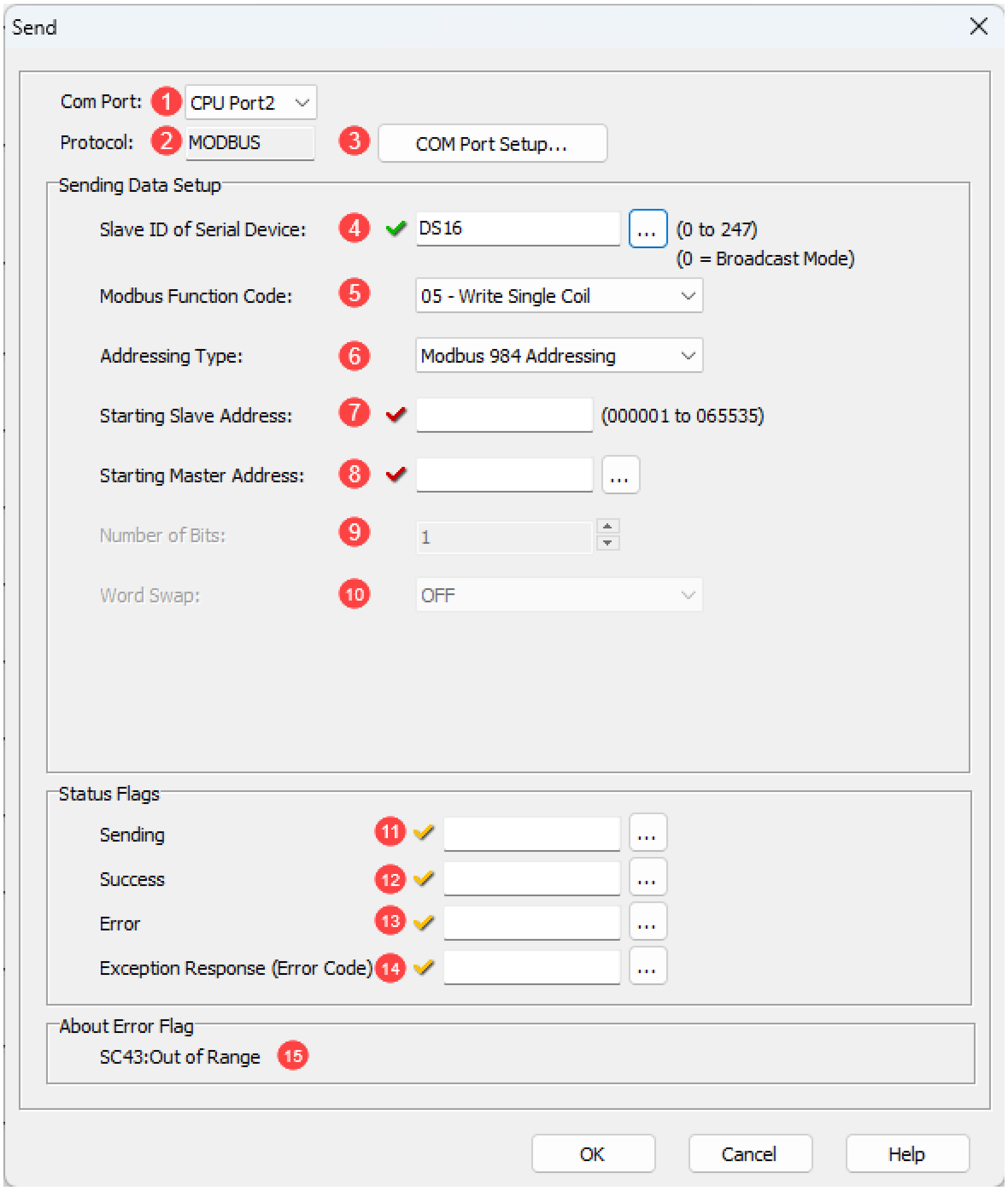
The **Send** instruction allows you to use **Com Port 2** or **3** (if available) on the **CLICK CPU** modules, or **Com Port 1 or 2** of a **C2-DCM module**, as a network **Master** and write data to external devices. The **CLICK CPU** modules support the **MODBUS (RTU)** and **ASCII** protocols. This help topic covers the **MODBUS** protocol. Please refer to the help topic Send Instruction: ASCII for the **ASCII** protocol.

If you use a **Com Port** on the **CLICK CPU** modules as a **MODBUS Slave**, you don't need to use this instruction. Set up the **Com Port** to match the configuration of the **MODBUS** network and assign a unique **Node Address**.

Note: For the **Com Port** wiring information, please refer to the CLICK PLC Hardware Manual or CLICK PLUS PLC Hardware Manual.

Note: If you need more detailed information about the **MODBUS** protocol, we recommend to visit the following web site: http://www.MODBUS.org

Setup



Send

Com Port: 1 CPU Port2

Protocol: 2 MODBUS

3 COM Port Setup...

Sending Data Setup

Slave ID of Serial Device: 4 ✓ DS16 (0 to 247) (0 = Broadcast Mode)

Modbus Function Code: 5 05 - Write Single Coil

Addressing Type: 6 Modbus 984 Addressing

Starting Slave Address: 7 ✓ (000001 to 065535)

Starting Master Address: 8 ✓

Number of Bits: 9 1

Word Swap: 10 OFF

Status Flags

Sending 11 ✓

Success 12 ✓

Error 13 ✓

Exception Response (Error Code) 14 ✓

About Error Flag

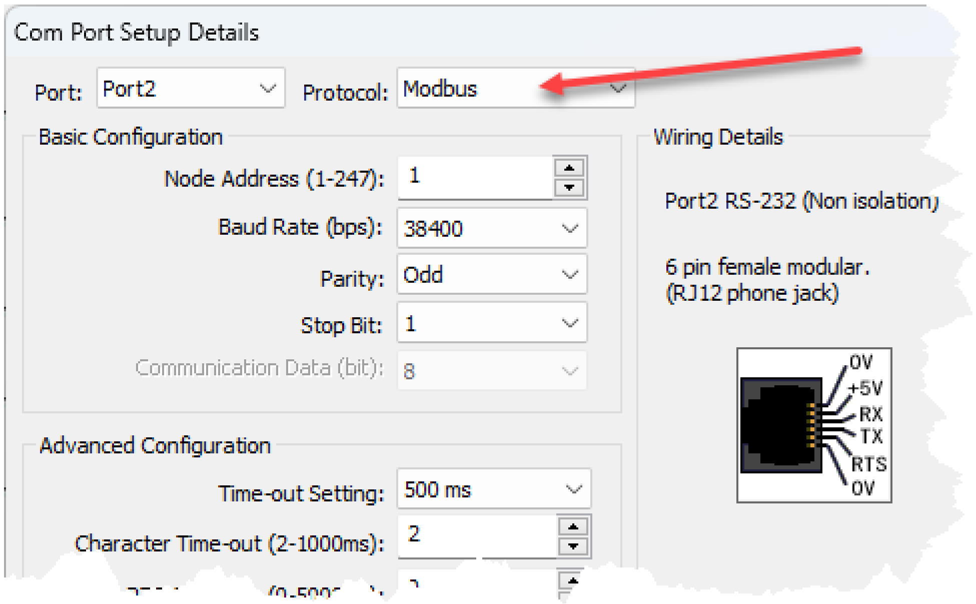
SC43:Out of Range 15

OK Cancel Help

1 Com Port: Click the down arrow to select a **Port** from the available list. **Port2** is default.

2 Protocol: This field displays the **Protocol Type** that is currently selected. The two choices are **MODBUS** and **ASCII**. To change from **MODBUS** to **ASCII**, see **Item3**.

3 COM Port Setup: Click on this button to open the **Com Port Setup** window shown below. From this window click on the down arrow for the **Protocol** field and select **ASCII** or **MODBUS**.



Com Port Setup Details

Port: Port2 Protocol: Modbus

Basic Configuration

Node Address (1-247): 1

Baud Rate (bps): 38400

Parity: Odd

Stop Bit: 1

Communication Data (bit): 8

Advanced Configuration

Time-out Setting: 500 ms

Character Time-out (2-1000ms): 2

Wiring Details

Port2 RS-232 (Non isolation)

6 pin female modular. (RJ12 phone jack)

Note: If the **Protocol** parameter is grayed out, it means that there is at least one**Receive** or **Send** instruction assigned to the **Com Port** already. If you still want to change the **Protocol**, please delete those **Receive/Send** instructions first.

4 Slave ID (0-247): Select the **Node ID** of the **MODBUS Slave** that you want to write data to, or select **0**. **Slave ID 0** is called **Broadcast Mode**. In the **Broadcast Mode**, the **MODBUS Master** can send a message to all **Slaves** in the network at the same time. In this mode, **Slaves** do not reply to the **MODBUS Master**. When you use this **Broadcast Mode**, please check if the **Slaves** in the network support this mode. The Slave ID may also be set using a DS Register. You can either enter a value or open the Address Picker to find an address. If the DS value is outside the range of 0-247, SC43 Out of range will be set TRUE.

Note: The **CLICK PLC** supports **Broadcast Mode** when it is used as a **MODBUS Slave**.

5 MODBUS Function Code: Select one of the following **MODBUS Function Codes**:

05 - Write Single Coil

06 - Write Single Register

15 - Write Multiple Coils

16 - Write Multiple Registers

6 Addressing Type: Select one of the following **MODBUS Addressing Types**:

Addressing Type	Explanation
MODBUS 984 Addressing	This **Addressing** is patterned after the **Modicon PLC Addressing**, which many devices support. **0******* are **Coils (Read/Write)** **4******* are **Holding Registers (Read/Write)**
MODBUS Hex Addressing	This **Addressing** is patterned after what the **MODBUS** protocol actually requests, which is simply a **Function Code** **+** an **Offset**.
CLICK Addressing	If the **MODBUS Slave** is the **CLICK PLC**, we recommend to use this **CLICK Addressing** because you can use the **Address Picker** to select the **Starting Slave Address**.

7 Starting Slave Address: Enter the **Slave Address** to start writing data to. Here are the valid **Slave addresses**. If you selected the **CLICK Addressing**, you can use the **Address Picker** by clicking the ⋯ icon.

MODBUS Addressing	**Function Code**			
	05	06	15	16
MODBUS 984	1 to 65535	400001 to 465535	1 to 65535	400001 to 465535
MODBUS Hex	0h to FFFEh	0h to FFFEh	0h to FFFEh	0h to FFFEh
CLICK	Y and C	DS, DH, SD, TD, YD & TXT	Y and C	CTD, DS, DD, DH, DF, SD, TD & TXT

8 Starting Master Address: Enter the **Starting Memory Address** of the **Master CLICK CPU** module to write the data to the **MODBUS Slave(s)**. You can use the **Address Picker** by clicking the ⋯ icon.

Function Code			
05	06	15	16
X, Y, C, T, CT and SC	DS, DH, XD, YD, TD, SD and TXT	X, Y, C, T, CT and SC	CTD, DS, DD, DH, DF, TD, SD and TXT

9 Number of Bits/Master Addresses: Enter the size of the data to write to the **MODBUS Slave(s)**.

10 Word Swap: When the **Starting Master Address** is a **DH**, **TD**, **DS** or **SD Memory Address**, this option is available. These **Memory Addresses** have **32** bit data length (**2 Words**), so each **Address** has data that can be written to two registers in the **MODBUS Slave**. You can **Swap** the order of those two word data in the **DH**, **TD**, **DS** or **SD Memory Addresses** to write to the **Slave**.

11 Sending: Assign a **C bit** as the **Sending Flag**. This **C bit** turns **ON** when the **Com Port** is **Sending** a write request to the **MODBUS Slave(s)** with this instruction. Please refer to the **Timing Chart**shown below.

Note: This **Sending Flag** is **OFF** when a **Receive** is reading data from a **MODBUS Slave** or another **Send** instruction is writing data to a **MODBUS Slave(s)**.

12 Success: Assign a **C bit** as the **Success Flag**. This **C bit** turns **ON** after the **Com Port** wrote data to the **MODBUS Slave(s)** successfully. It stays **ON** until this instruction is re-enabled. Please refer to the **Timing Chart**shown below.

Note: In the **Broadcast Mode (Slave ID = 0)**, there is no response from the **MODBUS Slaves**. The **Success bit** turns **ON** automatically after the **CLICK PLC** executed the **Send** instruction.

13 Error: Assign a **C bit** as the **Error Flag**. This **C bit** turns **ON** after the **Com Port** could not write data to the **MODBUS slave(s)** successfully. It stays **ON** until this instruction is re-enabled. Please refer to the **Timing Chart** shown below.

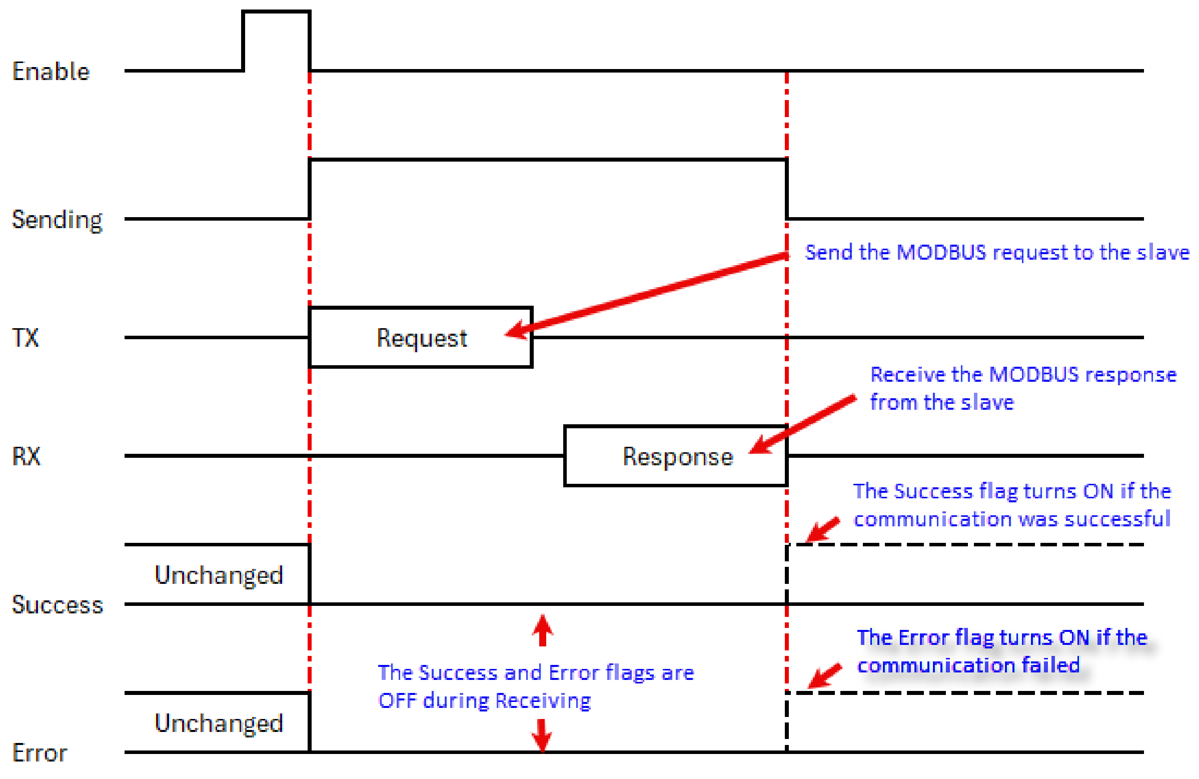
14 Exception Response (Error Code): Assign a **DS** or **DD Memory Address** to store the **Exception Response** from the **MODBUS Slave**.

15 About Error Flag If the Slave ID is outside the range of 0-247, SC43 - Out of Range will be set True

Note: Each MODBUS Slave supports a different set of the Exception Response. Please refer to the user manual/specifications of each MODBUS Slave. If you are using a CLICK PLC as a MODBUS Slave, please refer to this topic for the Exception Response that the CLICK PLC supports.

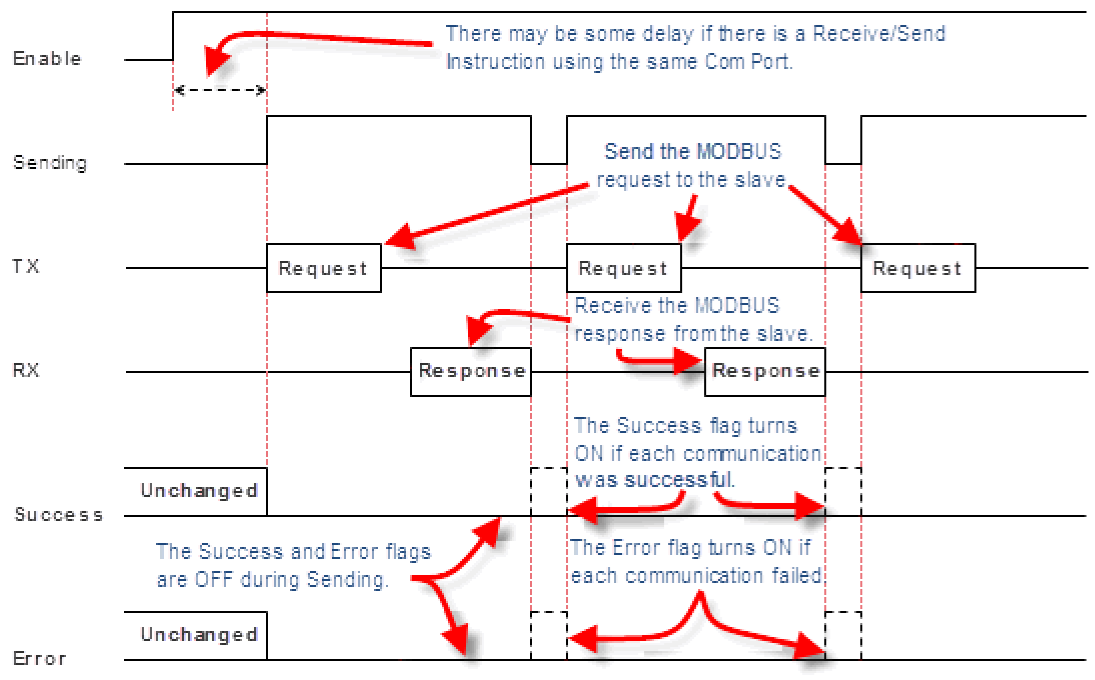
Timing Charts

Send Data to a MODBUS Slave Once



Enable
Sending
Send the MODBUS request to the slave
TX
Request
Receive the MODBUS response from the slave
RX
Response
The Success flag turns ON if the communication was successful
Unchanged
Success
The Error flag turns ON if the communication failed
The Success and Error flags are OFF during Receiving
Unchanged
Error

Send Data to a MODBUS Slave Continuously


Enable
There may be some delay if there is a Receive/Send Instruction using the same Com Port.
Sending
Send the MODBUS request to the slave.
TX
Request
Request
Request
Receive the MODBUS response from the slave.
RX
Response
Response
The Success flag turns ON if each communication was successful.
Unchanged
Success
The Success and Error flags are OFF during Sending.
The Error flag turns ON if each communication failed.
Unchanged
Error

Related Topics:

[Send Instruction: ASCII]

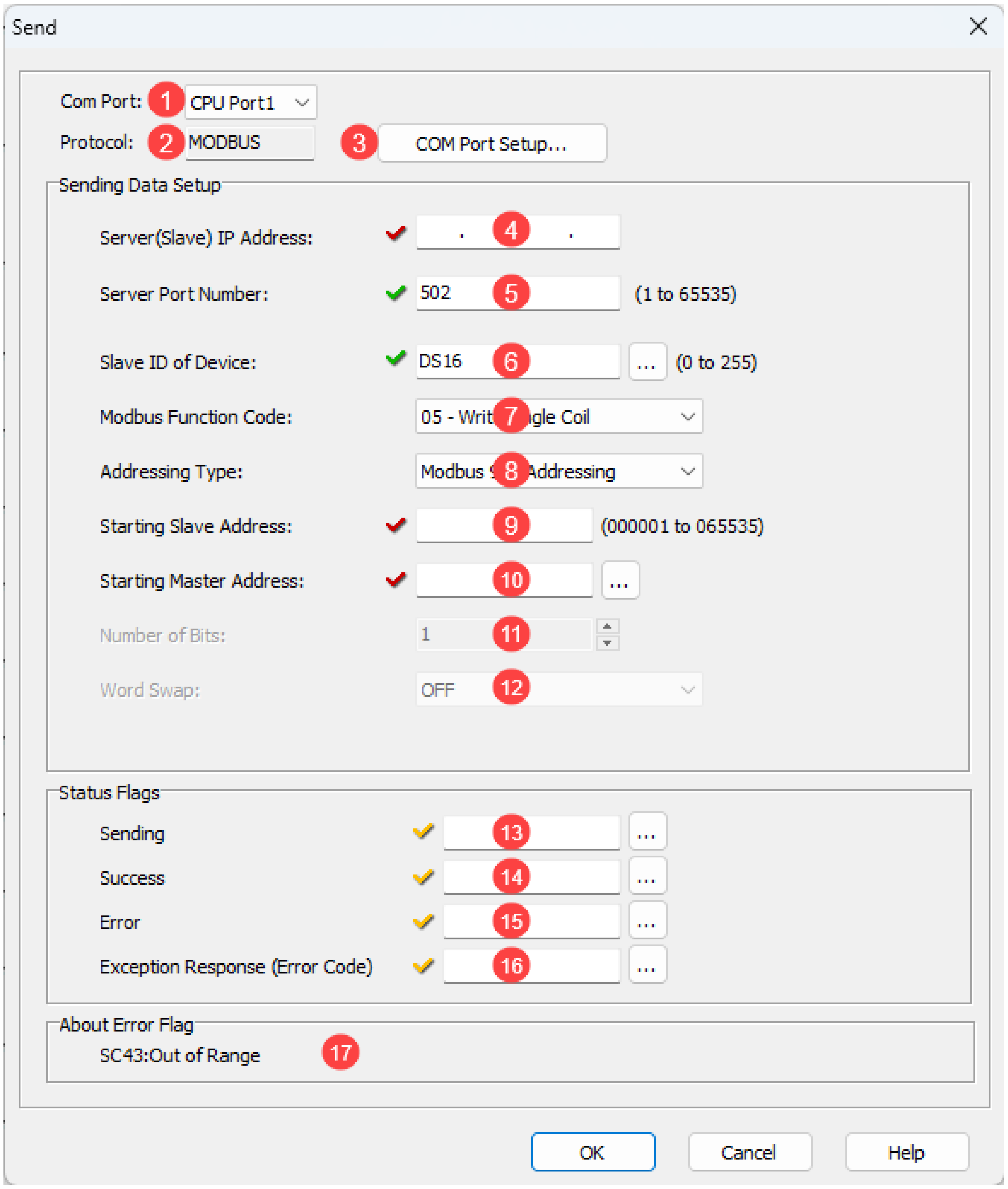
Send Instruction: MODBUS TCP

Description

The **Send** instruction allows you to use **Com Port 1** (**Ethernet**), **2** or **3** (if available) on the **CLICK CPU** modules as a network master and write data to external devices. The **CLICK CPU** modules support the **MODBUS RTU**, **MODBUS TCP** and **ASCII** protocols. This help topic covers the **MODBUS TCP** protocol. Please refer to the help topic Send Instruction: MODBUS RTU for the **MODBUS RTU** protocol or Send Instruction: ASCII for the **ASCII** protocol.

If you use **Com Port 1** (**Ethernet**) on the **CLICK CPU** modules as a **MODBUS TCP** server (slave), you don't need to use this instruction.

If you need more detailed information about the **MODBUS** protocol, we recommend to visit the following web site: www.MODBUS.org


Send

Com Port: CPU Port1
Protocol: MODBUS
COM Port Setup...

Sending Data Setup

Server(Slave) IP Address: . .
Server Port Number: 502 (1 to 65535)
Slave ID of Device: DS16 ... (0 to 255)
Modbus Function Code: 05 - Write Single Coil
Addressing Type: Modbus Addressing
Starting Slave Address: (000001 to 065535)
Starting Master Address: ...
Number of Bits: 1
Word Swap: OFF

Status Flags

Sending ...
Success ...
Error ...
Exception Response (Error Code) ...

About Error Flag
SC43:Out of Range

OK Cancel Help

1 COM Port: Select **Port 1** to use the **Ethernet** port.

2 Protocol:COM Port 1 only supports the **MODBUS TCP** protocol.

3 COM Port Setup: Click on this button to open the **Com Port Setup** window.

4 Server (Slave) IP Address: Enter the **IP Address** of the target server.

5 Server Port Number: Enter the **Port Number** of the target server. ‘**502**’ is the default.

6 Slave ID of Serial Device: Typically used for **Modbus TCP** to Modbus **RTU** gateways but some server devices may require a certain value. This value may be either a number or a DS Address.

7 MODBUS Function Code: Select one of the following **MODBUS** function codes:

- **01 – Read Coil Status**
- **02 – Read Input Status**
- **03 – Read Holding Registers**
- **04 – Read Input Registers**

8 Addressing Type: Select one of the following **MODBUS Addressing Types**:

Addressing Type	Explanation
MODBUS 984 Addressing	This **Addressing** is patterned after the **Modicon PLC Addressing**, which many devices support. **0******* are **Coils (Read/Write)** **4******* are **Holding Registers (Read/Write)**
MODBUS Hex Addressing	This **Addressing** is patterned after what the **MODBUS** protocol actually requests, which is simply a **Function Code +** an **Offset**.
CLICK Addressing	If the **MODBUS Slave** is the **CLICK PLC**, we recommend to use this **CLICK Addressing** because you can use the **Address Picker** to select the **Starting Slave Address**.

9 Starting Slave Address: Enter the **Slave Address** to start writing data to. Here are the valid **Slave addresses**. If you selected the **CLICK Addressing**, you can use the **Address Picker** by clicking the ⋯ icon.

MODBUS Addressing	Function Code			
	05	06	15	16
MODBUS 984	1 to 65535	400001 to 465535	1 to 65535	400001 to 465535
MODBUS Hex	0h to FFFEh	0h to FFFEh	0h to FFFEh	0h to FFFEh

CLICK	Y and C	CTD, DS, DH, SD, TD, YD & TXT	Y and C	CTD, DS, DD, DH, DF, SD, TD & TXT

10 Starting Master Address: Enter the **Starting Memory Address** of the **Master CLICK CPU** module to write the data to the **MODBUS Slave(s)**. You can use the **Address Picker** by clicking the icon.

Function Code			
05	06	15	16
X, Y, C, T, CT and SC	CTD, DS, DH, XD, YD, TD, SD and TXT	X, Y, C, T, CT and SC	CTD, DS, DH, XD, YD, SD, SD and TXT

11 Number of Bits: Enter the size of the data to write to the **MODBUS Slave(s)**.

12 Word Swap: When the **Starting Master Address** is a **DD**, **DF** or **CTD Memory Address**, this option is available. These **Memory Addresses** have **32** bit data length (**2 Words**), so each **Address** has data that can be written to two registers in the **MODBUS Slave**. You can **Swap** the order of those two word data in the **DD**, **DF** or **CTD Memory Addresses** to write to the **Slave**.

13 Sending: Assign a **C bit** as the **Sending Flag**. This **C bit** turns **ON** when the **Com Port** is **Sending** a write request to the **MODBUS Slave(s)** with this instruction. Please refer to the **Timing Chart**shown below.

Note: This **Sending Flag** is **OFF** when a **Receive** is reading data from a **MODBUS Slave** or another **Send** instruction is writing data to a **MODBUS Slave(s)**.

14 Success: Assign a **C bit** as the **Success Flag**. This **C bit** turns **ON** after the **Com Port** wrote data to the **MODBUS Slave(s)** successfully. It stays **ON** until this instruction is re-enabled. Please refer to the **Timing Chart**shown below.

Note: In the **Broadcast Mode (Slave ID = 0)**, there is no response from the **MODBUS Slaves**. The **Success bit** turns **ON** automatically after the **CLICK PLC** executed the **Send** instruction.

15 Error: Assign a **C bit** as the **Error Flag**. This **C bit** turns **ON** after the **Com Port** could not write data to the **MODBUS slave(s)** successfully. It stays **ON** until this instruction is re-enabled. Please refer to the **Timing Chart** shown below.

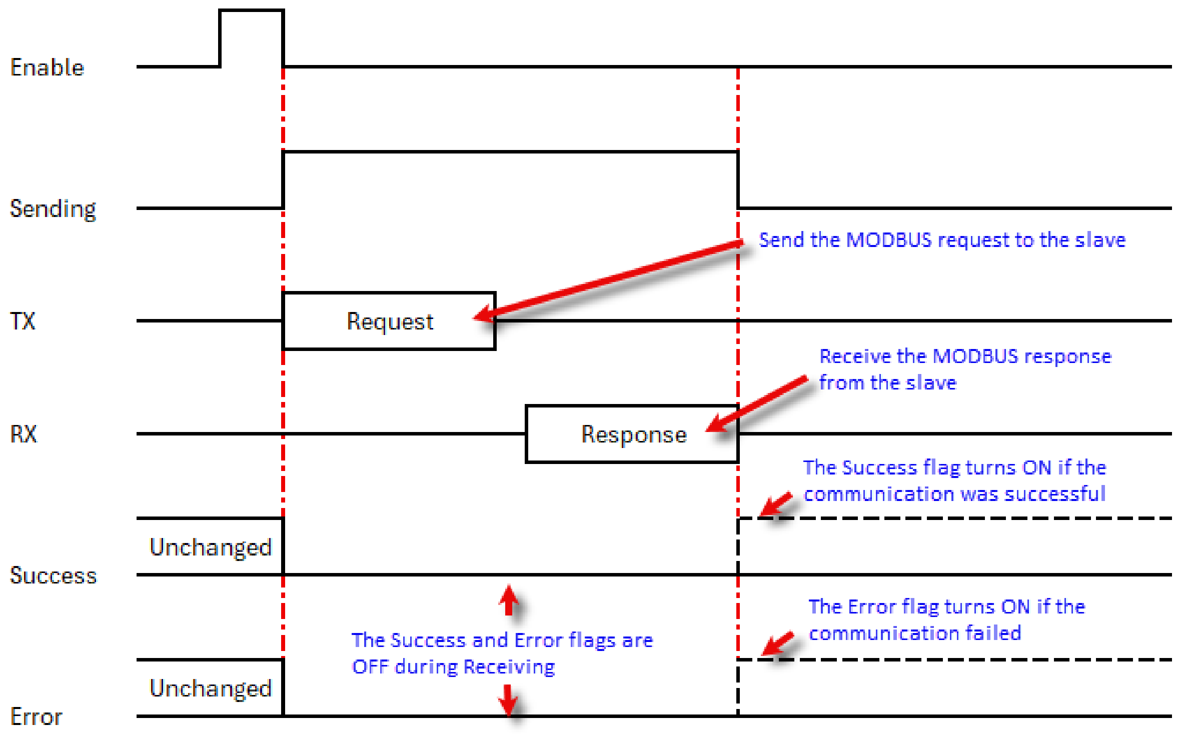
16 Exception Response (Error Code): Assign a **DS** or **DD Memory Address** to store the **Exception Response** from the **MODBUS Slave**.

Note: Each MODBUS Slave supports a different set of the Exception Response. Please refer to the user manual/specifications of each MODBUS Slave. If you are using a CLICK PLC as a MODBUS Slave, please refer to this topic for the Exception Response that the CLICK PLC supports.

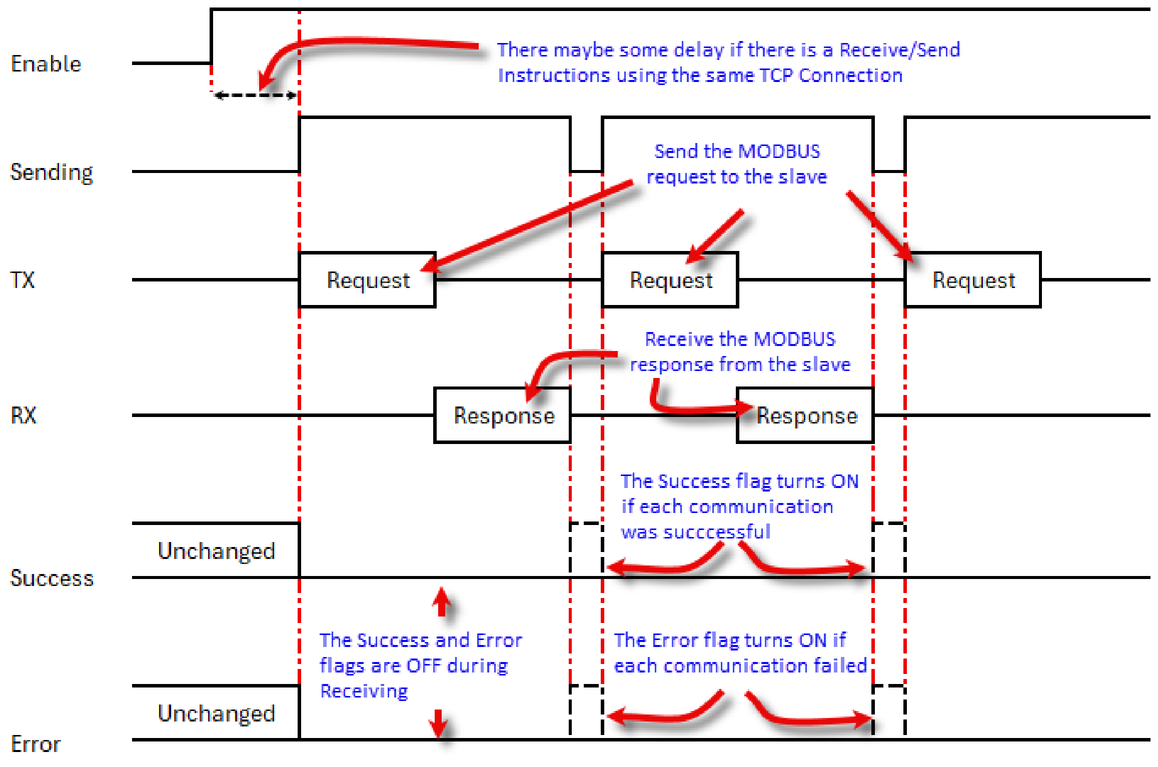
17 About Error Flag: If the Slave ID Is outside the range of 0 to 255, SC43: Out of Range will be set TRUE.

Timing Charts

Send Data to a MODBUS Slave Once


Enable
Sending
TX
RX
Success
Error
Request
Response
Unchanged
Unchanged
Send the MODBUS request to the slave
Receive the MODBUS response from the slave
The Success flag turns ON if the communication was successful
The Error flag turns ON if the communication failed
The Success and Error flags are OFF during Receiving

Send Data to a MODBUS Slave Continuously


Enable
Sending
TX
RX
Success
Error
There maybe some delay if there is a Receive/Send Instructions using the same TCP Connection
Send the MODBUS request to the slave
Request
Request
Request
Receive the MODBUS response from the slave
Response
Response
The Success flag turns ON if each communication was succcessful
Unchanged
The Success and Error flags are OFF during Receiving
The Error flag turns ON if each communication failed
Unchanged

Related Topics

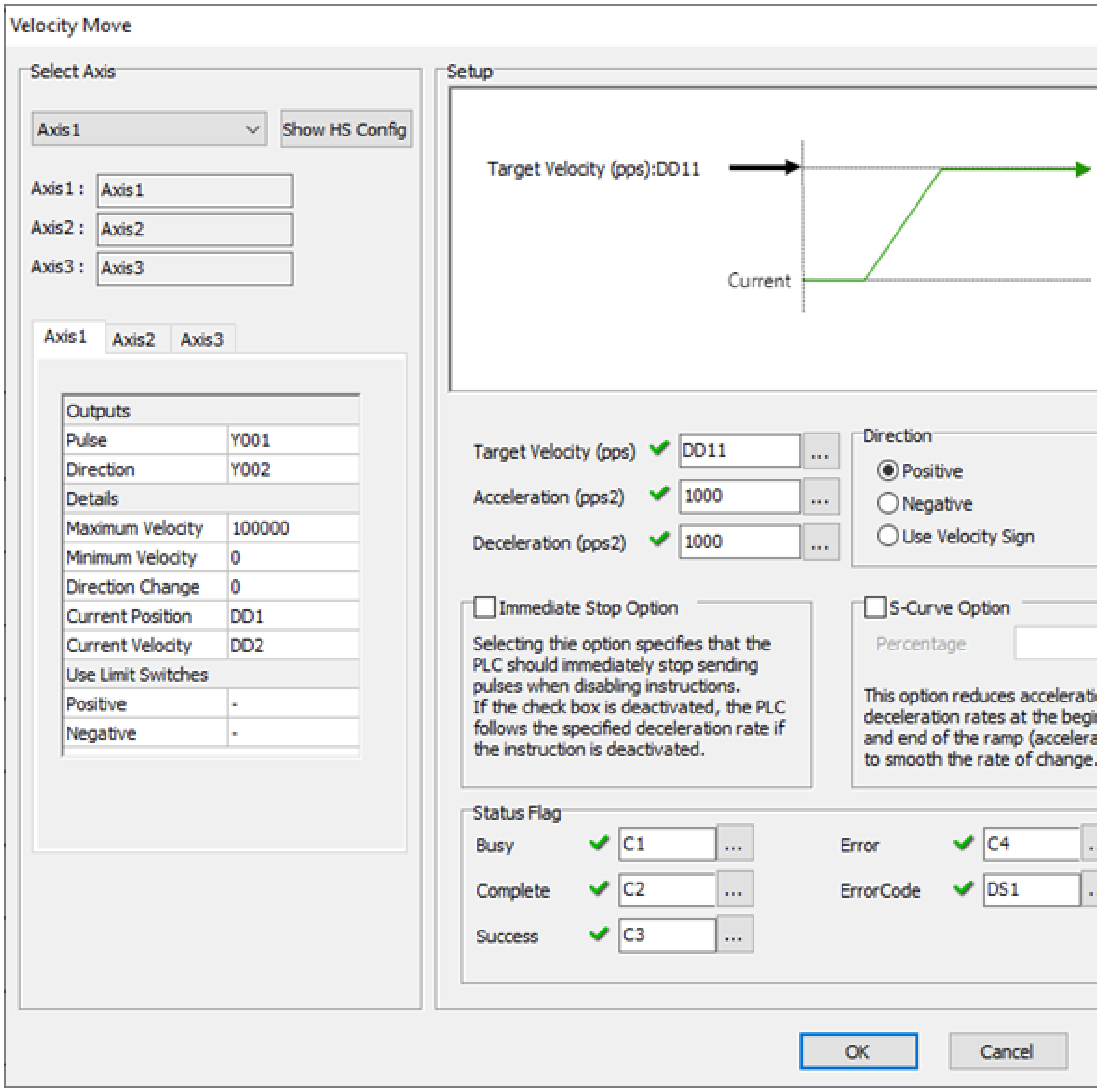
Topic: CL294

Topic: CL295

Velocity Move for PTO Axis

Purpose

The Velocity Move Instruction is used to continuously control the Axis by the Target Velocity (ignoring the current position). The Velocity can be any value between +/- of the configured Maximum Velocity. A velocity of Zero is also valid and will pause the Axis. The change in Velocity is affected by four settings: Minimum Velocity (set in Pulse Train Output configuration), Acceleration, Deceleration, and the S-Curve Option. The Target Velocity can be changed at any time during the Move.


Velocity Move

Select Axis

Axis1 | Show HS Config

Axis1 : Axis1
Axis2 : Axis2
Axis3 : Axis3

Axis1 | Axis2 | Axis3

Outputs	
Pulse	Y001
Direction	Y002
Details	
Maximum Velocity	100000
Minimum Velocity	0
Direction Change	0
Current Position	DD1
Current Velocity	DD2
Use Limit Switches	
Positive	-
Negative	-

Setup

Target Velocity (pps) ✓ DD11
Acceleration (pps2) ✓ 1000
Deceleration (pps2) ✓ 1000

Direction
[✓ Positive]
Negative
Use Velocity Sign

☐ Immediate Stop Option

Selecting thie option specifies that the PLC should immediately stop sending pulses when disabling instructions.
If the check box is deactivated, the PLC follows the specified deceleration rate if the instruction is deactivated.

☐ S-Curve Option

Percentage

This option reduces accelerati
deceleration rates at the begi
and end of the ramp (accelera
to smooth the rate of change.

Status Flag

Busy ✓	C1	Error ✓	C4
Complete ✓	C2	ErrorCode ✓	DS1
Success ✓	C3		

OK | Cancel

1 Select Axis: The PTO Axis must be created in the High Speed I/O Setup before placing the Velocity Move Instruction into ladder.

2 Axis Names: These are the Axis names as defined in the High Speed I/O Setup.

3 Axis Settings: This is a read-only view of the Axis settings from the High Speed I/O Setup.

4 Target Velocity: Constant or DD Register. The Axis will continuously follow this Velocity using the settings as configured by the rules configured for: Minimum Velocity, Maximum Velocity, Acceleration, Deceleration, and the S-Curve Option.

5 Acceleration: Constant or DD Register. The change of rate value used only during Acceleration (Pulses Per Second Squared).

6 Deceleration: Constant or DD Register. The change of rate value used only during Deceleration. (Pulses Per Second Squared)

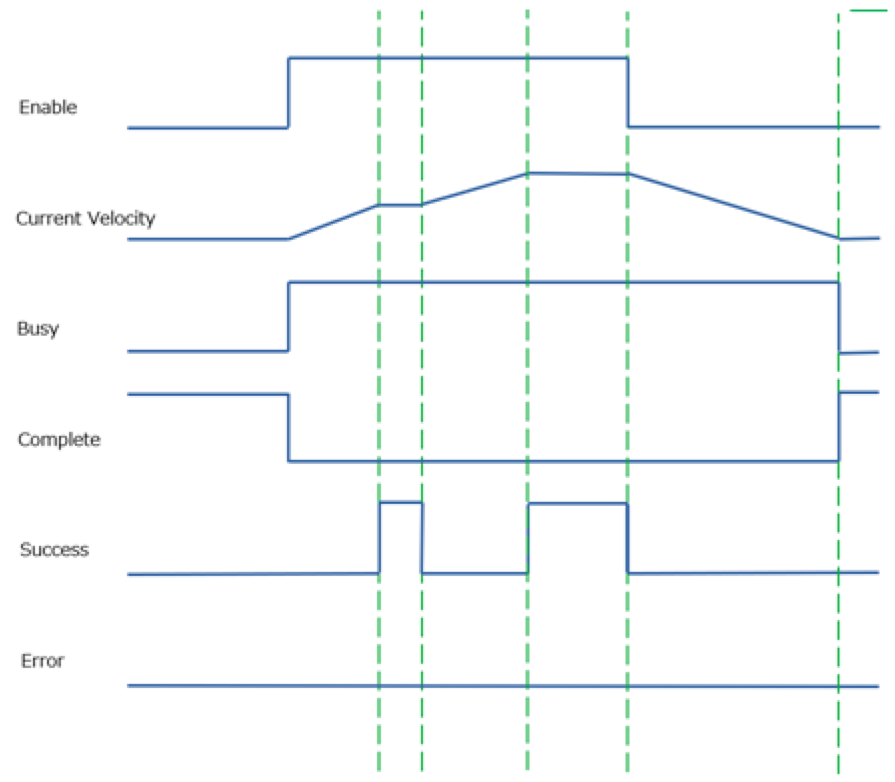
7 Direction: These settings control how the value in the Target Velocity will be interpreted.

- Positive: Ignore the Target Velocity Sign and always travel in the Positive Direction.
- Negative: Ignore the Target Velocity Sign and always travel in the Negative Direction.
- Use Velocity Sign: Travel in the Positive Direction for Positive Velocities, Travel in the Negative Direction for Negative Velocities.

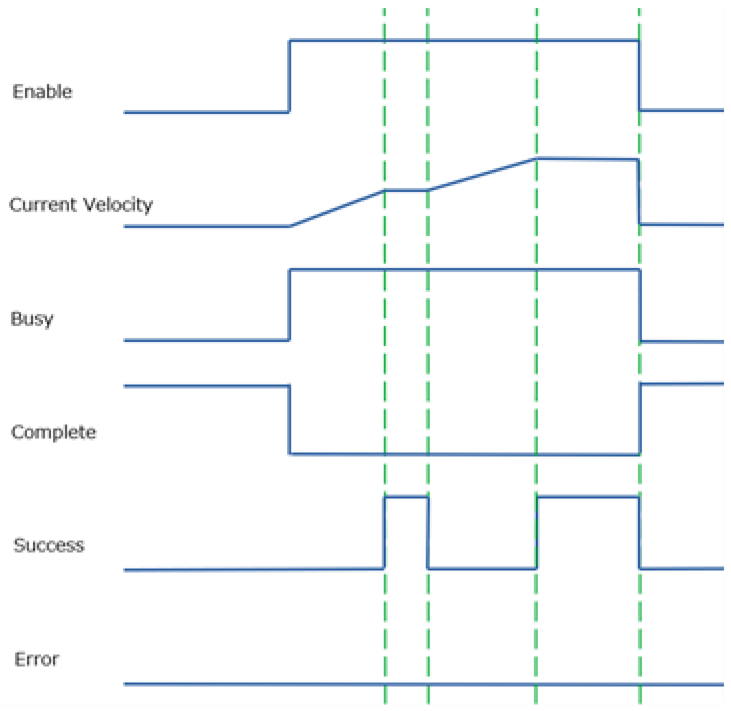
8 Immediate Stop Option: The behavior of an Aborted Move can be chosen.

- Enabled: Selecting this option specifies that the PLC should immediately stop sending pulses when disabling instructions. This will abort the Move abruptly and likely lose the Axis Home reference if traveling at high speeds.
- Disabled: If the check box is deactivated, the PLC follows the specified deceleration rate if the instruction is deactivated. The Axis will continue to follow its configured rules until reaching zero velocity.

Status Flag behavior with Immediate Stop Option Disabled

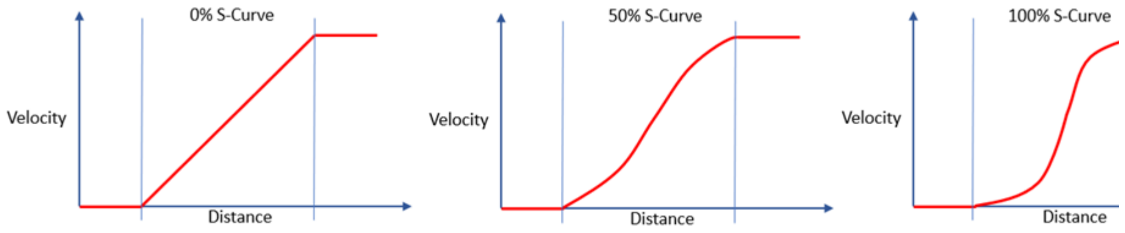


Status Flag behavior with Immediate Stop Option Enabled



9 S-Curve Option: This option smooths the velocity change to create non-linear transitions during Acceleration and Deceleration. The valid range is 1 to 100%.

Increasing the S-Curve Percentage does not increase the time during the Velocity change because the rate of change is increased. At 100% S-Curve Percentage, the maximum Rate of Change will be 200% of the specified Acceleration or Deceleration values. The limits of the actual system must be considered, and these parameters adjusted to prevent position loss due to inertia or friction loses.



10 Busy: This C bit turns ON while the instruction is in operation and controlling the Axis. Only a single instruction can be in control of each Axis during any moment of time. This bit turns OFF when the instruction is Complete by Success or by Error.

11 Complete: This C bit turns ON when the instruction is Complete by Success or by Error. It stays ON until the instruction is re-enabled.

12 Success: This C bit turns ON to indicate the Axis has matched the commanded Target Velocity. During Acceleration and Deceleration the bit will be OFF. If the Target Velocity is above the Maximum Velocity, the Axis will run at the Maximum Velocity and the Success Flag will not be ON during this situation.

13 Error: This C bit turns ON to indicate a failure of the Instruction or Axis. It stays ON until the instruction is re-enabled. The Value in the Error Code register will indicate the problem.

14 Error Code: Assign a DS or DD register to store the Numeric Error Code.

Note: The System Bits SC150-SC152 "_PTO_AxisX_Ready_Flag" provide status information.

Related Topics

PTO Error Codes
Pulse Train Output
Pulse Width Modulation

**Hom
e for**

PTO
Axis

Purpose

The Home Instruction is used to move the Axis from the Current Position to a known physical Position found by an Input Switch. The Home Instruction can only control a single Axis, but separate Home Instructions can occur simultaneously. The Home search can be configured to travel in the Positive or Negative Direction. Optionally the Current Position Register can be set when the instruction completes successfully. The settings (Initial Velocity, Creep Velocity, Acceleration, Deceleration) cannot be changed while the instruction is enabled.

There are 6 Modes of operation:

1) Move to Switch 1 and Creep to Switch 2 and Halt

2) Move to Switch 1 and Halt

3) Move to Switch 1 and Deceleration

4) Move to Switch 1 and Return to Switch 1

5) Move to Switch 1 and Decelerate to a Stopping distance (Registration type move)

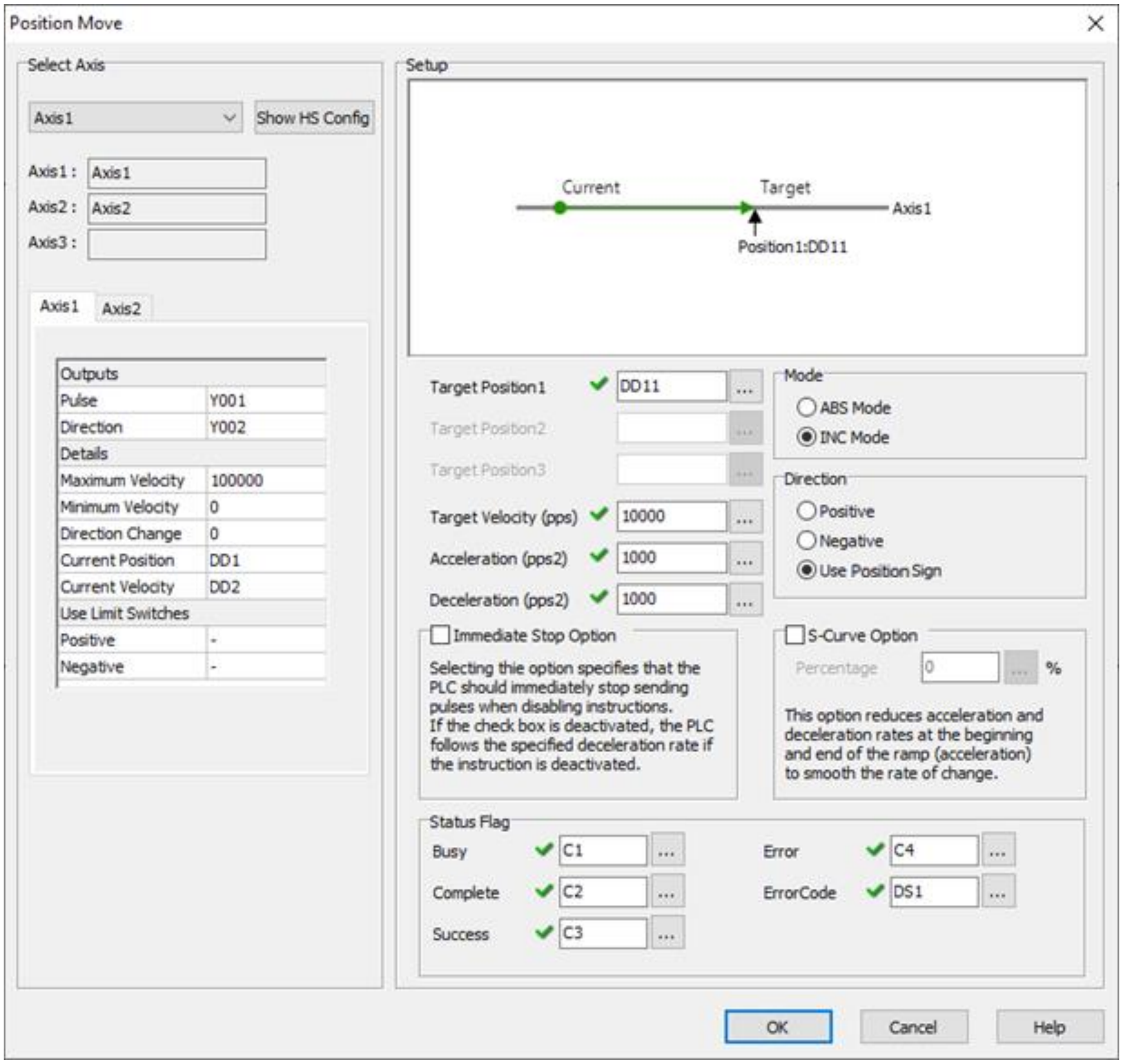
6) Set Current Position To Immediate Value

Topic: CL296

Position Move for PTO Axis

Purpose

The Position Move Instruction is used to move the Axis from the Current Position to a new Target Position. The Target Position can either be an Absolute(ABS) Position, or an Incremental(INC) Position. The Move can include a Single Axis, or up to 3-Axis using Interpolation. The motion profile is Trapezoidal and includes 3 segments: Acceleration, Target Velocity, Deceleration. The settings (Target Position, Target Velocity, Acceleration, Deceleration) cannot be changed while the instruction is enabled.



Position Move

Select Axis

Axis1 | Show HS Config

Axis1 : Axis1
Axis2 : Axis2
Axis3 :

Axis1 | Axis2

Outputs	
Pulse	Y001
Direction	Y002
Details	
Maximum Velocity	100000
Minimum Velocity	0
Direction Change	0
Current Position	DD1
Current Velocity	DD2
Use Limit Switches	
Positive	-
Negative	-

Setup

Target Position1	✓ DD11
Target Position2	
Target Position3	
Target Velocity (pps)	✓ 10000
Acceleration (pps2)	✓ 1000
Deceleration (pps2)	✓ 1000

Mode: ABS Mode [✓ INC Mode]

Direction: Positive Negative [✓ Use Position Sign]

☐ Immediate Stop Option

Selecting thie option specifies that the PLC should immediately stop sending pulses when disabling instructions.
If the check box is deactivated, the PLC follows the specified deceleration rate if the instruction is deactivated.

☐ S-Curve Option

Percentage 0 %

This option reduces acceleration and deceleration rates at the beginning and end of the ramp (acceleration) to smooth the rate of change.

Status Flag

Busy	✓ C1	Error	✓ C4
Complete	✓ C2	ErrorCode	✓ DS1
Success	✓ C3		

1 Select Axis: The PTO Axis must be created in the High Speed I/O Setup before placing the Position Move Instruction into ladder.

2 Axis Names: These are the Axis names as defined in the High Speed I/O Setup.

3 Axis Settings: This is a read-only view of the Axis settings from the High Speed I/O Setup.

4 Target Position: Constant or DD Register. The value is either ABS or INC Mode. When using an Interpolated mode, each Axis has a separate Target Position.

5 Target Velocity: Constant or DD Register. For a single axis, the Axis will attempt to run at this speed, but is controlled by the rules configured for: Minimum Velocity, Maximum Velocity, Acceleration, Deceleration.

If Multiple axis are selected, the combined movement will be along a vector. The target velocity will be the combined axis' vector velocity.

6 Acceleration: Constant or DD Register. The change of rate value used only during Acceleration (Pulses Per Second Squared).

If Multiple axis are selected, the combined movement will be along a vector. The target acceleration will be the combined acceleration on the vector.

7 Deceleration: Constant or DD Register. The change of rate value used only during Deceleration (Pulses Per Second Squared).

If Multiple axis are selected, the combined movement will be along a vector. The target deceleration will be the combined deceleration along the vector.

8 Mode:

- **ABS**: Absolute Mode moves the axis to an absolute position relative to the Home position Ignore the Target Position Sign and always travel in the Positive Direction.
- **INC**: Incremental Mode moves the axis to position relative to its Current position.

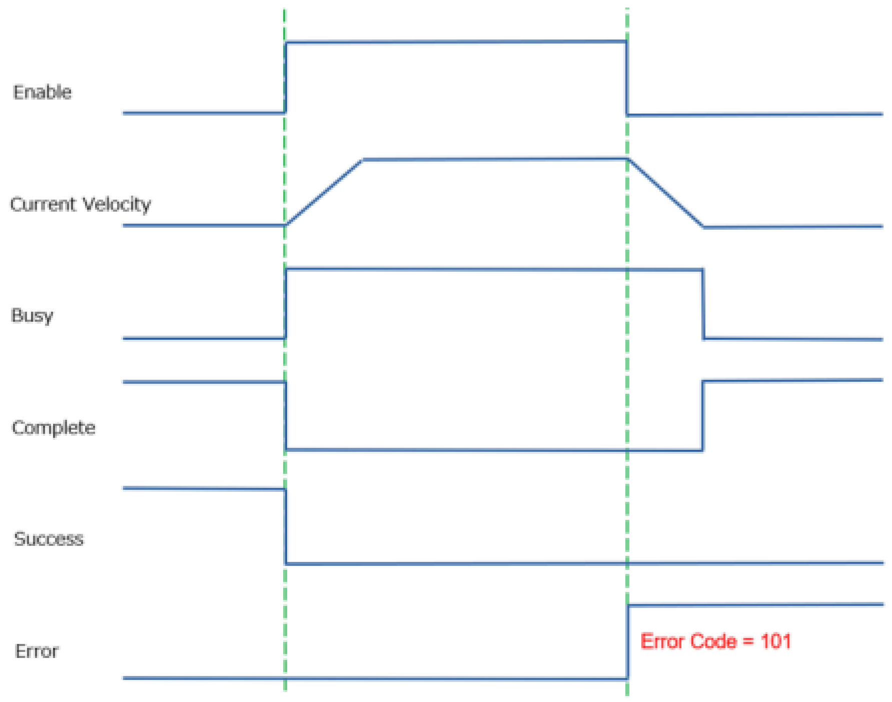
9 Direction: These settings control how the value in the Target Position will be interpreted.

- **Positive:** Ignore the Target Position Sign and always travel in the Positive Direction.
- **Negative:** Ignore the Target Position Sign and always travel in the Negative Direction.
- **Use Velocity Sign:** Travel in the Positive Direction for Positive Positions, Travel in the Negative Direction for Negative Positions.

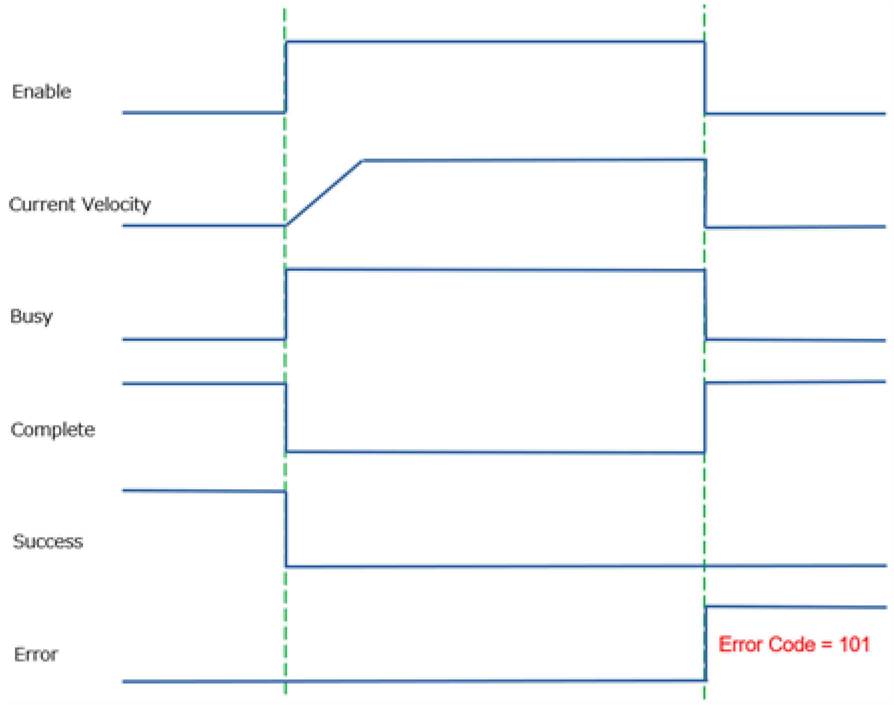
10 Immediate Stop Option: The behavior of an Aborted Move can be chosen.

- **Enabled:** Selecting this option specifies that the PLC should immediately stop sending pulses when disabling instructions. This will abort the Move abruptly and likely lose the Axis Home reference if travelling at high speeds.
- **Disabled:** If the check box is deactivated, the PLC follows the specified deceleration rate if the instruction is deactivated. The Axis will continue to follow its configured rules until reaching zero velocity.

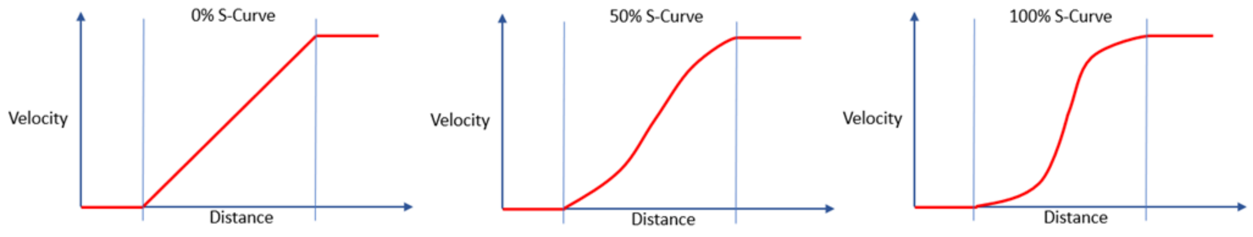
Status Flag behavior with Immediate Stop Option Disabled



Status Flag behavior with Immediate Stop Option Enabled



11 S-Curve Option: This option smooths the velocity change to create non-linear transitions during Acceleration and Deceleration. The valid range is 1 to 100%. Increasing the S-Curve Percentage does not increase the time during the Velocity change because the rate of change is increased. At 100% S-Curve Percentage, the maximum Rate of Change will be 200% of the specified Acceleration or Deceleration values. The limits of the actual system must be considered, and these parameters adjusted to prevent position loss due to inertia or friction loses.



12 Busy: This C bit turns ON while the instruction is in operation and controlling the Axis. Only a single instruction can be in control of each Axis during any moment of time. This bit turns OFF when the instruction is Complete by Success or by Error.

13 Complete: This C bit turns ON when the instruction is Complete by Success or by Error. It stays ON until the instruction is re-enabled.

14 Success: This C bit turns ON to indicate the Axis has reached the commanded Target Position.

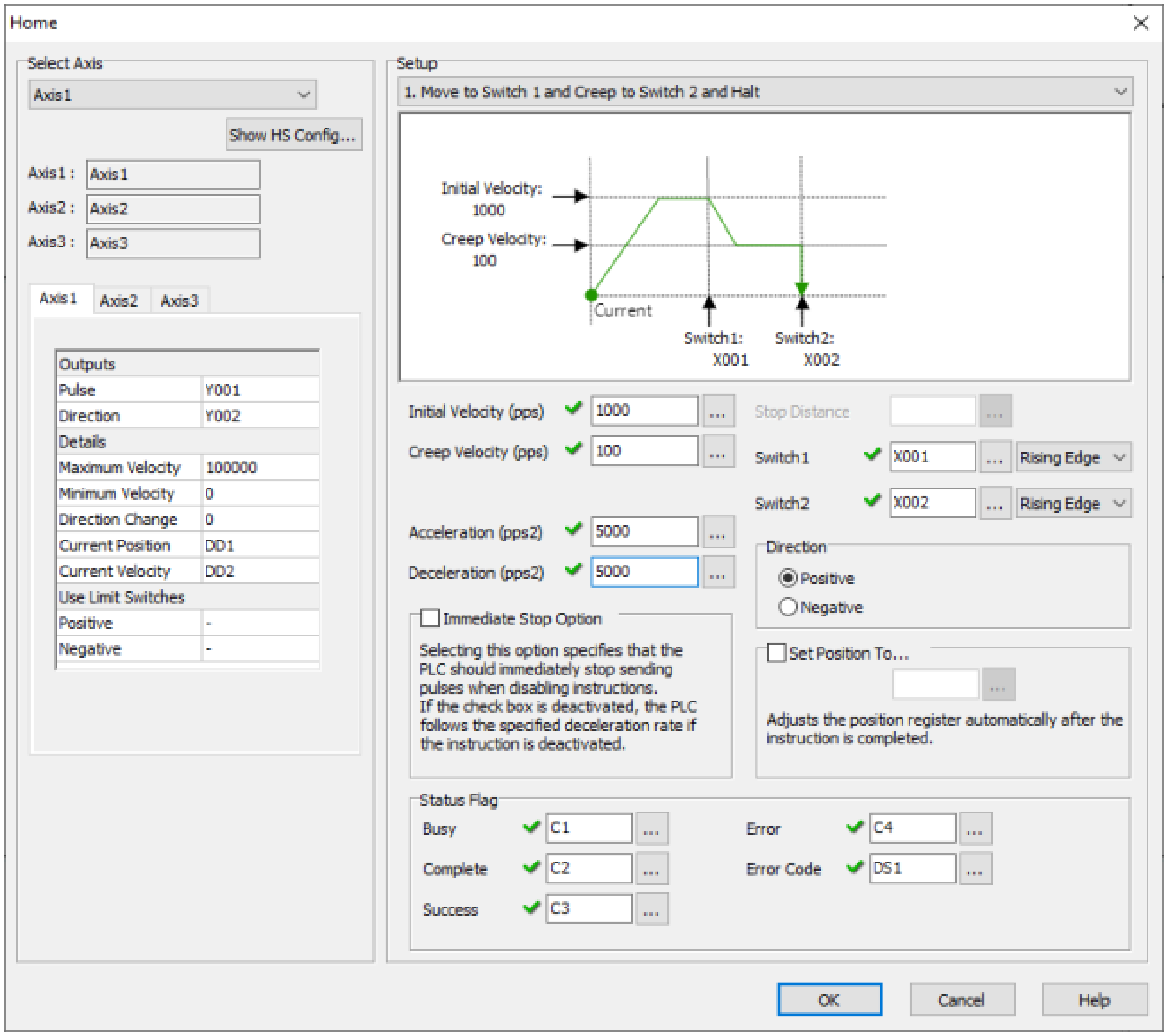
15 Error: This C bit turns ON to indicate a failure of the Instruction or Axis. It stays ON until the instruction is re-enabled. The Value in the Error Code register will indicate the problem.

16 Error Code: Assign a DS or DD register to store the Numeric Error Code.

Note: The System Bits SC150-SC152 “_PTO_AxisX_Ready_Flag” provide status information.

Related Topics

PTO Error Codes
Home for PTO Axis
Velocity Move for PTO Axis
Pulse Train Output
Pulse Width Modulation


Home

Select Axis: Axis1

Show HS Config...

Axis1 :	Axis1
Axis2 :	Axis2
Axis3 :	Axis3

Axis1 | Axis2 | Axis3

Outputs	
Pulse	Y001
Direction	Y002
Details	
Maximum Velocity	100000
Minimum Velocity	0
Direction Change	0
Current Position	DD1
Current Velocity	DD2
Use Limit Switches	
Positive	-
Negative	-

Setup: 1. Move to Switch 1 and Creep to Switch 2 and Halt

Initial Velocity (pps)	✓ 1000	Stop Distance	
Creep Velocity (pps)	✓ 100	Switch1	✓ X001 Rising Edge
		Switch2	✓ X002 Rising Edge
Acceleration (pps2)	✓ 5000		
Deceleration (pps2)	✓ 5000		

Direction: [✓ Positive] Negative

☐ Immediate Stop Option

Selecting this option specifies that the PLC should immediately stop sending pulses when disabling instructions.
If the check box is deactivated, the PLC follows the specified deceleration rate if the instruction is deactivated.

☐ Set Position To...

Adjusts the position register automatically after the instruction is completed.

Status Flag

Busy	✓ C1	Error	✓ C4
Complete	✓ C2	Error Code	✓ DS1
Success	✓ C3		

OK | Cancel | Help

Select Axis: The PTO Axis must be created in the High Speed I/O Setup before placing the Home Instruction into ladder.

Axis Names: These are the Axis names as defined in the High Speed I/O Setup.

Axis Settings: This is a read-only view of the Axis settings from the High Speed I/O Setup.

Initial Velocity: Constant or DD Register. The speed at which the Home search will begin.

Creep Velocity: Constant or DD Register. Used with Mode 1 and Mode 4. A slower velocity used to approach the second Switch.

Acceleration: Constant or DD Register. The change of rate value used only during Acceleration. (Pulses Per Second Squared)

Deceleration: Constant or DD Register. The change of rate value used only during Deceleration. (Pulses Per Second Squared)

Direction: This setting selects the Direction of Travel for the Home search.

Stop Distance: Constant or DD Register. This setting is only used with Mode 5.

Switch 1 and 2: Must be an external Input X bit. For their specific function in each Mode, see the Mode Descriptions and diagrams below.

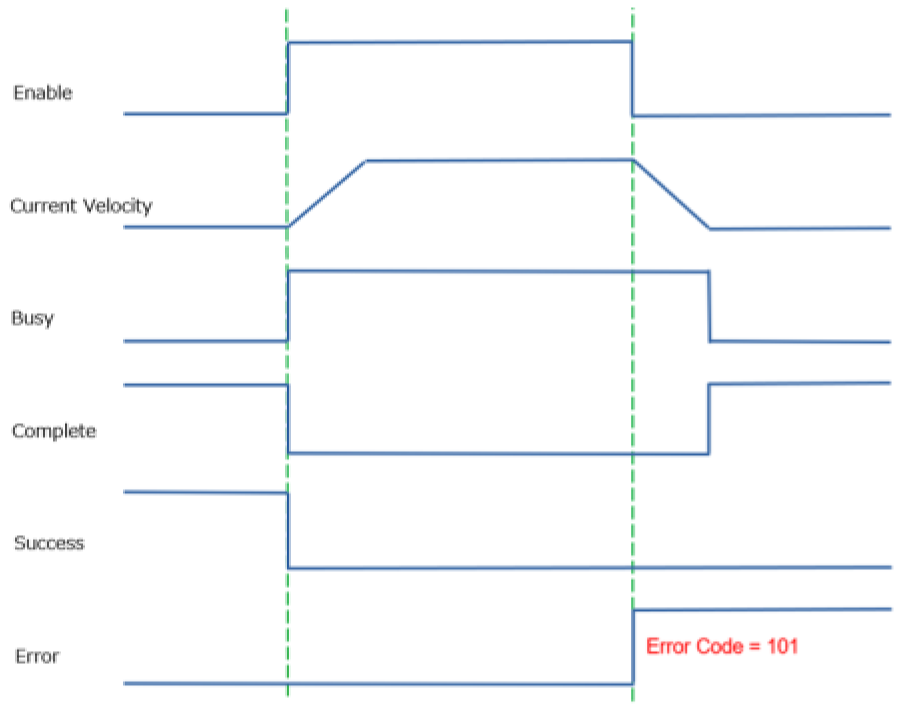
Rising Edge or Falling Edge: Select the Active Edge for the Switch Inputs. Normally Open switches will use Rising Edge, Normally Closed switches will use Falling Edge.

Immediate Stop Option: The behavior of an Aborted Move can be chosen.

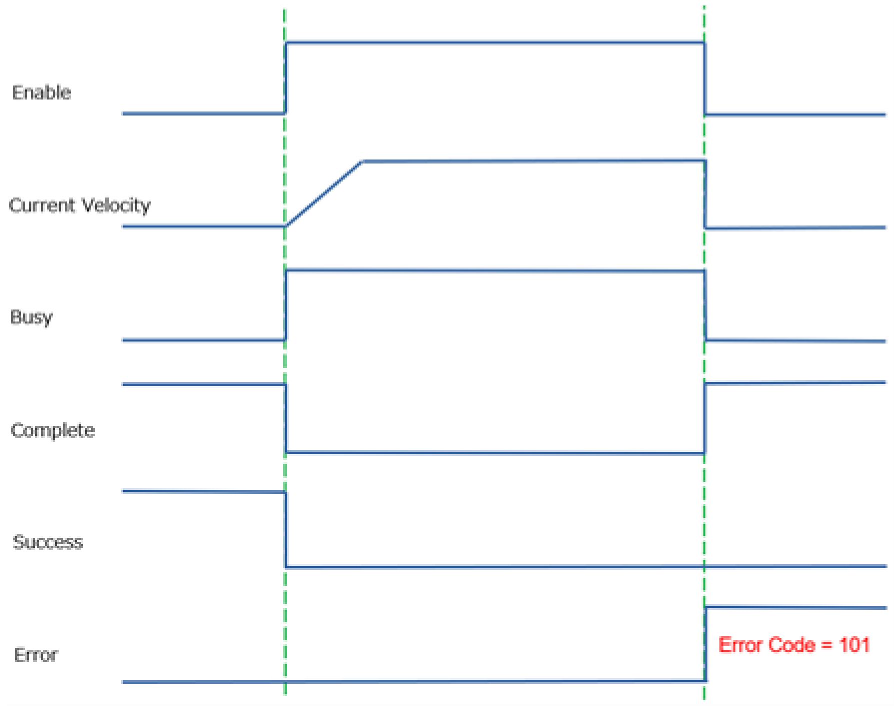
- Enabled: Selecting this option specifies that the PLC should immediately stop sending pulses when disabling instructions. This will abort the Move abruptly and likely lose the Axis Home reference if traveling at high speeds.
- Disabled: If the check box is deactivated, the PLC follows the specified deceleration rate if the instruction is deactivated. The Axis will decelerate until reaching zero velocity. During the Abort deceleration the Switch Inputs are ignored.

Set Position To: When enabled, the Current Position Register is set to the Constant or DD Register value when the instruction is completed.

Status Flag behavior with Immediate Stop Option Disabled



Status Flag behavior with Immediate Stop Option Enabled



Busy: This C bit turns ON while the instruction is in operation and controlling the Axis. Only a single instruction can be in control of each Axis during any moment of time. This bit turns OFF when the instruction is Complete by Success or by Error.

Complete: This C bit turns ON when the instruction is Complete by Success or by Error. It stays ON until the instruction is re-enabled.

Success: This C bit turns ON to indicate the Axis has reached the commanded Target Position.

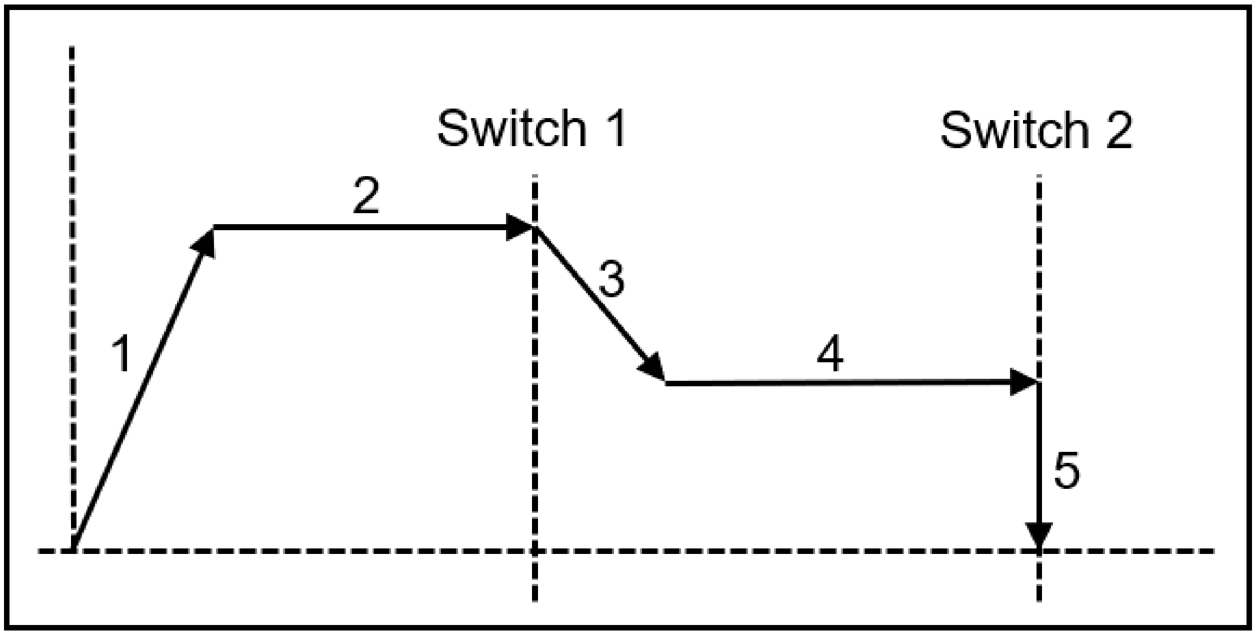
Error: This C bit turns ON to indicate a failure of the Instruction or Axis. It stays ON until the instruction is re-enabled. The Value in the Error Code register will indicate the problem.

Error Code: Assign a DS or DD register to store the Numeric Error Code.

Note: The System Bits SC150-SC152 "_PTO_AxisX_Ready_Flag" provide status information.

Home Mode1 (Motion Path)

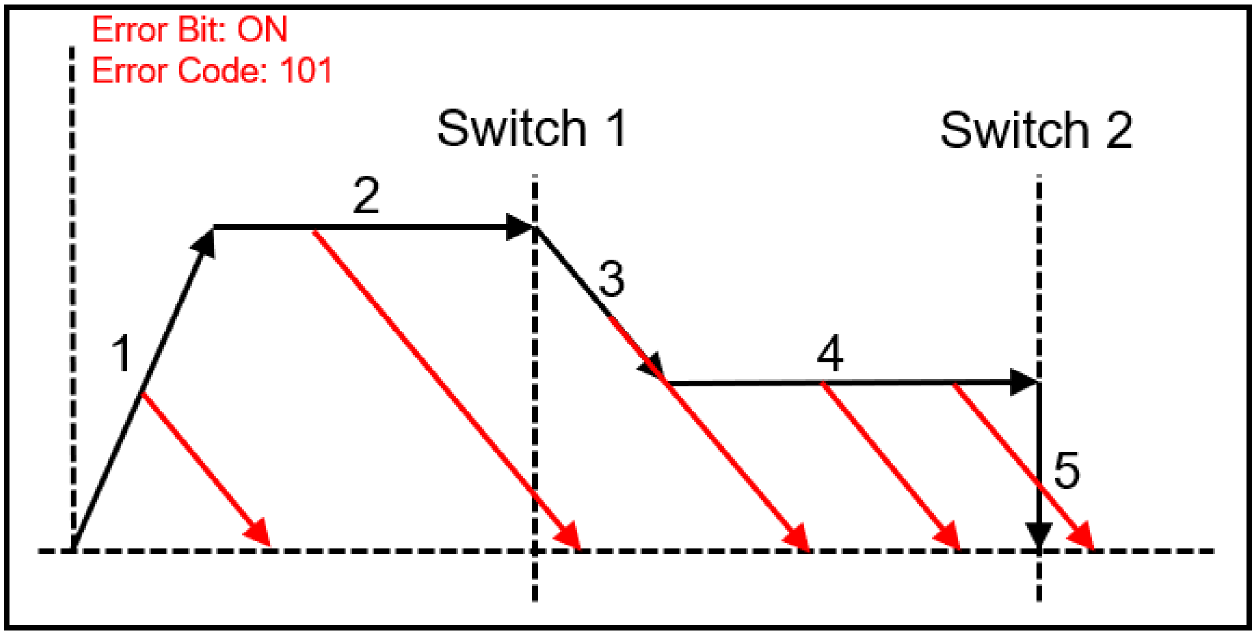
1. Accelerate to the speed of Initial Velocity.
2. Move at Initial Velocity until Switich1 turns ON.
3. Decelerate to Creep velocity.
4. Moves at Creep speed until Switch2 is turned ON.
5. Immediately stops when it reaches Switch2.



Home Mode1 (Disable Instruction)

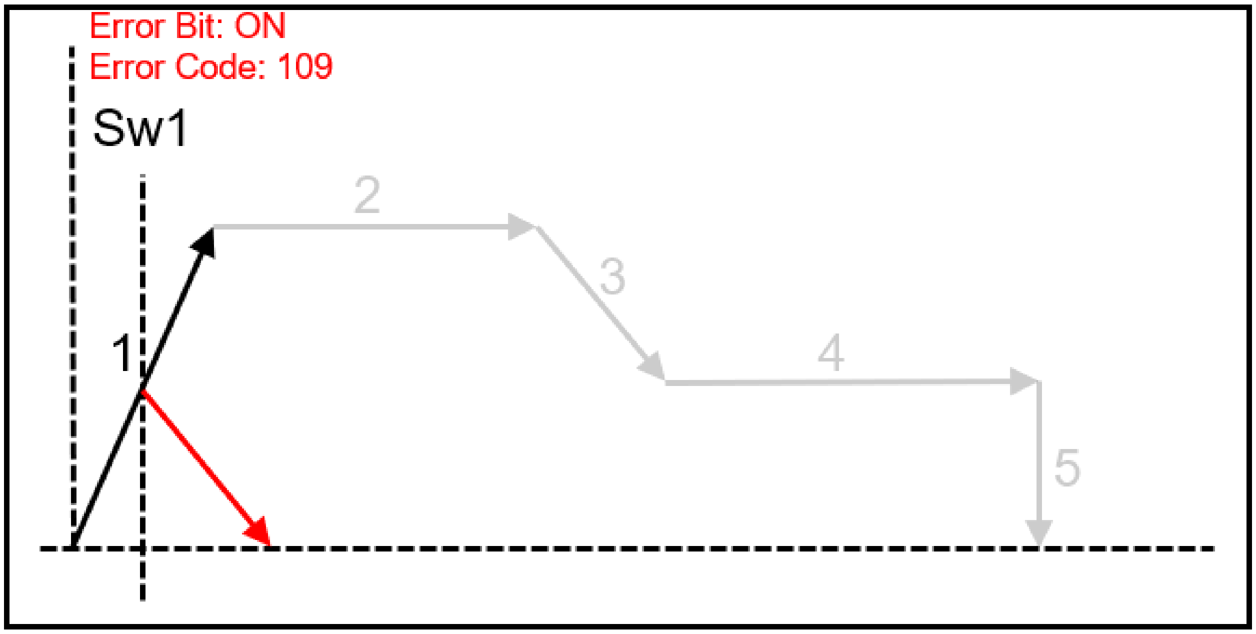
- Disabling during any segment will cause the Error Bit to turn ON, and the Error Code 101.

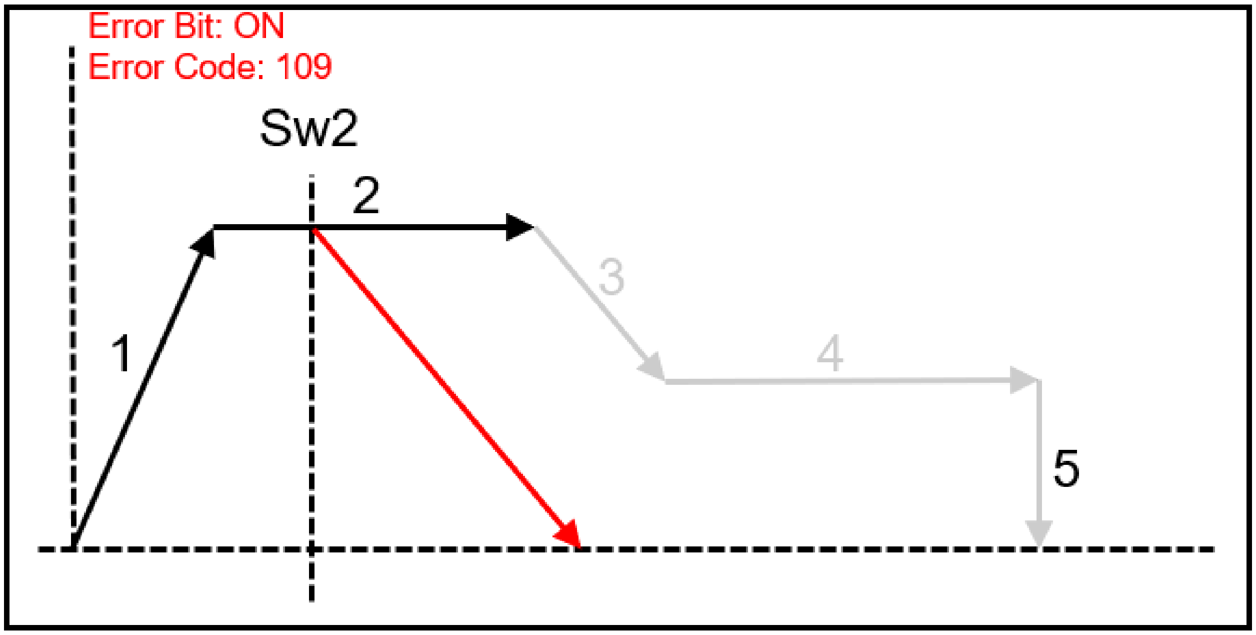
- The Axis will use the Deceleration Rate to bring the Axis to a Stop.
- Switch inputs are ignored after the instruction is disabled.

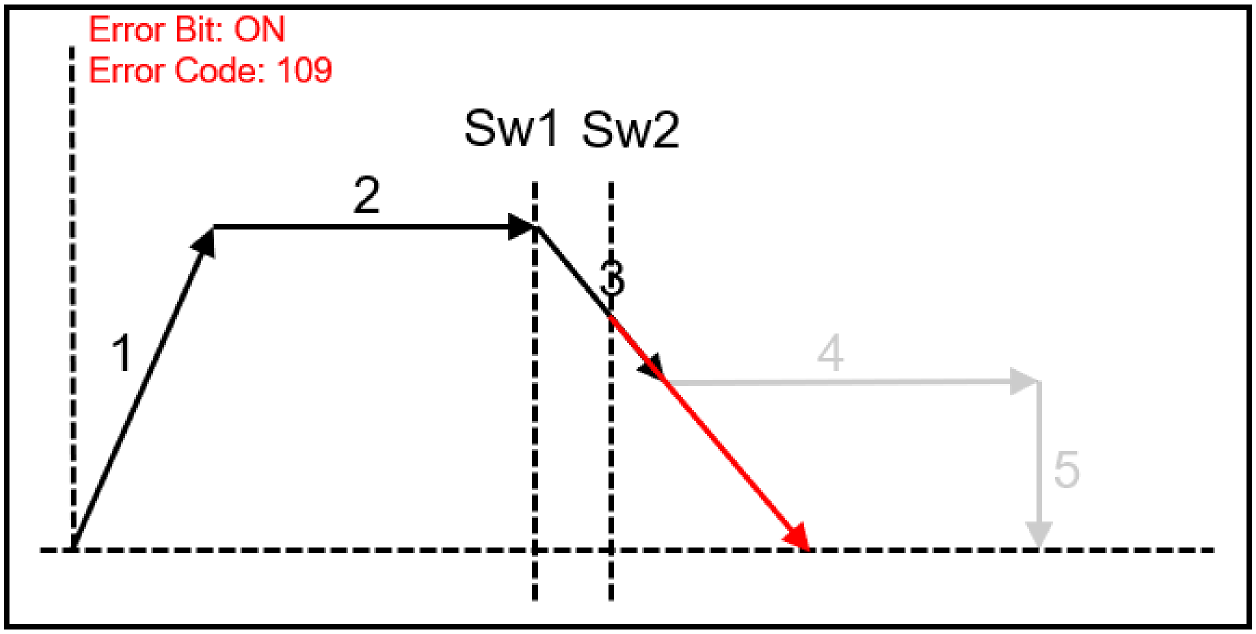


Home Mode1 (Switch Events)

- When Switch1 is turned on during acceleration, the Error Bit is turned ON and Error Code 109 is set.
- If Switch2 is turned on before Switch1, the Error Bit is turned ON and Error Code 109 is set.
- If Switch2 is turned on during deceleration, the Error Bit is turned ON and Error Code 109 is set.
- Switch1 cannot be shared with Limit Switch.
- Switch2 can be shared with Limit Switch.

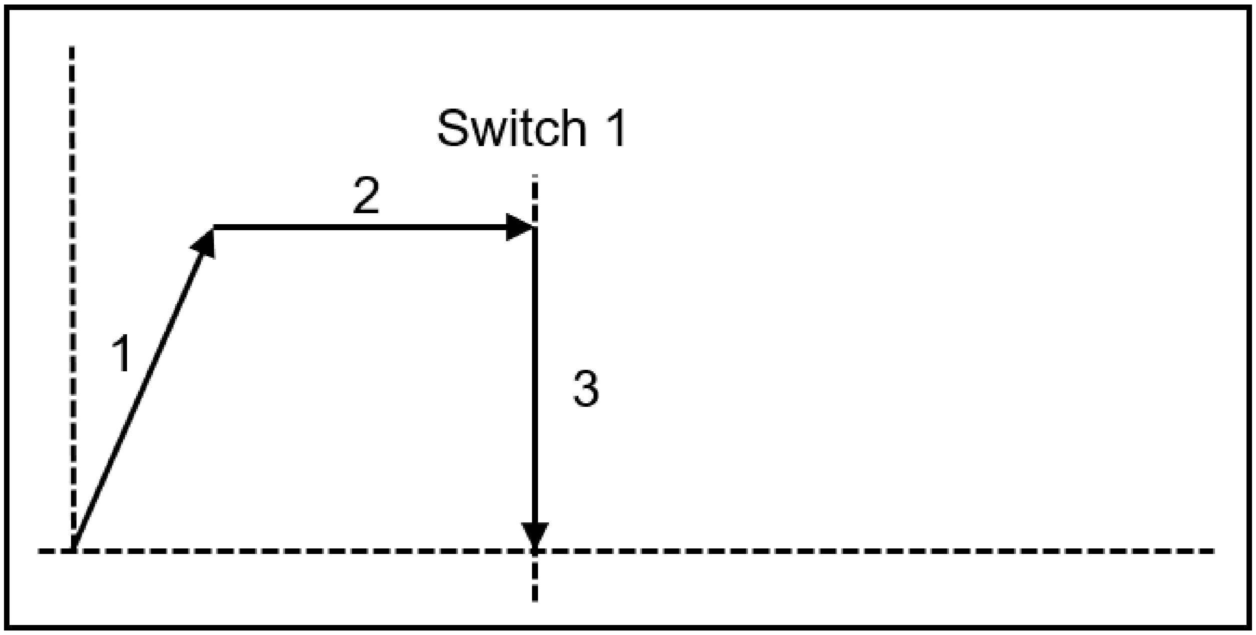

Error Bit: ON
Error Code: 109
Sw1
1
2
3
4
5


Error Bit: ON
Error Code: 109
Sw2
1
2
3
4
5



Home Mode2 (Motion Path)

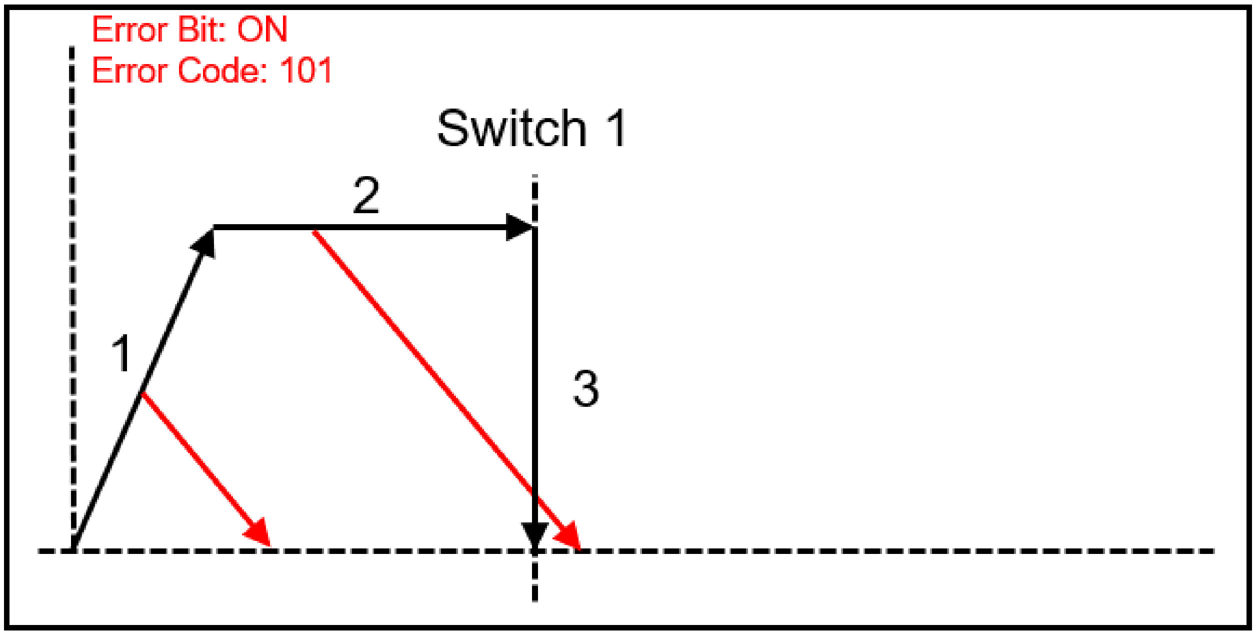
1. Accelerate to the speed of Initial Velocity.
2. Move at Initial Velocity until Switich1 turns ON.
3. Immediately stops when it reaches Switch1.



Home Mode2 (Disable Instruction)

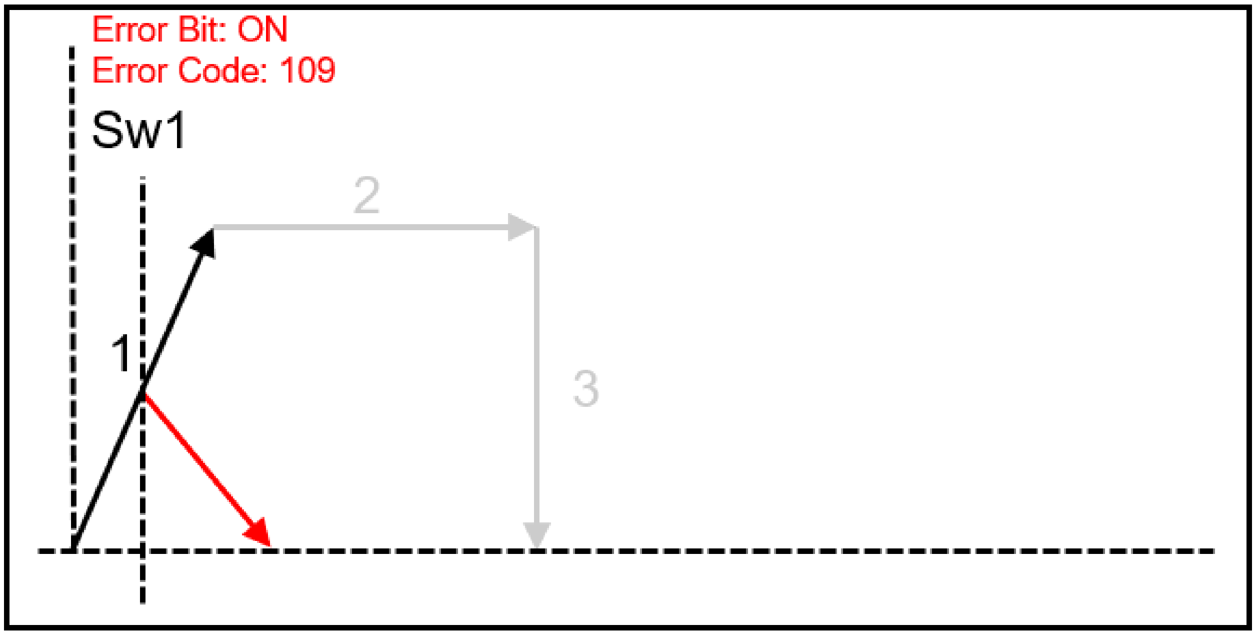
- Disabling during any segment will cause the Error Bit to turn ON, and the Error Code 101.

- The Axis will use the Deceleration Rate to bring the Axis to a Stop.
- Switch inputs are ignored after the instruction is disabled.



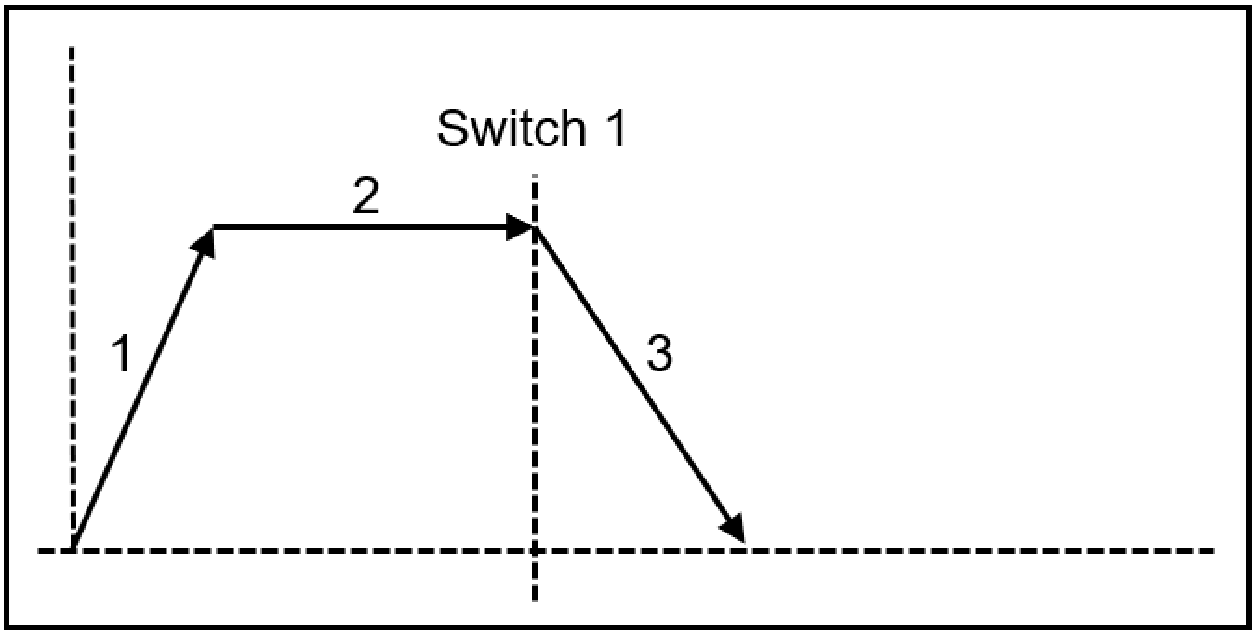
Home Mode2 (Switch Events)

- When Switch1 is turned on during acceleration, the Error Bit is turned ON and Error Code 109 is set.
- Switch1 can be shared with Limit Switch.



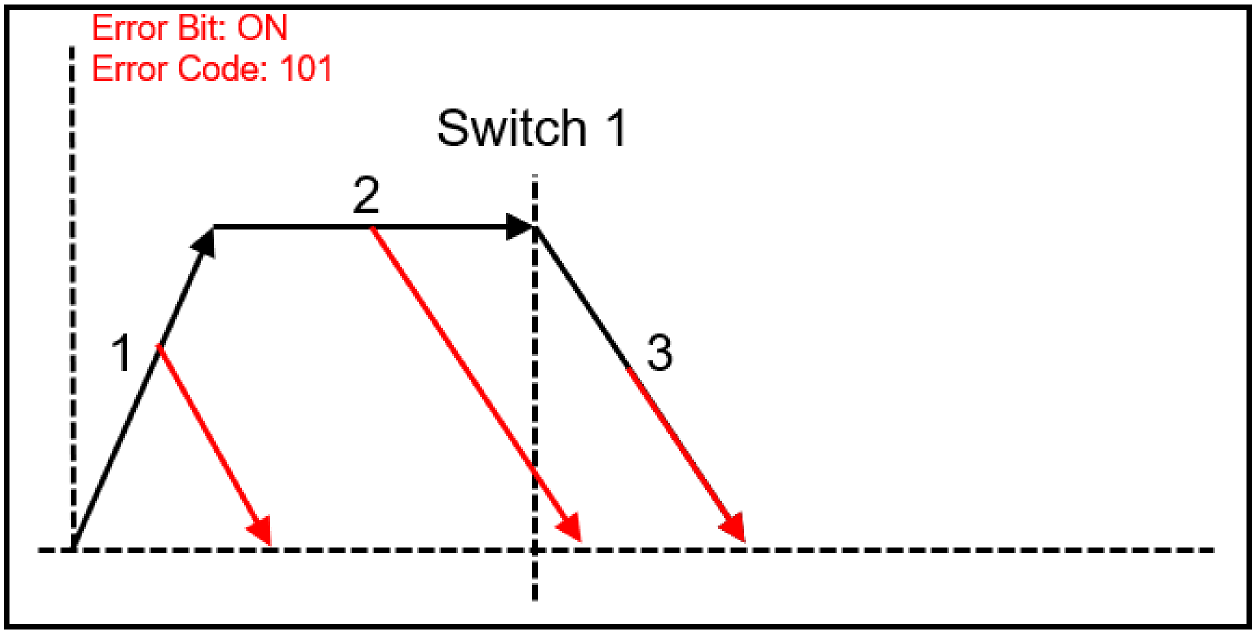
Home Mode3 (Motion Path)

1. Accelerate to the speed of Initial Velocity.
2. Move at Initial Velocity until Switich1 turns ON.
3. Decelerate according to the Deceleration setting and stop.



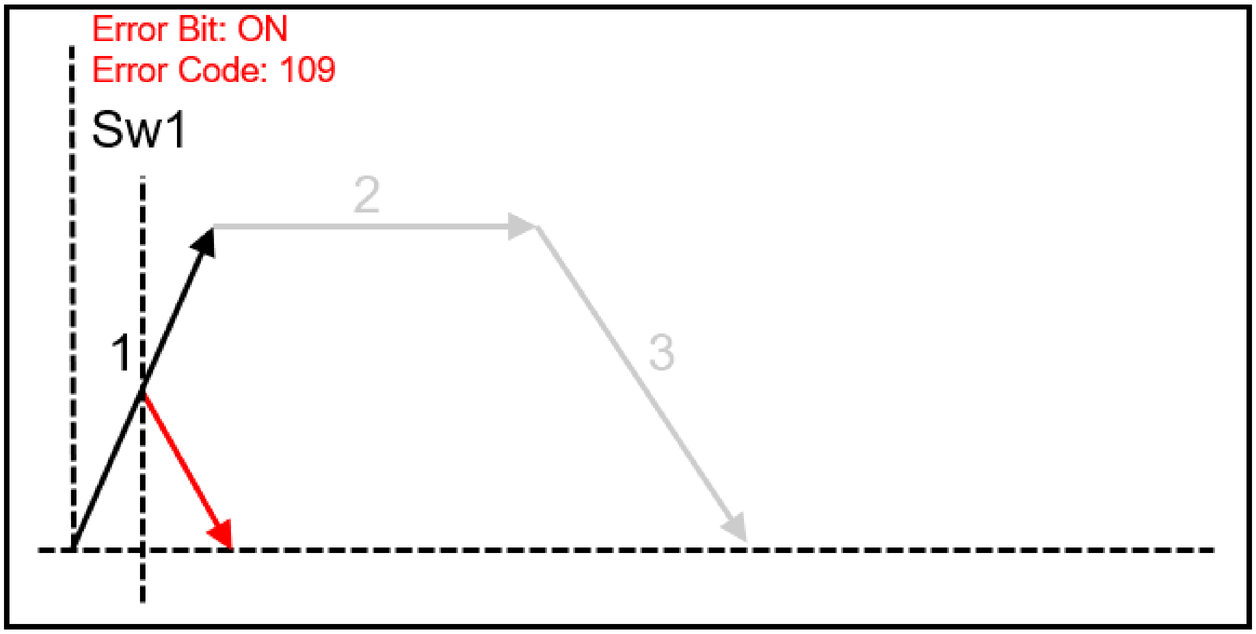
Home Mode3 (Disable Instruction)

- Disabling during any segment will cause the Error Bit to turn ON, and the Error Code 101.
- The Axis will use the Deceleration Rate to bring the Axis to a Stop.
- Switch inputs are ignored after the instruction is disabled.



Home Mode3 (Switch Events)

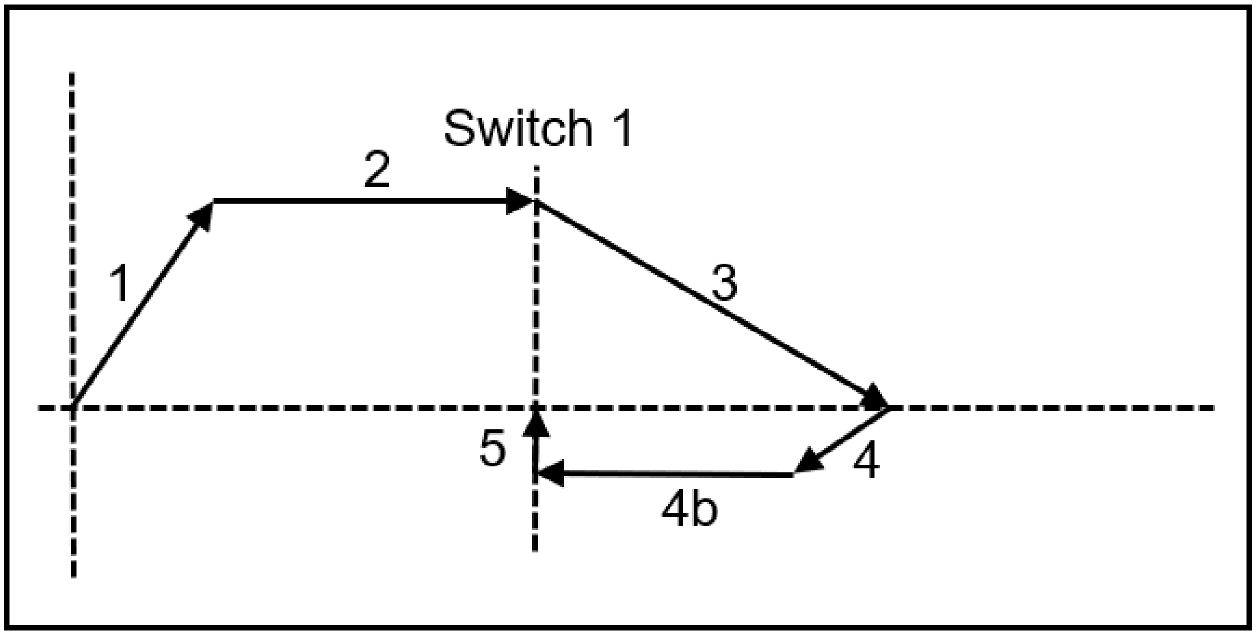
- When Switch1 is turned on during acceleration, the Error Bit is turned ON and Error Code 109 is set.
- Switch1 cannot be shared with Limit Switch.



Home Mode4 (Motion Path)

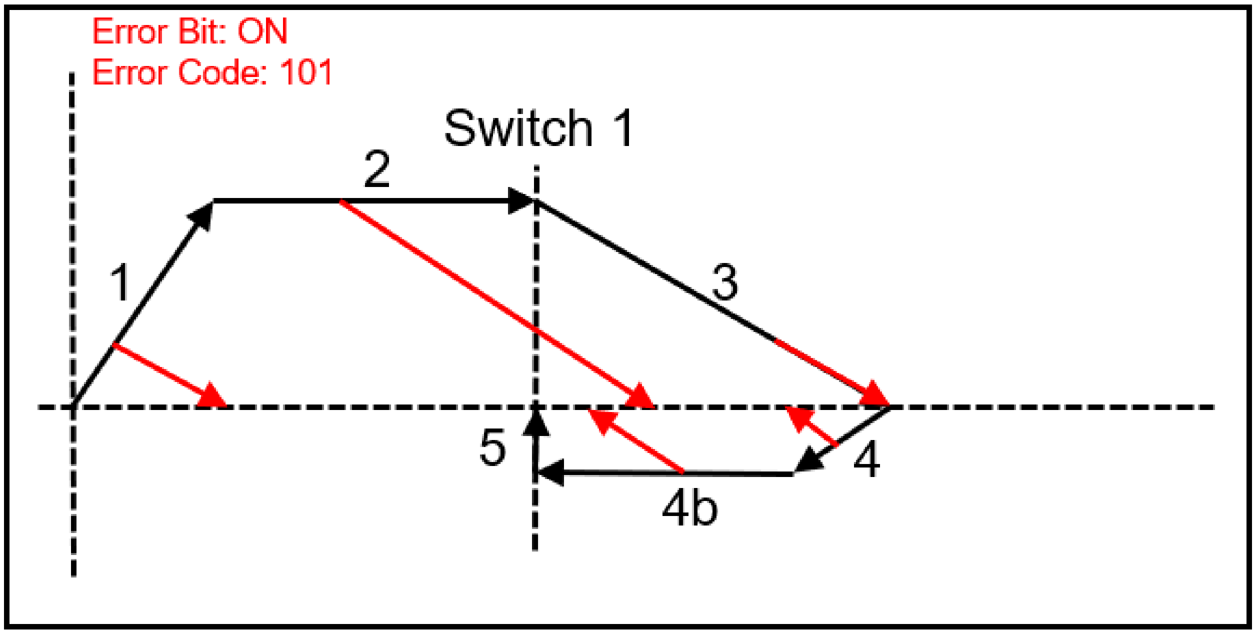
1. Accelerate to the speed of Initial Velocity.
2. Move at Initial Velocity until Switich1 turns ON.

3. Decelerate to a stop.
4. Accelerate to Creep Velocity towards Switch1.
5. Immediately stops when it reaches Switch1.



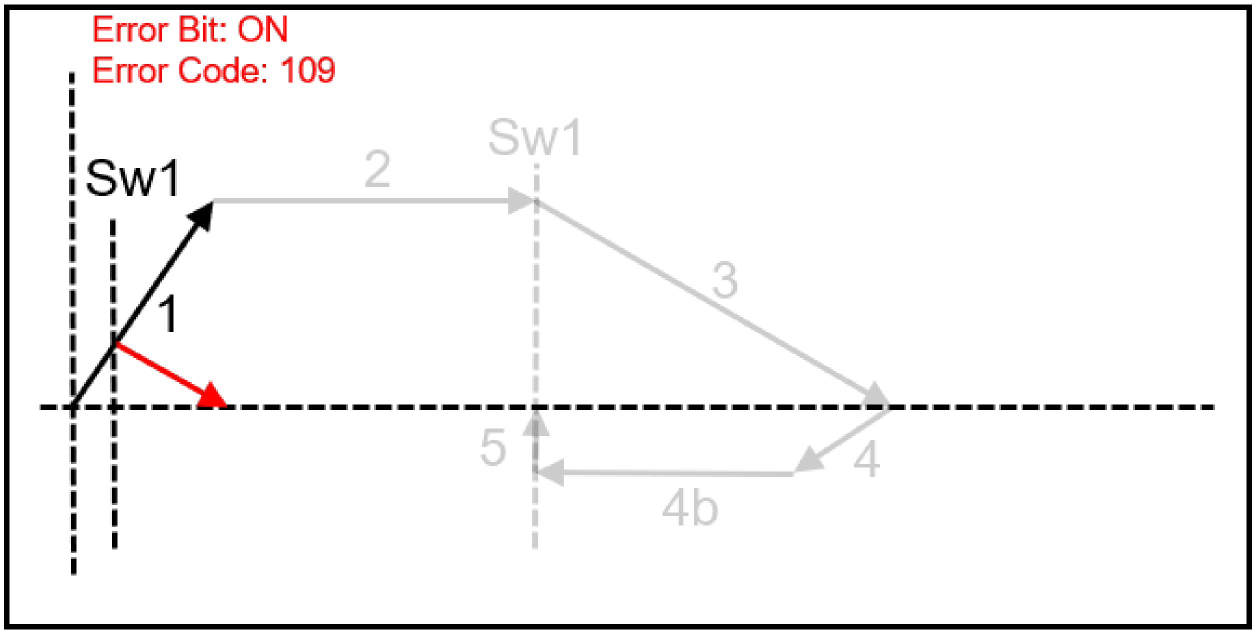
Home Mode4 (Disable Instruction)

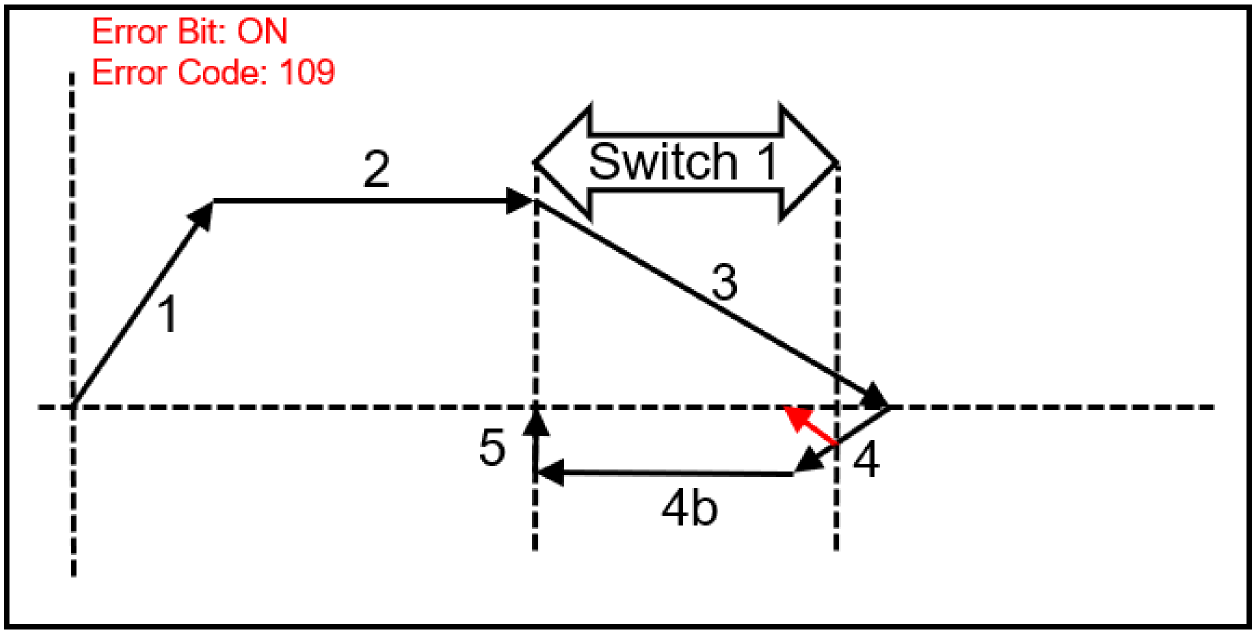
- Disabling during any segment will cause the Error Bit to turn ON, and the Error Code 101.
- The Axis will use the Deceleration Rate to bring the Axis to a Stop.
- Switch inputs are ignored after the instruction is disabled.

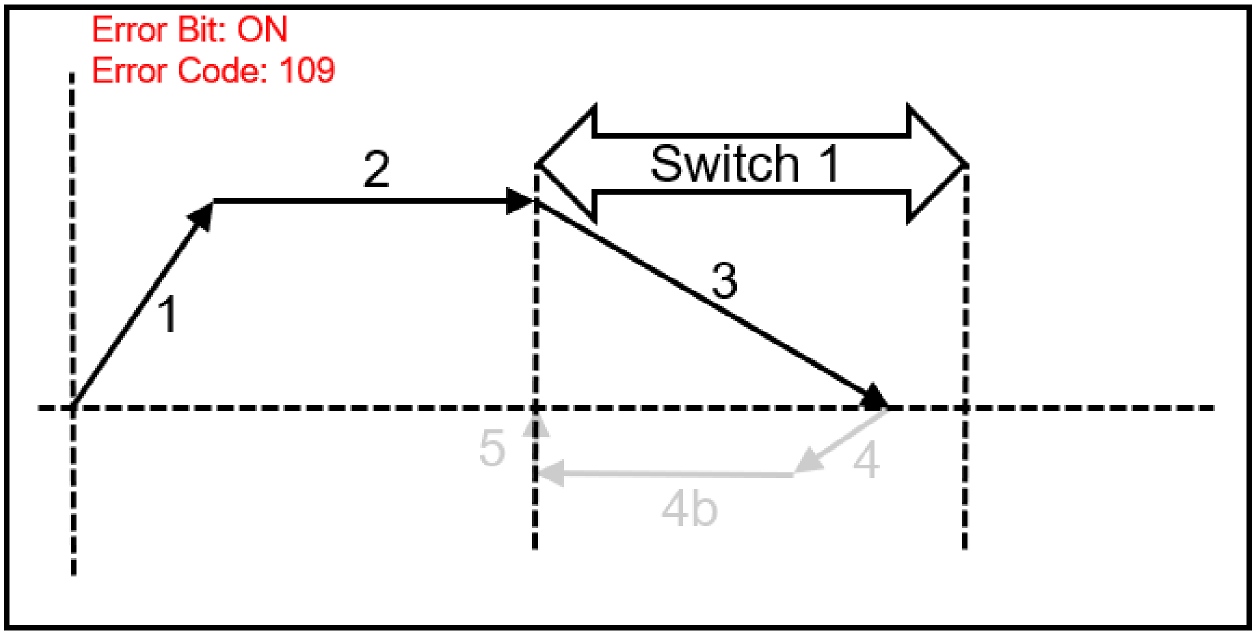


Home Mode4 (Switch Events)

- When Switch1 is turned ON during acceleration, the Error Bit is turned ON and Error Code 109 is set.
- The deceleration distance of section 3 must be longer than the Switch1 activation distance. If the switch is still active at section 4 the Error Bit is turned ON and Error Code 109 is set.
- Switch1 cannot be shared with Limit Switch.

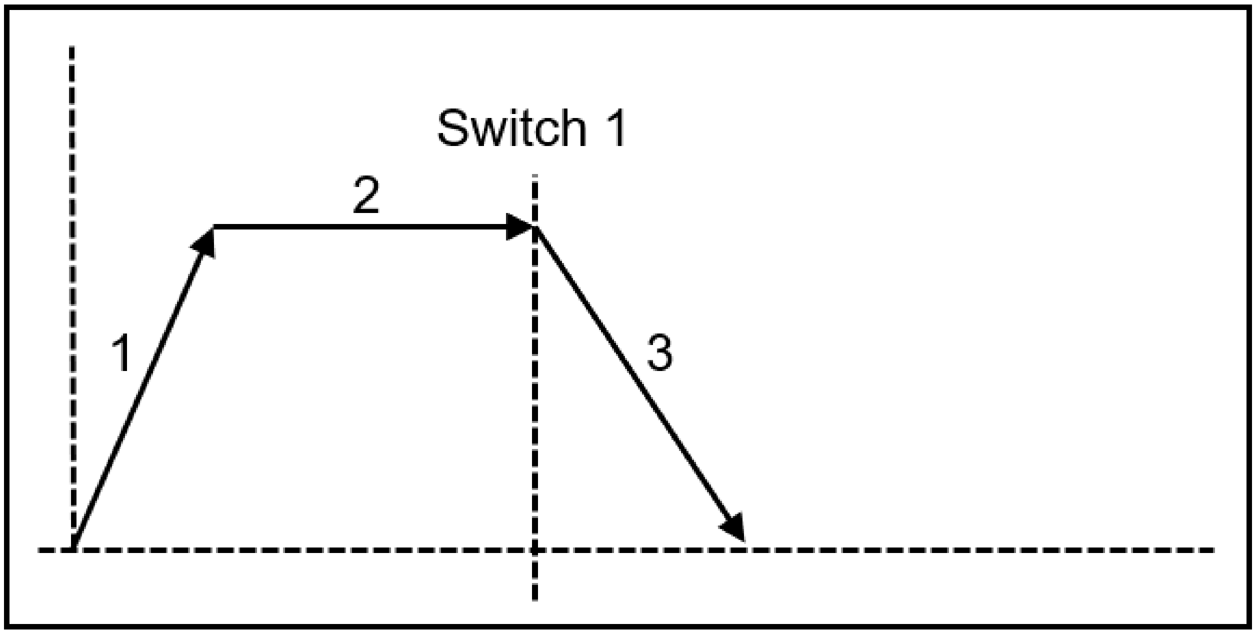






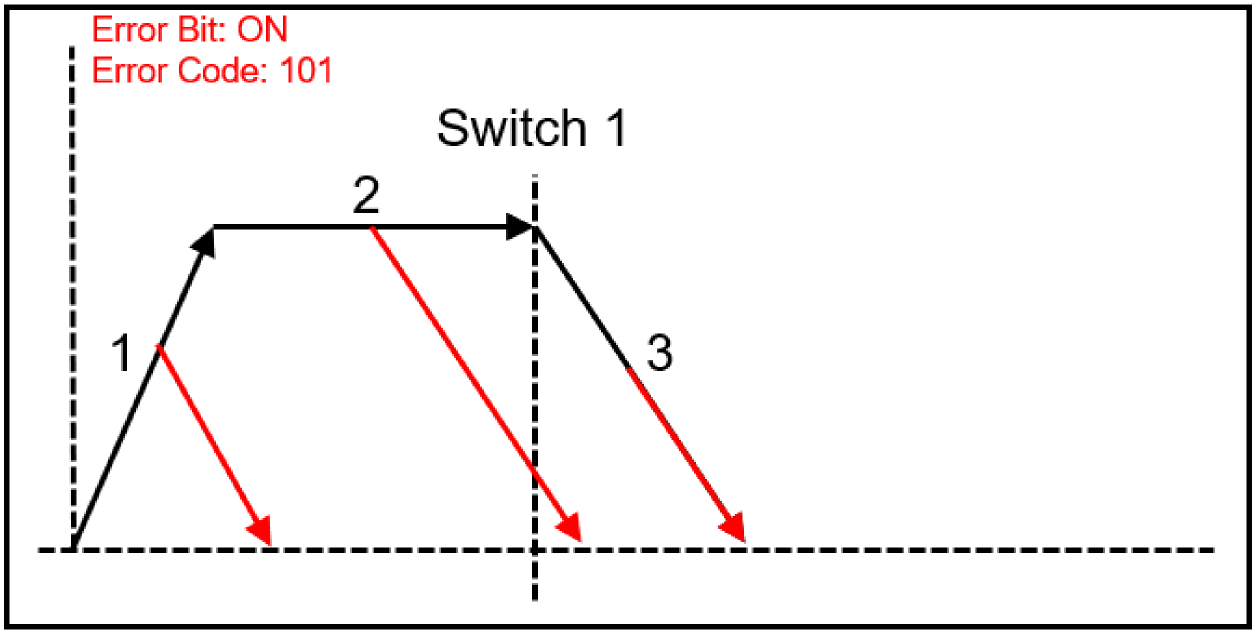
Home Mode5 (Motion Path)

1. Accelerate to the speed of Initial Velocity.
2. Move at Initial Velocity until Switich1 turns ON.
3. It will decelerate to the value set in Stop Distance and stop.



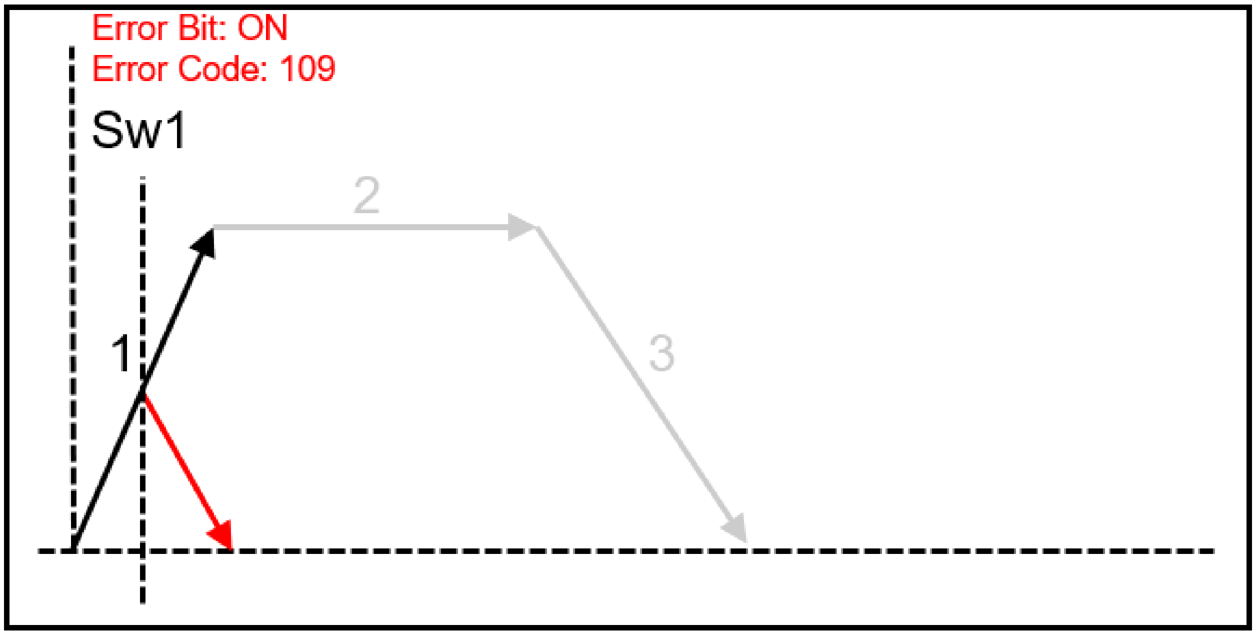
Home Mode5 (Disable Instruction)

- Disabling during any segment will cause the Error Bit to turn ON, and the Error Code 101.
- The Axis will use the Deceleration Rate to bring the Axis to a Stop.
- Switch inputs are ignored after the instruction is disabled.



Home Mode5 (Switch Events)

- When Switch1 is turned ON during acceleration, the Error Bit is turned ON and Error Code 109 is set.
- Switch1 cannot be shared with Limit Switch.
- Sets an error code if the distance traveled from the Position where Switch1 is turned ON to the Stop Distance exceeds the Current Position range (-2147483648 to 2147483647).



<u>Related Topics PTO and PWM Error Codes</u>

[Velocity Move for PTO Axis]
[Position Move for PTO Axis]
[Pulse Train Output]
[Pulse Width Modulation]