RockWell Automation

- PLC Overview
- CompactLogix Controller and Modules
- ControlLogix Controller and Modules
- Network Overview
- Connecting sensors and Actuators to Modules
- Controller Organizer
- Tasks and Tags Types
- Program and Routine
- Connecting PC to PLC via Serial and Ethernet
- Download and Test
- Basic Instructions
- Enhanced Instructions
- Analog Module
- Tasks and Tags In Controller
- Add-on Instruction
- Handling Minor, Major and I/O Faults

RockWell Software & Allen Bradley

Allen – Bradley Hardware:

- Programmable Controller
- HMI (Human Machine Interface)
- I/O ….v…v…

Rockwell Software:

- RSLogix 500
- RSLogix 5000
- RSLink…v..v..

Allen Bradley Hardware

FlexLogix I/O and FlexLogix

controllogix

compactlogix

Allen Bradley Hardware

- ControlLogix System.

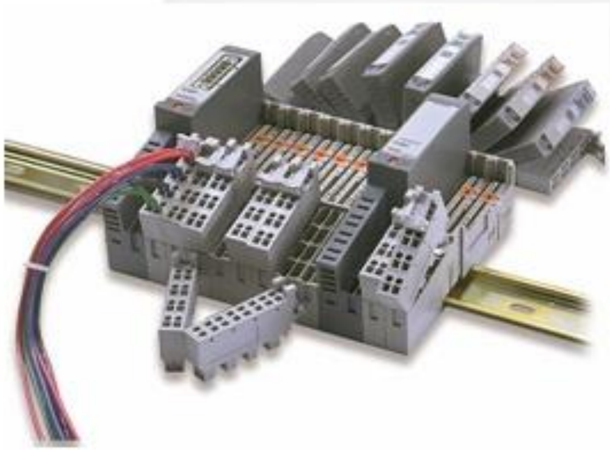
- CompactLogix System.

- FlexLogix System.

Allen – Bradley Hardware

PV terminal

PV plus terminal

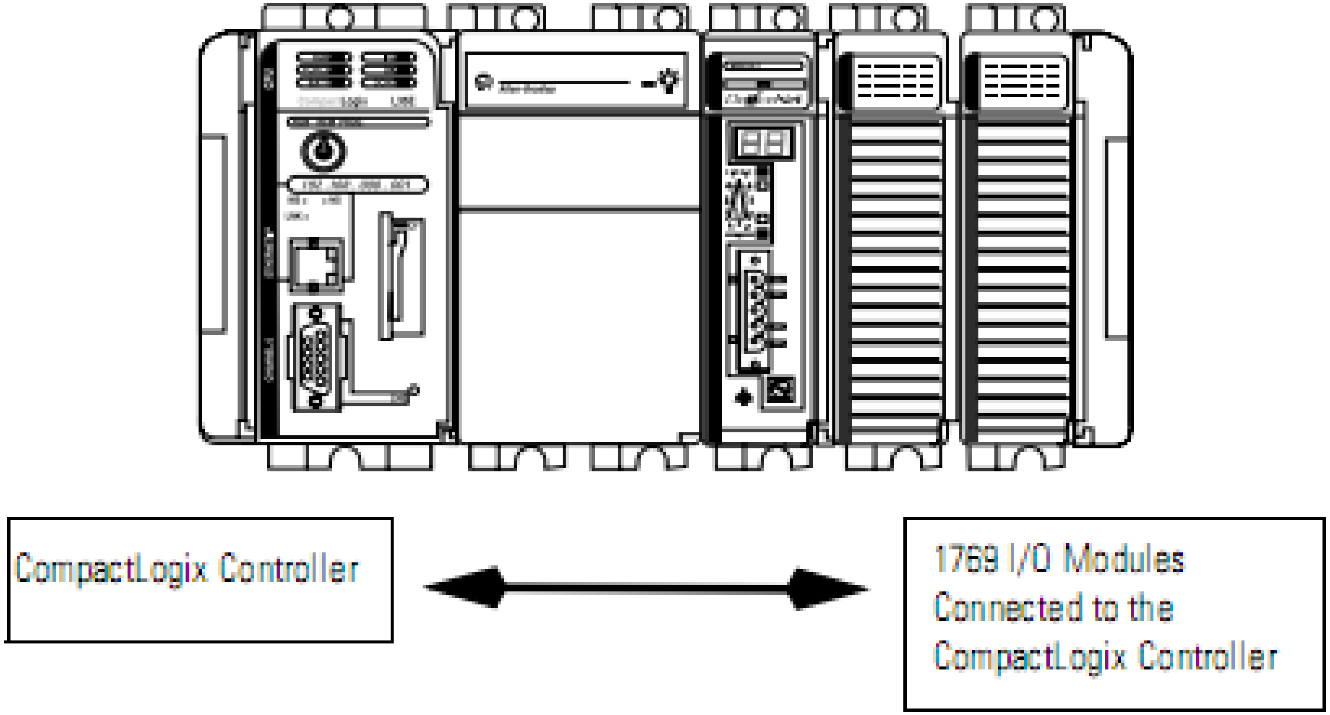


I/O

I/O - HMI

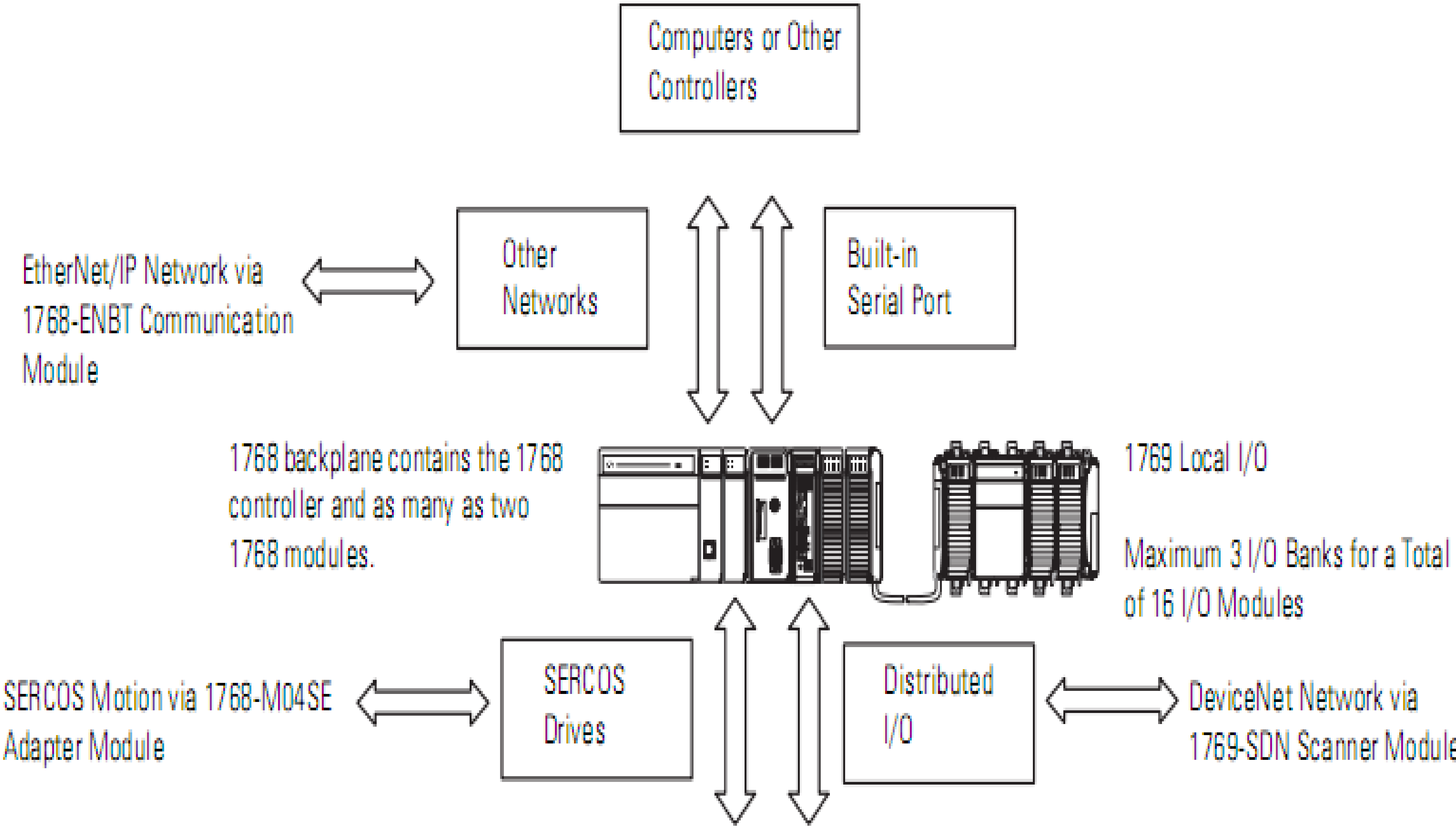
COMPACTLOGIX OVERVIEW

CompactLogix is designed to provide a Logic Solution for machine-level control applications with I/O modules, motion and network requirements.



COMPACTLOGIX OVERVIEW

Complex CompactLogix System



Computers or Other Controllers

EtherNet/IP Network via 1768-ENBT Communication Module

Other Networks

Built-in Serial Port

1768 backplane contains the 1768 controller and as many as two 1768 modules.

1769 Local I/O

Maximum 3 I/O Banks for a Total of 16 I/O Modules

SERCOS Motion via 1768-M04SE Adapter Module

SERCOS Drives

Distributed I/O

DeviceNet Network via 1769-SDN Scanner Module

COMPACTLOGIX OVERVIEW

Some CompactLogix Controllers

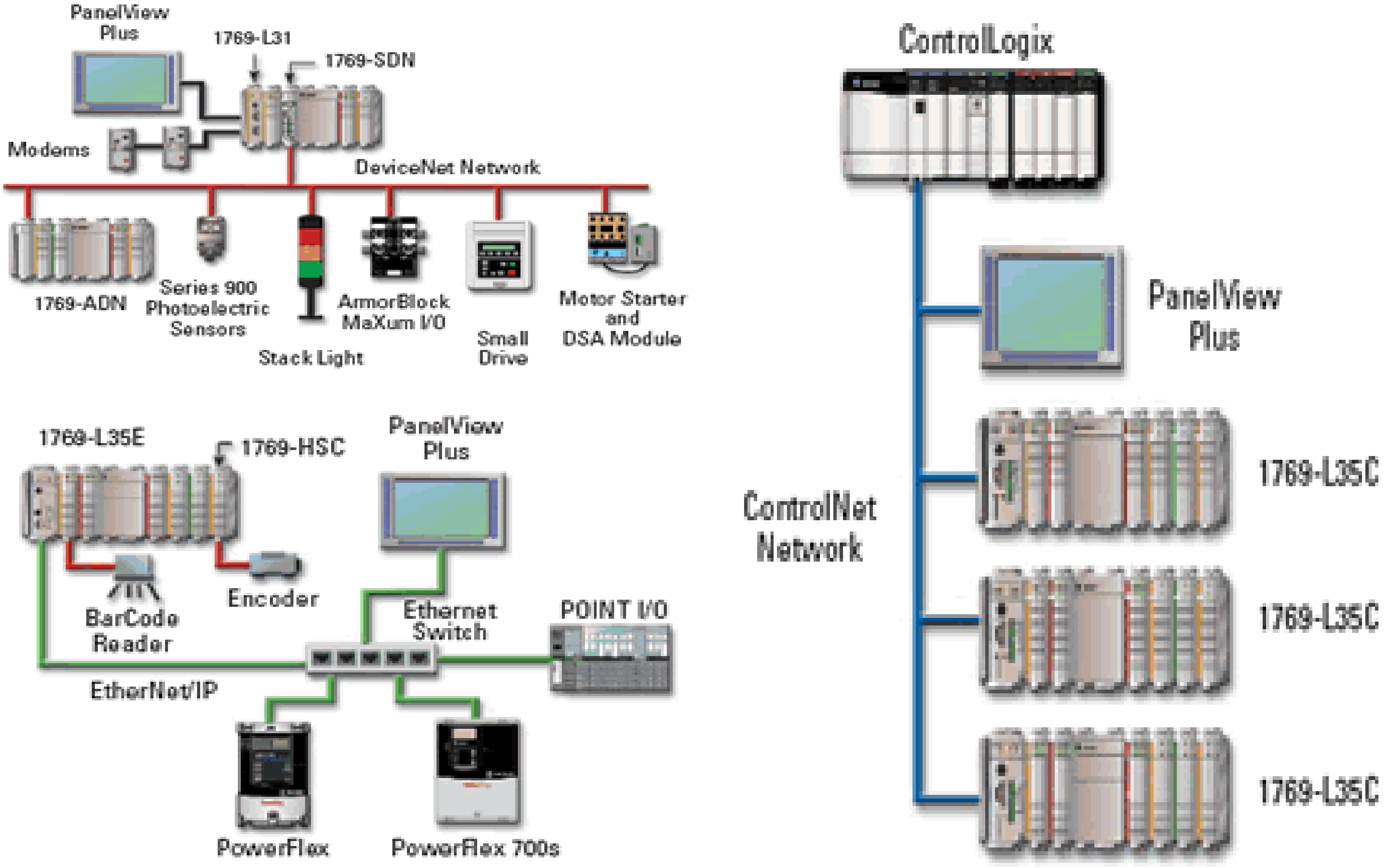
Compactlogix L4x: x = 3/5

Compactlogix L3xy: x = 1/2/5, y = E/C

CompactLogix-L2x

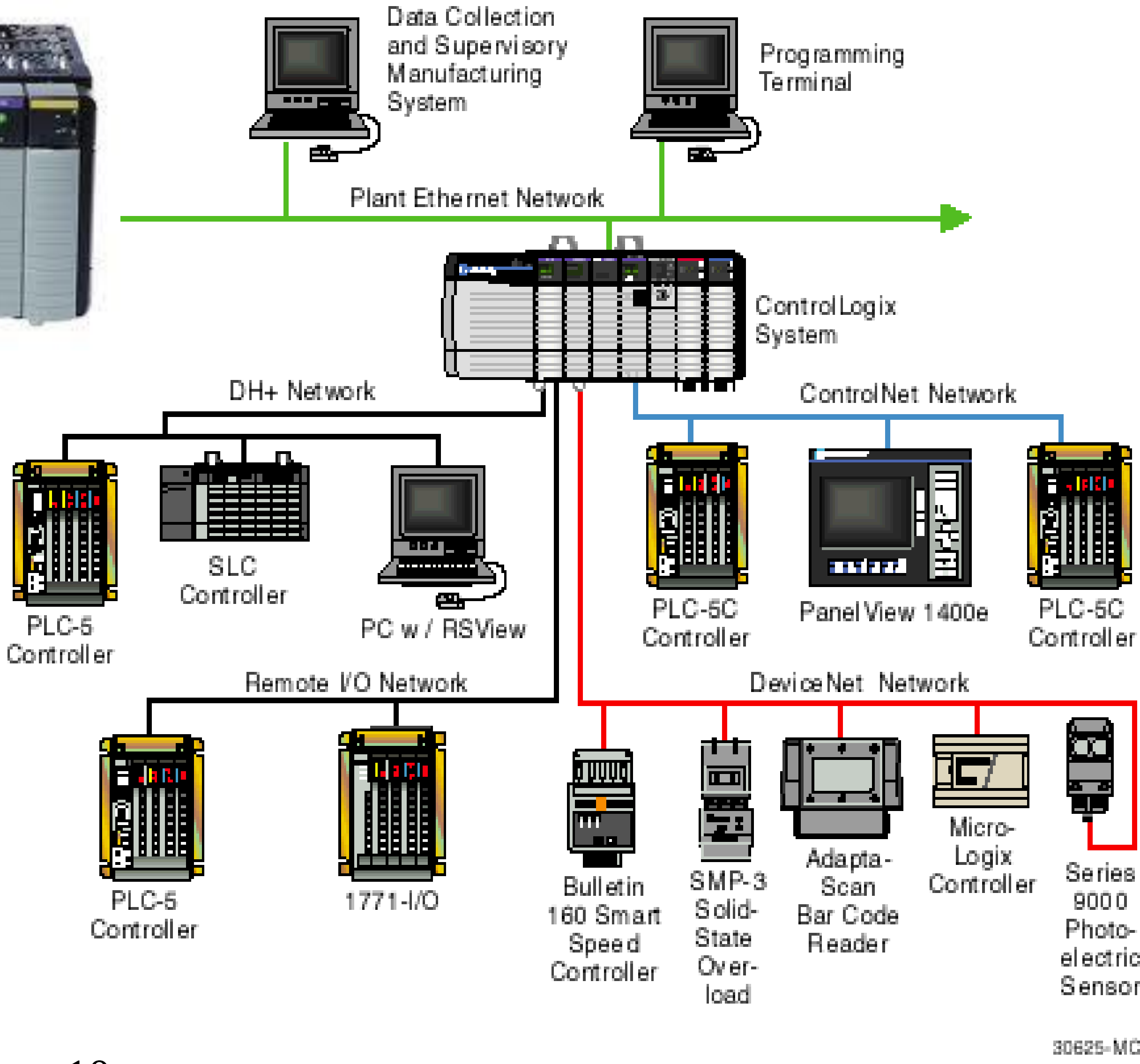
COMPACTLOGIX OVERVIEW

CompactLogix Network Systems

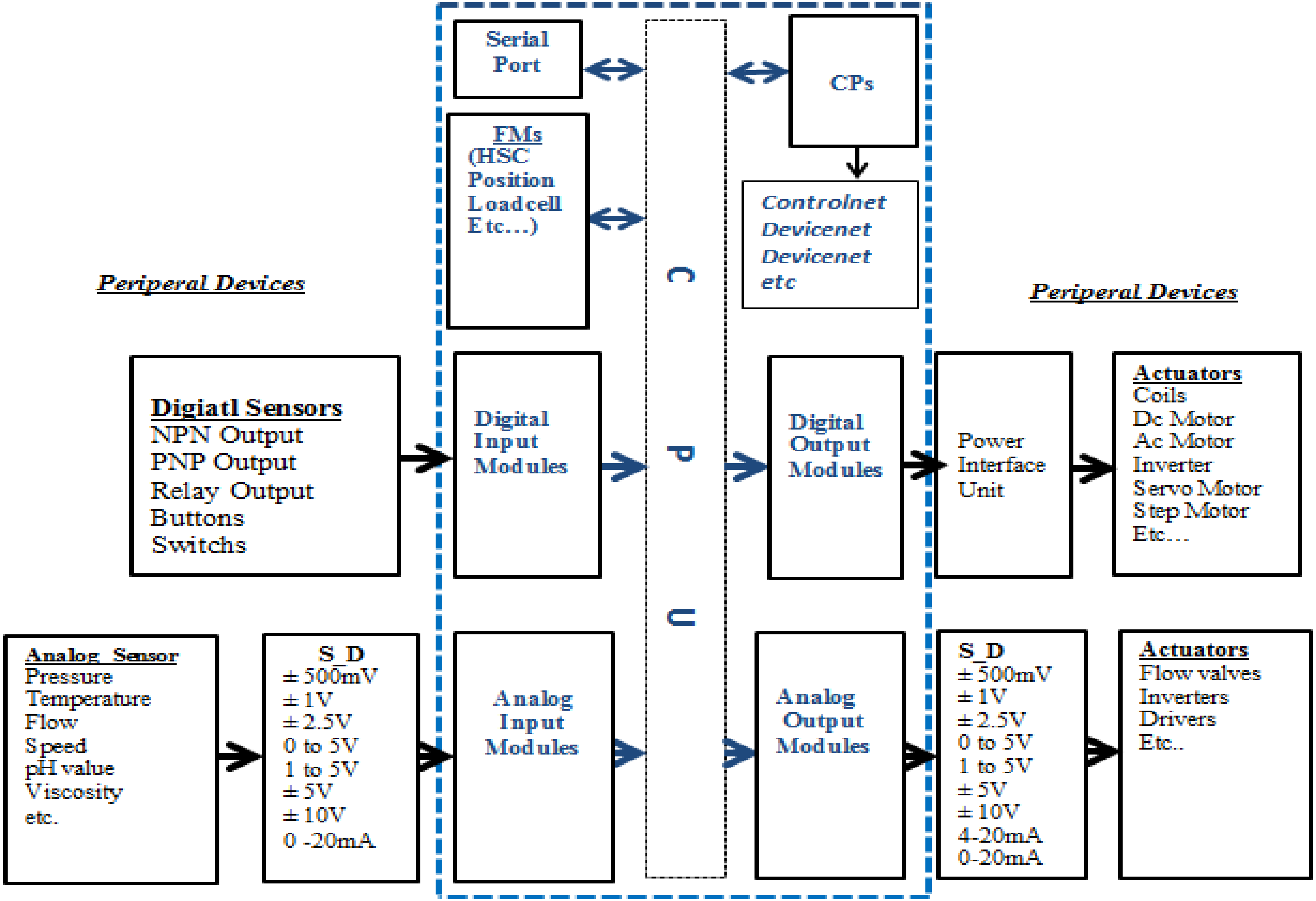


CONTROLLOGIX OVERVIEW

ControlLogix Network Systems



PLC OVERVIEW



Serial Port

FMs
(HSC
Position
Loadcell
Etc…)

C P U

CPs

Controlnet
Devicenet
Devicenet
etc

Periperal Devices

Digiatl Sensors
NPN Output
PNP Output
Relay Output
Buttons
Switchs

Digital Input Modules

Digital Output Modules

Periperal Devices

Power Interface Unit

Actuators
Coils
Dc Motor
Ac Motor
Inverter
Servo Motor
Step Motor
Etc…

Analog Sensor
Pressure
Temperature
Flow
Speed
pH value
Viscosity
etc.

S_D
± 500mV
± 1V
± 2.5V
0 to 5V
1 to 5V
± 5V
± 10V
0 -20mA

Analog Input Modules

Analog Output Modules

S_D
± 500mV
± 1V
± 2.5V
0 to 5V
1 to 5V
± 5V
± 10V
4-20mA
0-20mA

Actuators
Flow valves
Inverters
Drivers
Etc..

COMPACTLLOGIX MODULES

1769-IQ32 Sinking/Sourcing 24V DC Input

On state:

- Min voltage: 10V, I = 2mA
- Max voltage: 30V, I=10mA

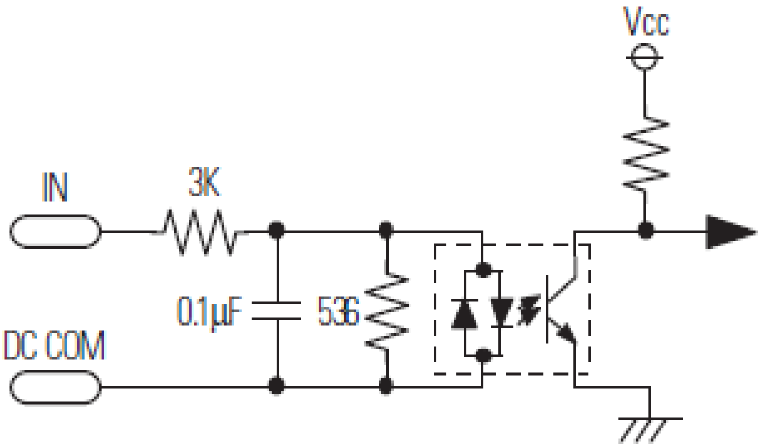
OFF state

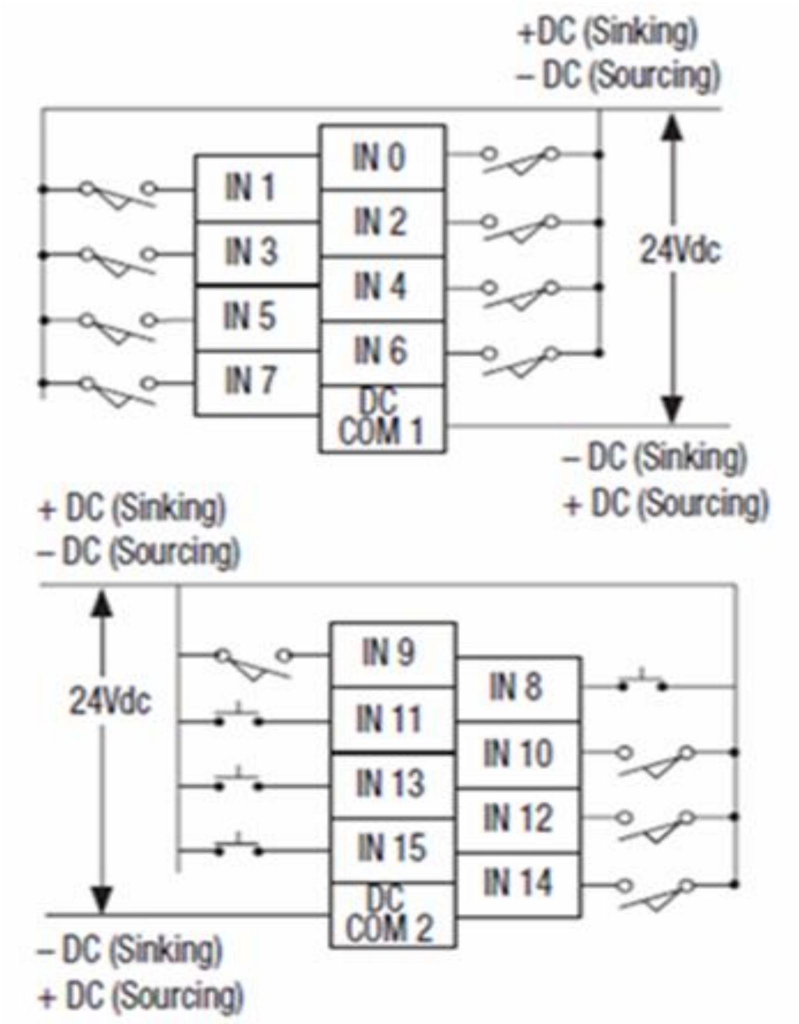
- Max voltage 5V.
- Max current: 1,5mA.

Time to change from ON and OFF state is 8ms.

COMPACTLLOGIX MODULES

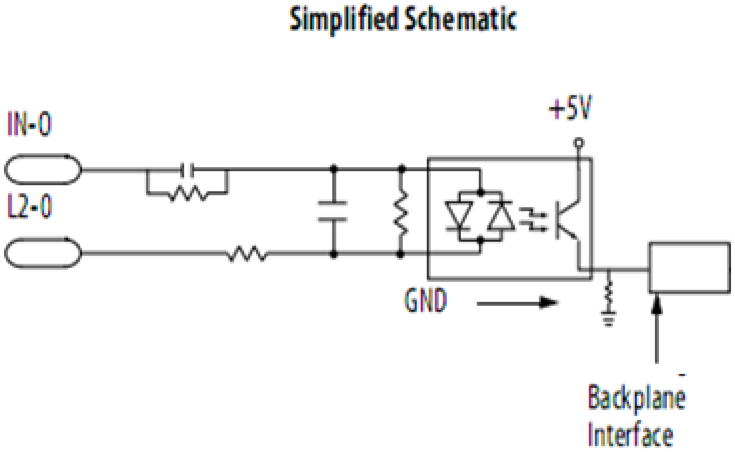
1769-IQ32 Sinking/Sourcing 24V DC Input

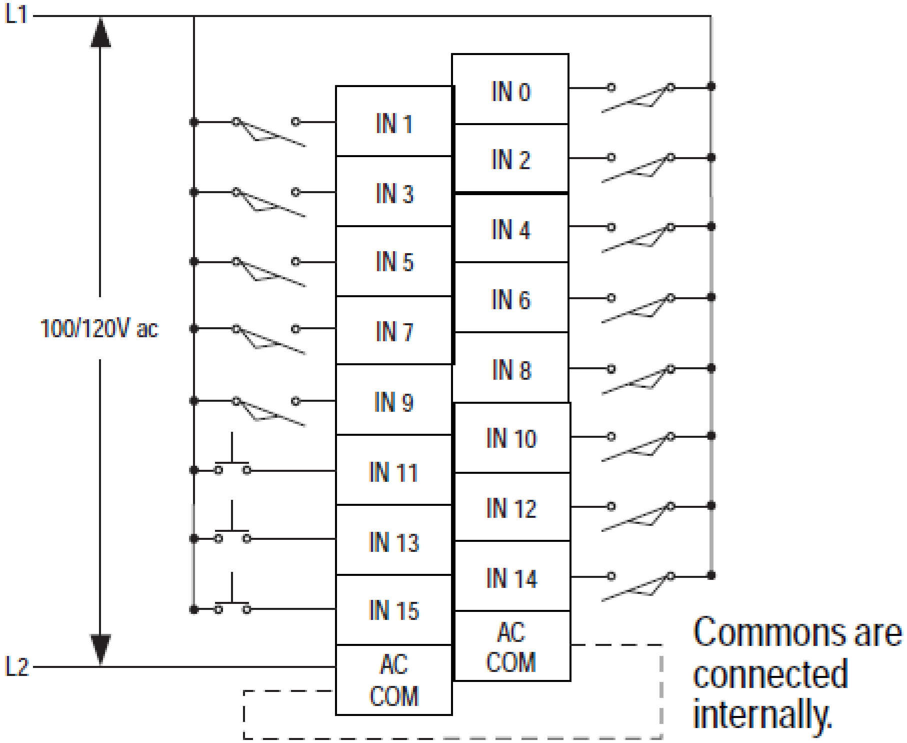




COMPACTLLOGIX MODULES

1769-IA16 Module Input Wiring

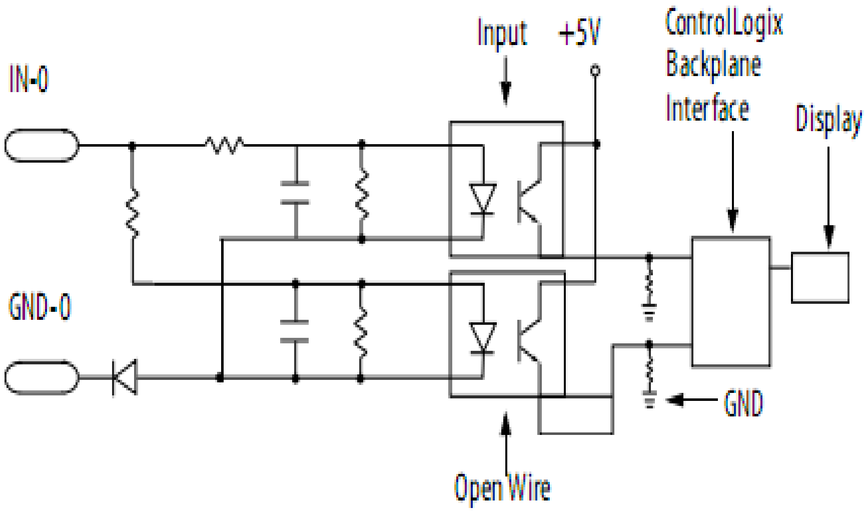


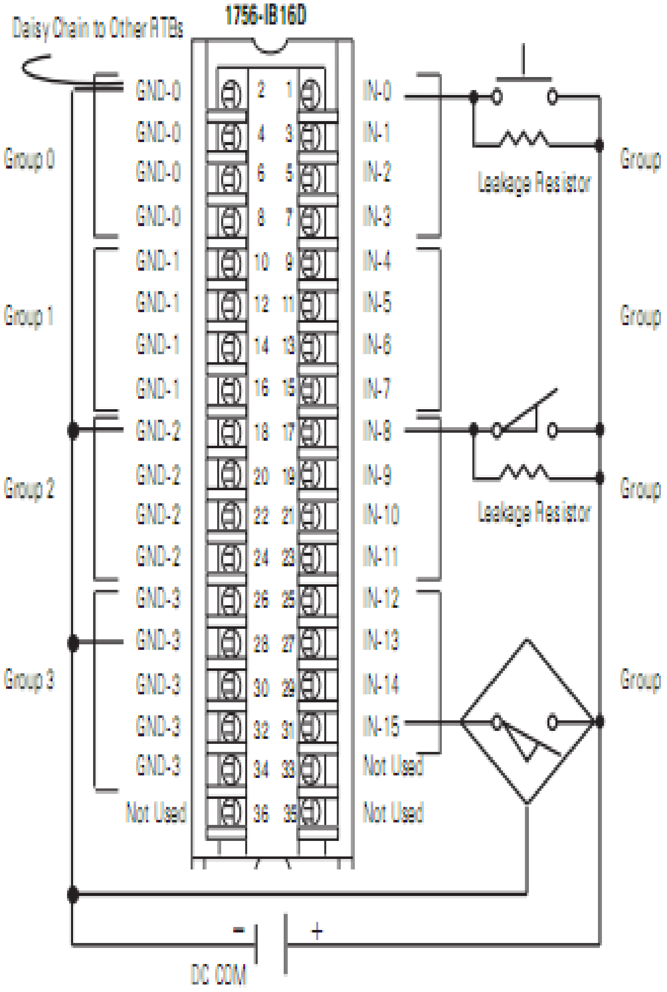


CONTROLLOGIX MODULE

ControlLogix DC (10..30V) diagnostic Input Module

Simplified Schematic

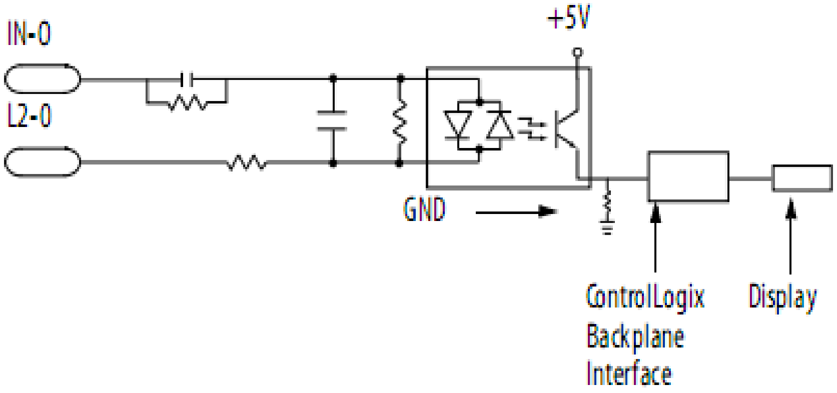


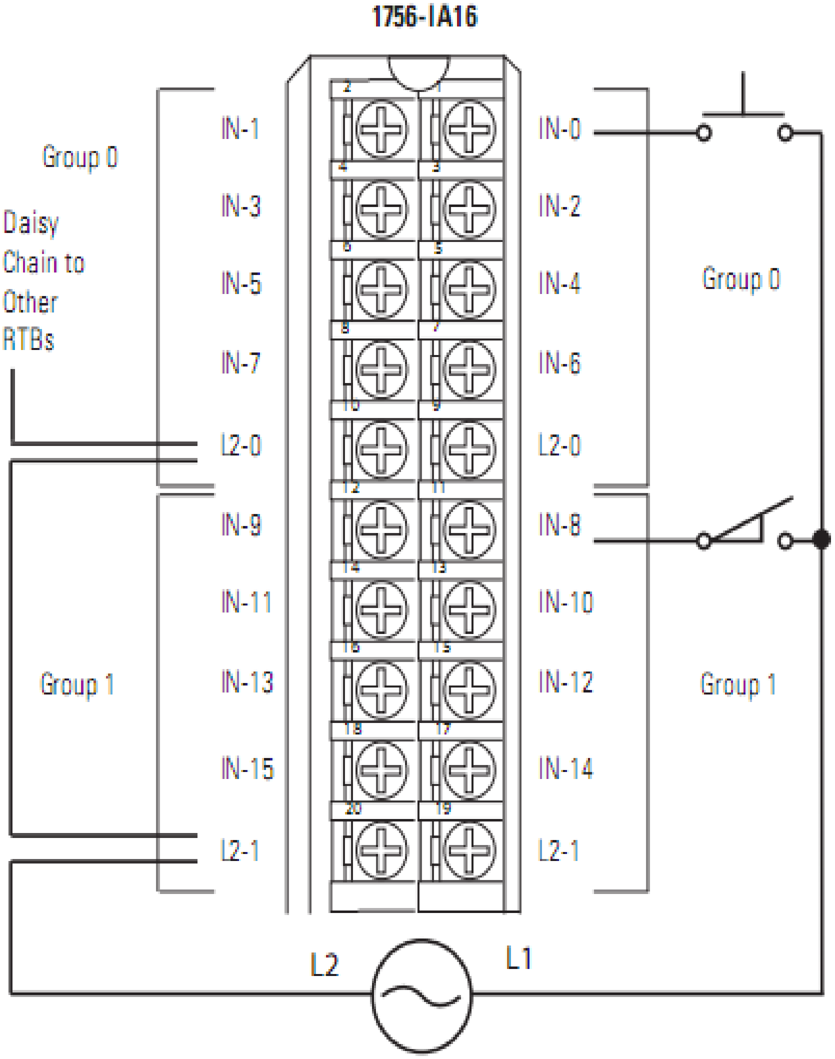


CONTROLLOGIX MODULE

ControlLogix AC (74..132V) Input Module

Simplified Schematic



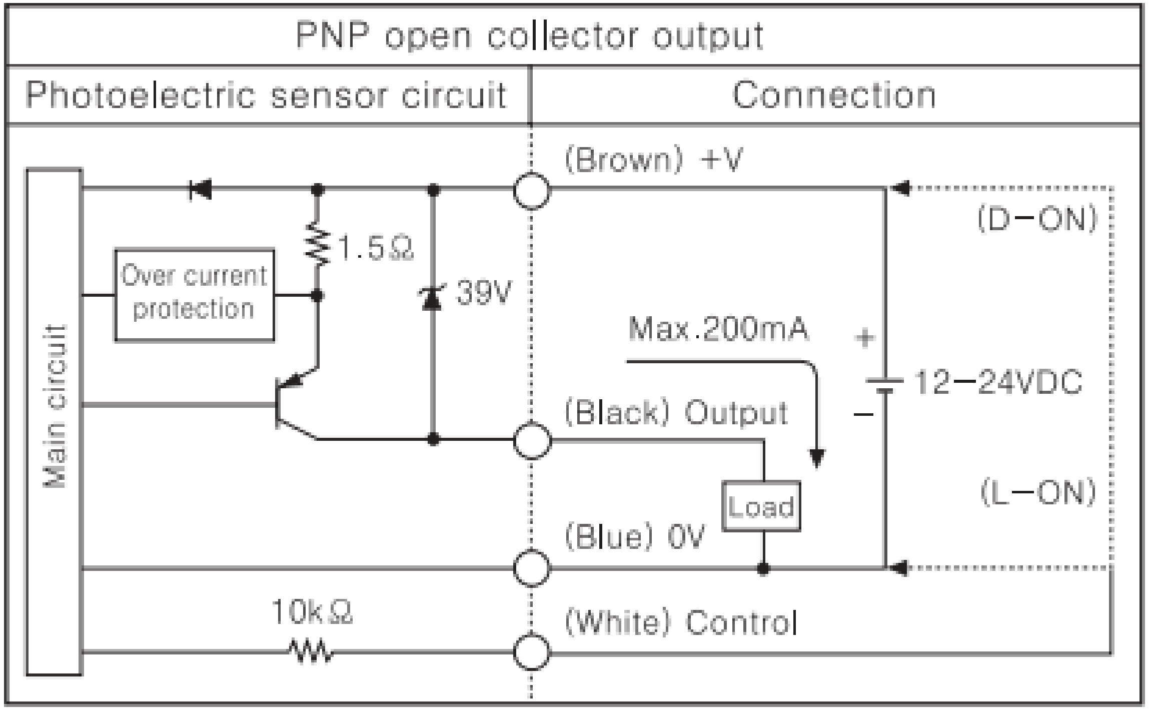


COMPACTLOGIX MODULES

Input Digital Module Connection

Ex1: Connecting PNP sensors to Input DC,AC module

- BR(P)100-DDT-P / BR(P)200-DDTN-P / BR(P)400-DDT-P
- BR20M-TDTD2-P / BR20M-TDTL2-P (Receiver)



COMPACTLOGIX MODULES

Input Digital Module Connection

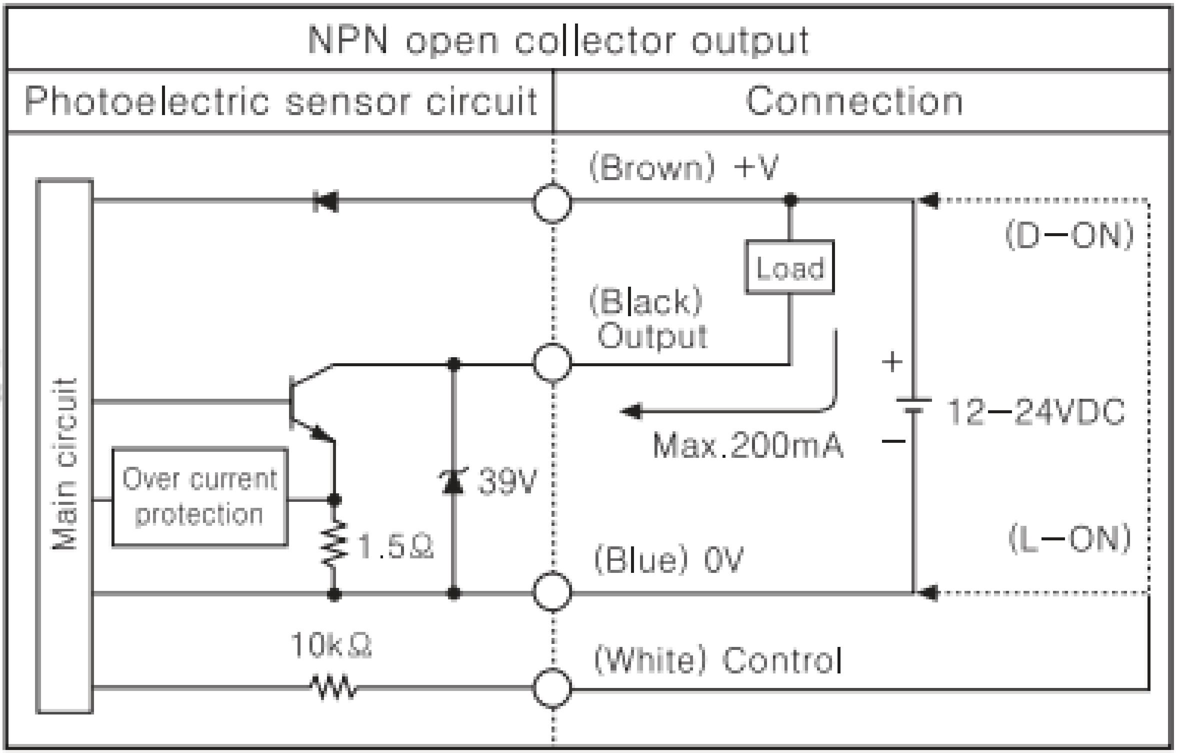
Ex1: Connecting PNP sensor to Input DC,AC module

COMPACTLOGIX MODULES

Input Digital Module Connection

Ex2: Connecting NPN sensors to Input DC,AC module

●BR(P)100-DDT / BR(P)200-DDTN / BR(P)400-DDT
●BR20M-TDTD2 / BR20M-TDTL2 (Receiver)



COMPACTLOGIX MODULES

Input Digital Module Connection

Ex2: Connecting NPN sensors to Input DC,AC module

COMPACTLLOGIX MODULES

1769-OB32 Current Sourcing 24V DC Output

1769-OB32

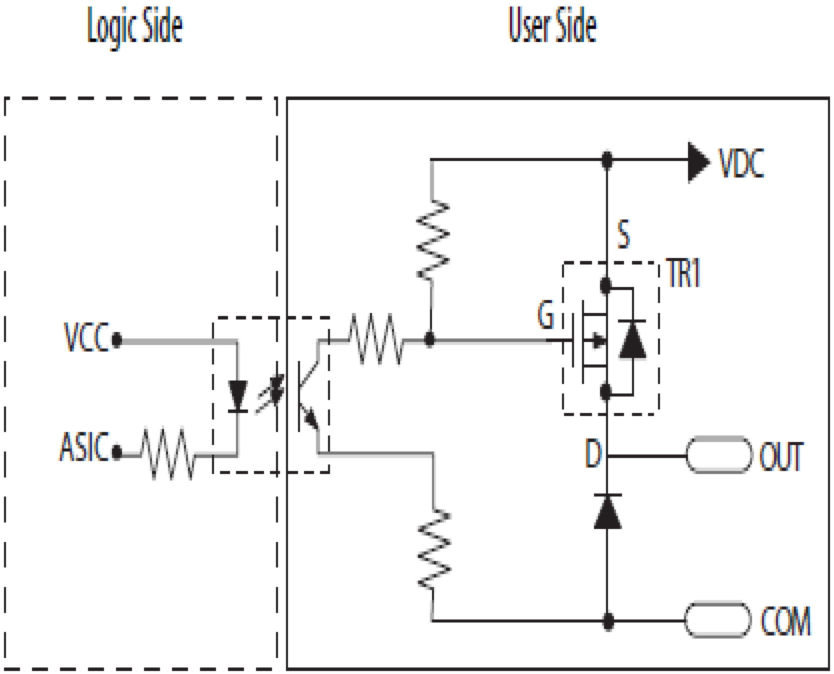
- Min Voltage: **20,4V DC, I = 1mA**
- Max Voltage: **26,4V DC, I = 1A**
- 32 digital Outputs

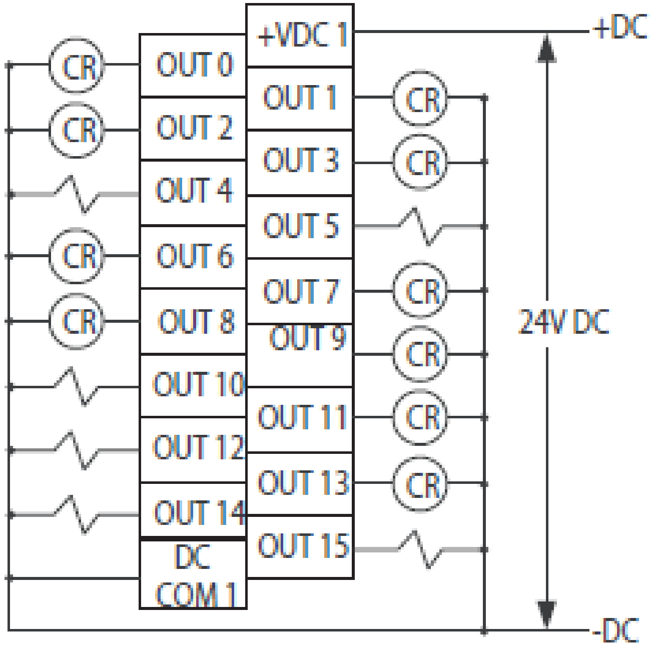
1769-OB32T(Terminated Ouput Module)

- Min Voltage: **10,2V DC, I = 1mA**
- Max Voltage: **26,4V DC, I = 0,5A**
- 32 digital Outputs

COMPACTLLOGIX MODULES

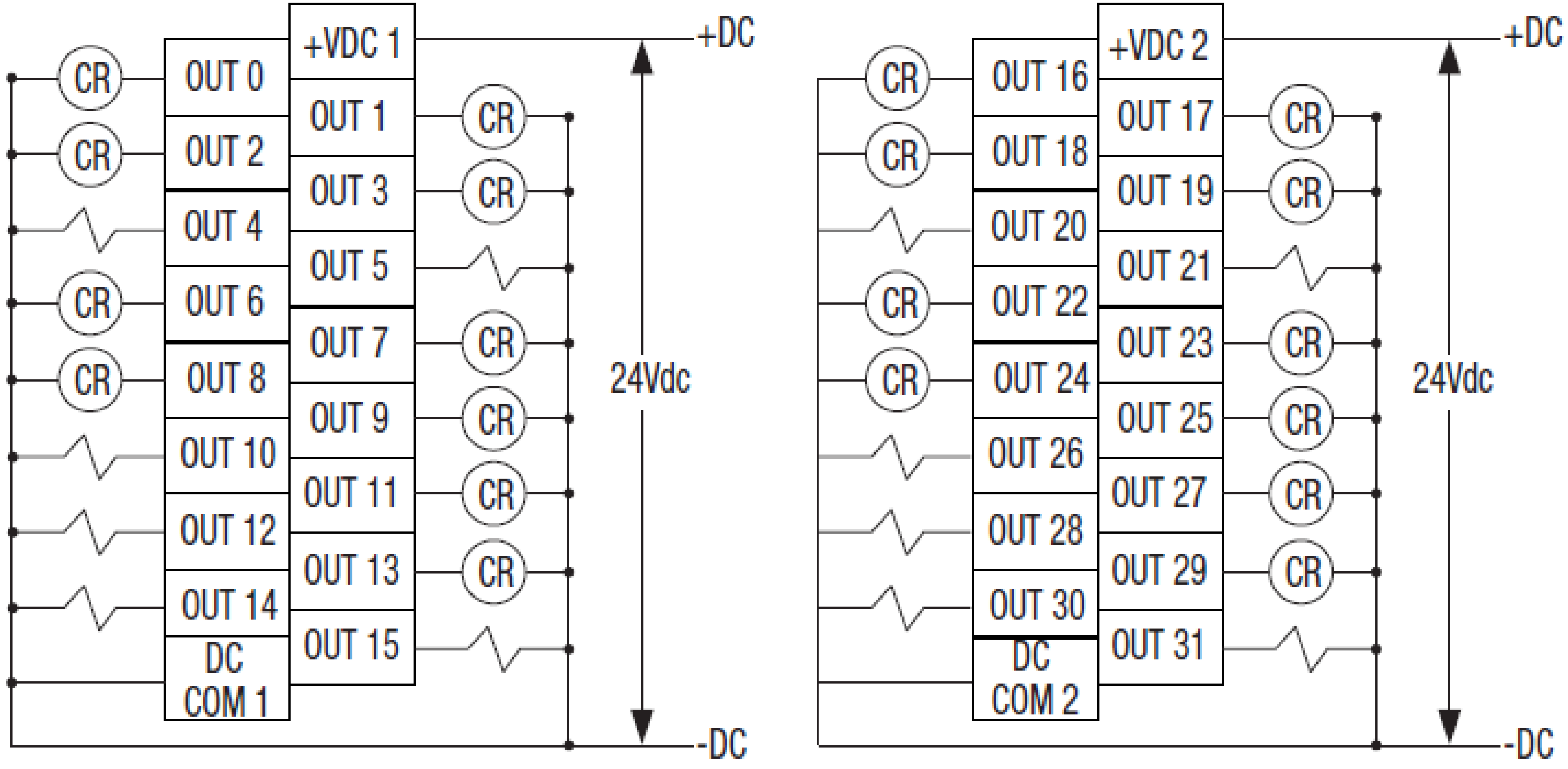
1769-OB32 Current Sourcing 24V DC Output





COMPACTLLOGIX MODULES

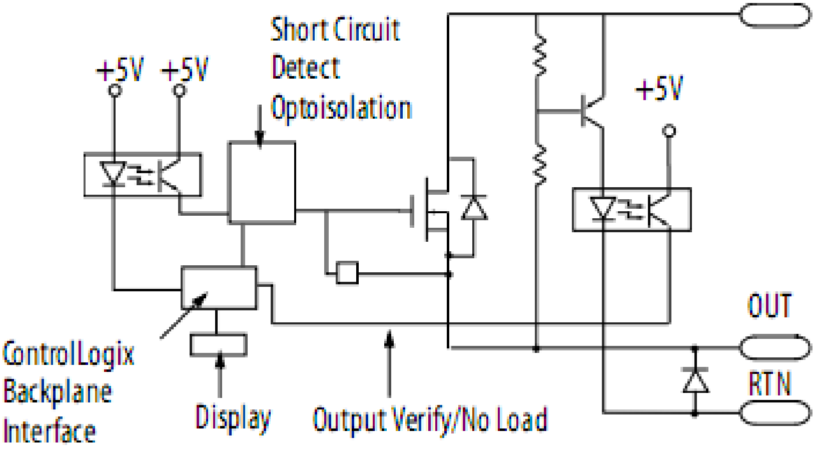
1769-OB32 Current Sourcing 24V DC Output



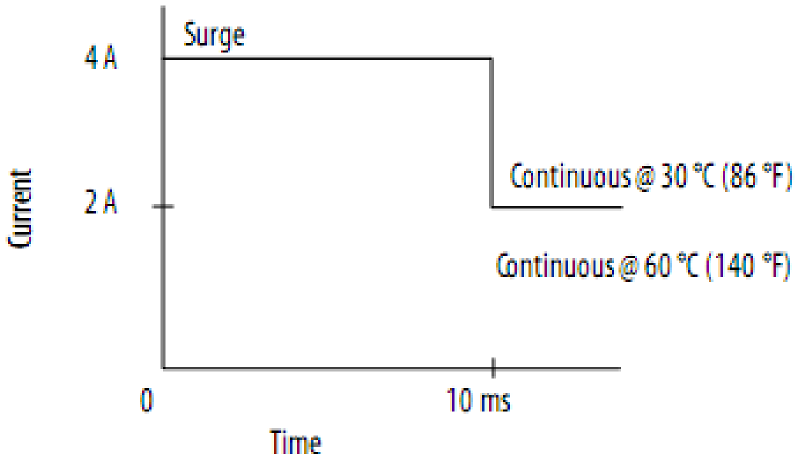
CONTROLLOGIX MODULES

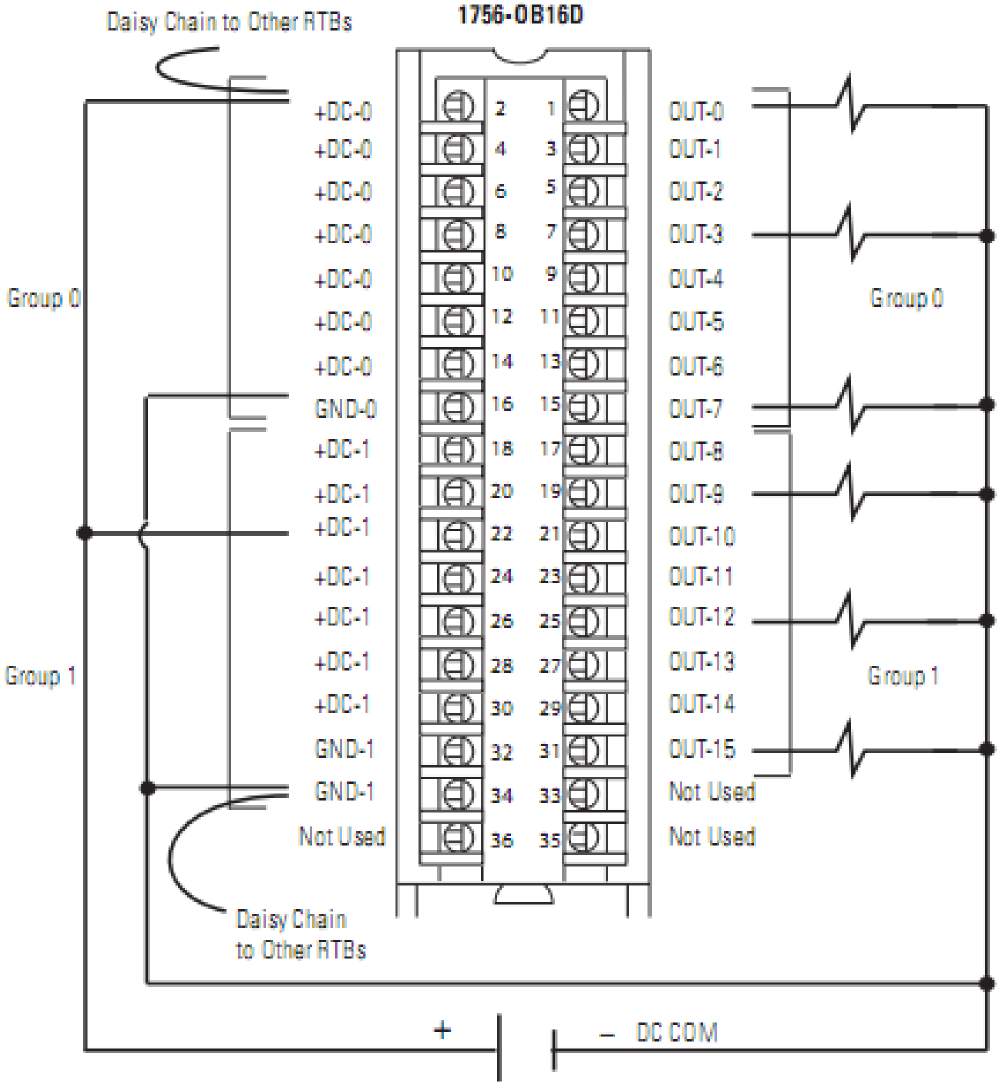
ControlLogix DC diagnostic Output Module

Simplified schematic



Surge Current Chart

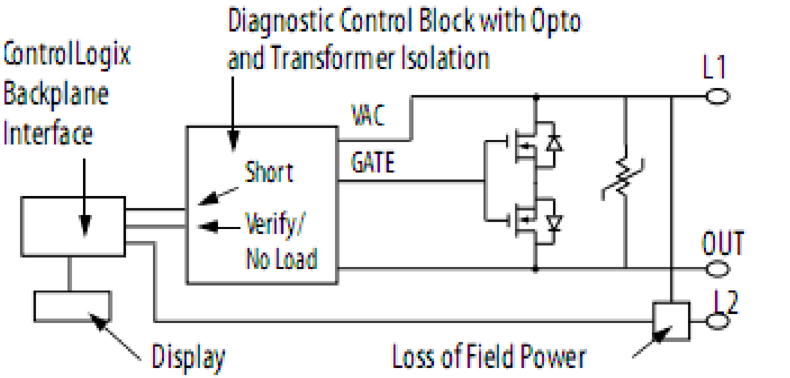




CONTROLLOGIX MODULES

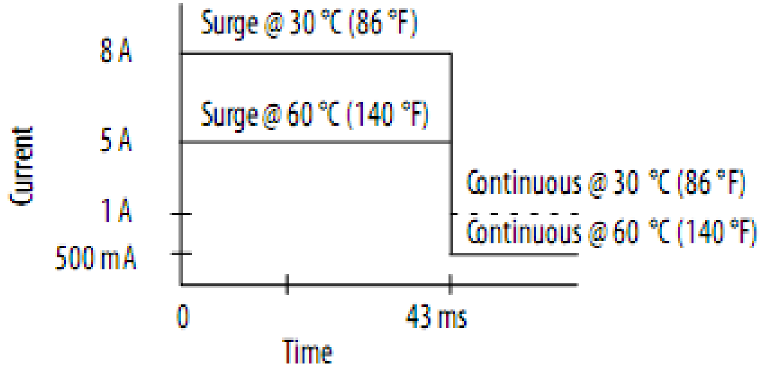
ControlLogix AC diagnostic Output Module

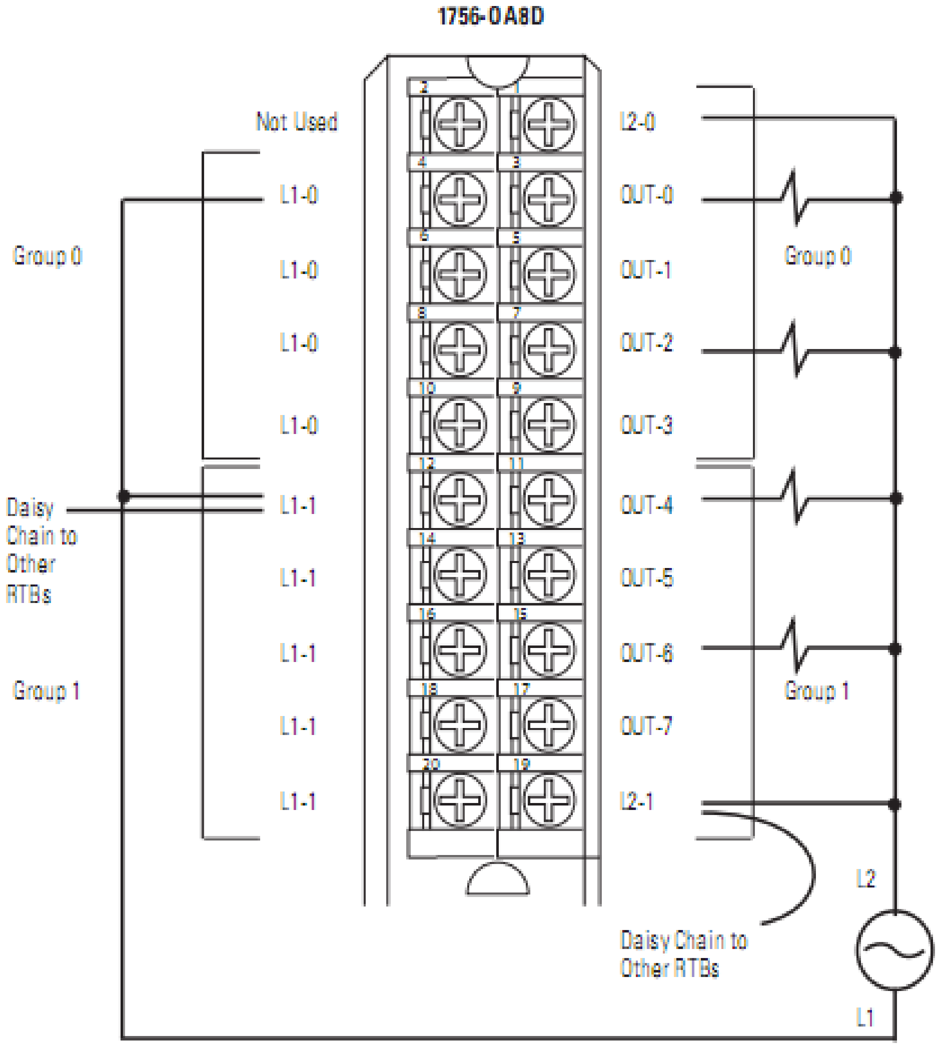
Simplified Schematic



ControlLogix Backplane Interface
Diagnostic Control Block with Opto and Transformer Isolation
L1
VAC
GATE
Short
Verify/ No Load
OUT
L2
Display
Loss of Field Power

Surge Current Chart

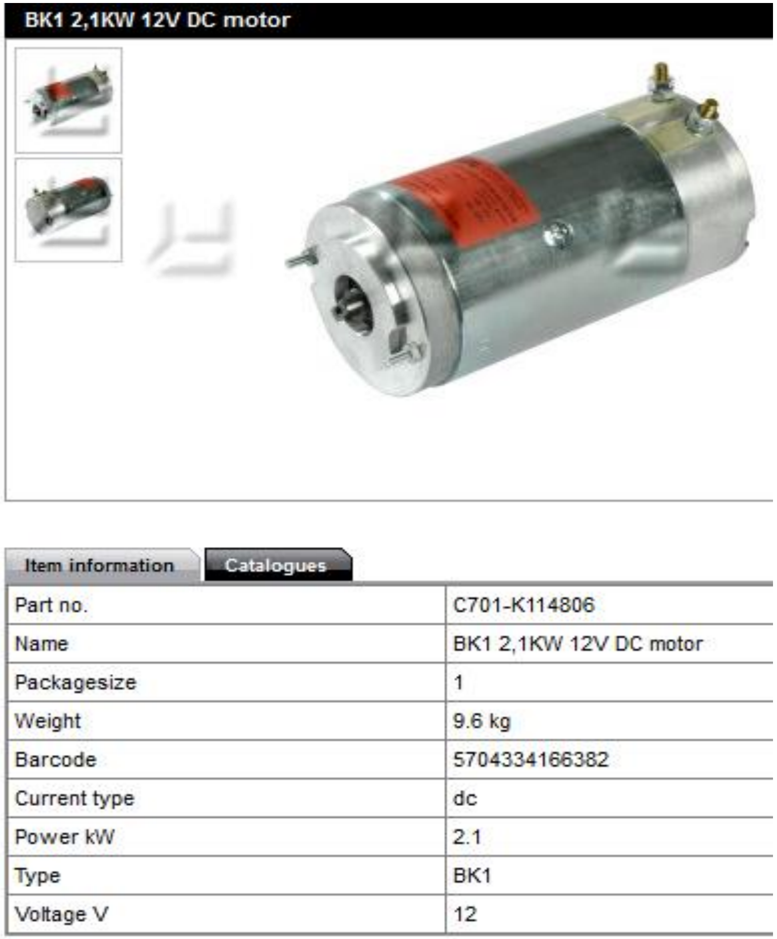




INPUT OUTPUT PLC CONNECTING

PLC Output Connection

Ex3: Connecting DC motor(ON_OF) to PLC output Module



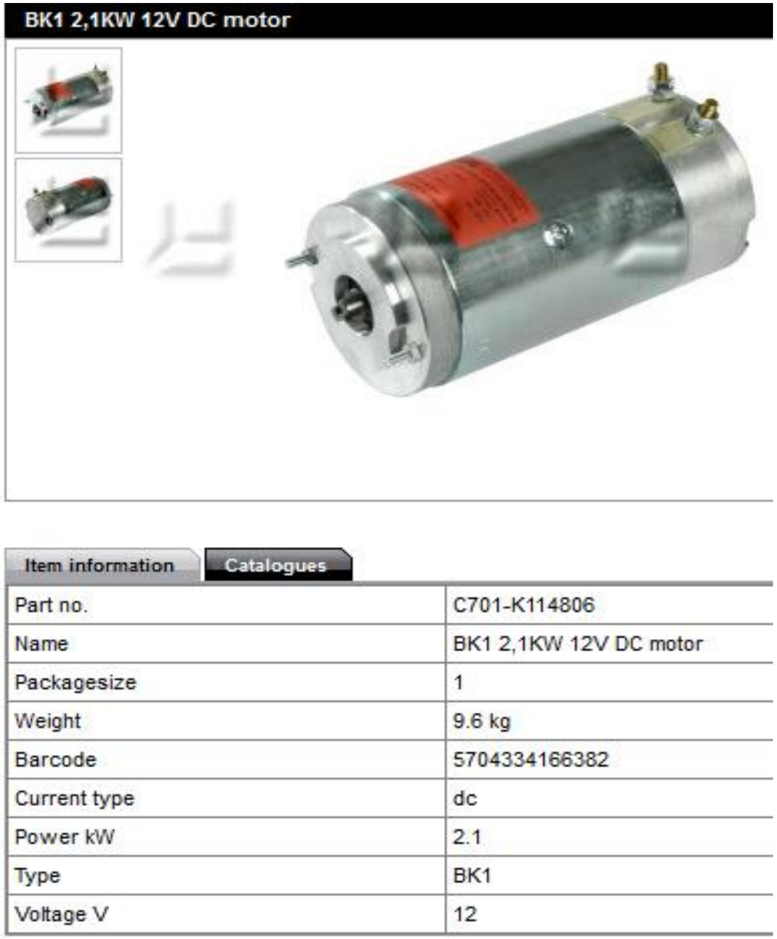
BK1 2,1KW 12V DC motor

Part no.	C701-K114806
Name	BK1 2,1KW 12V DC motor
Packagesize	1
Weight	9.6 kg
Barcode	5704334166382
Current type	dc
Power kW	2.1
Type	BK1
Voltage V	12

INPUT OUTPUT PLC CONNECTING

Connecting Actuators to output digital module

Ex4: Connecting DC motor(PWM mode) to PLC output module



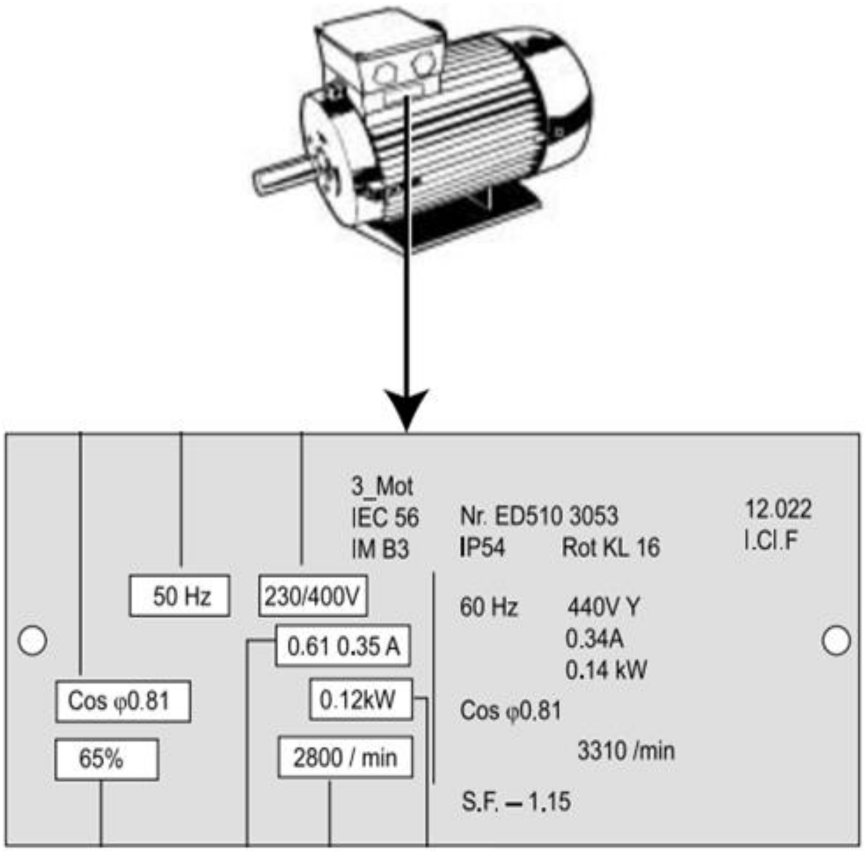
BK1 2,1KW 12V DC motor

Part no.	C701-K114806
Name	BK1 2,1KW 12V DC motor
Packagesize	1
Weight	9.6 kg
Barcode	5704334166382
Current type	dc
Power kW	2.1
Type	BK1
Voltage V	12

INPUT OUTPUT PLC CONNECTING

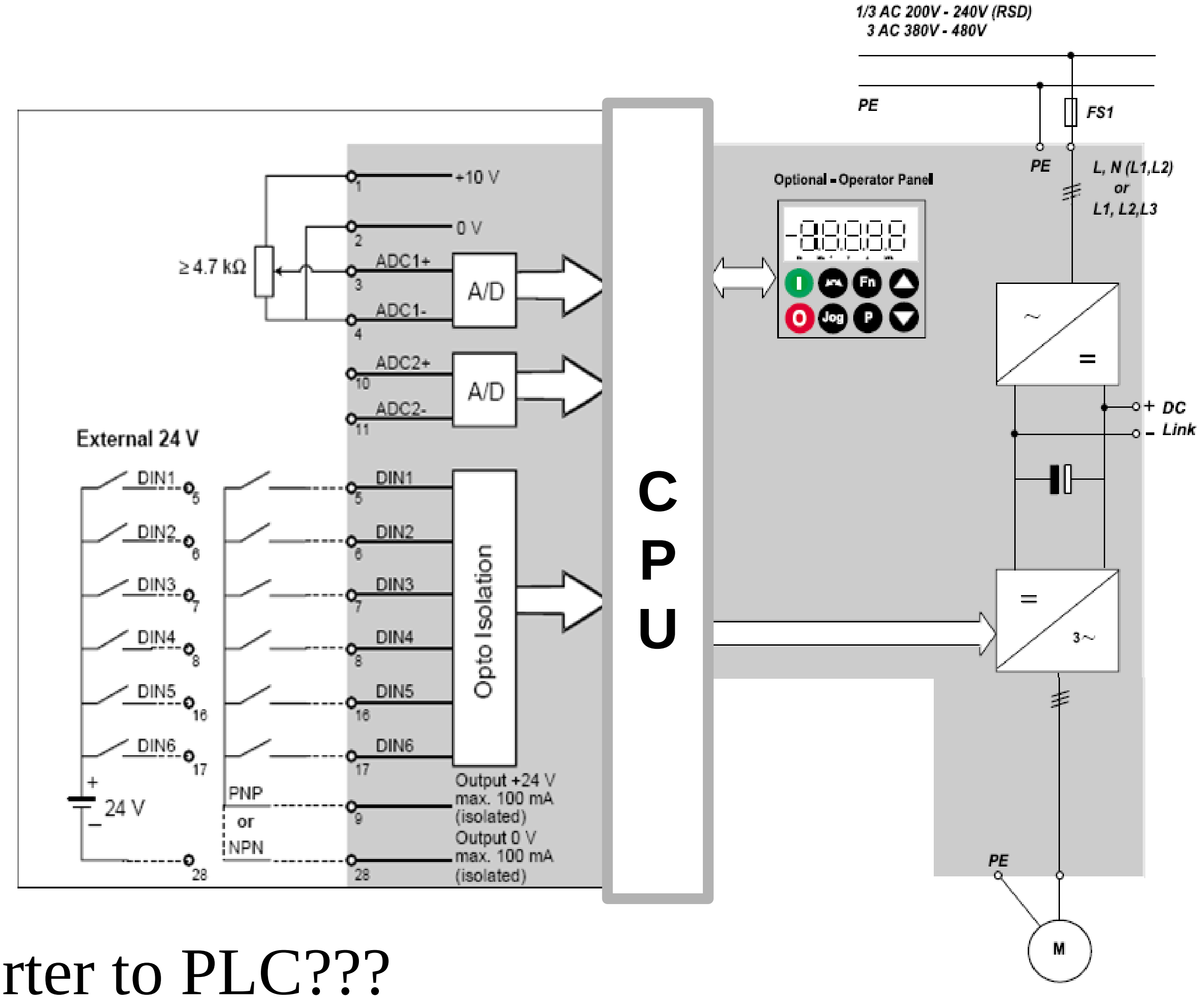
Connecting AC Motor to PLC output module

Ex5: Connecting a three phase motor to Output digital module



INPUT OUTPUT PLC CONNECTING

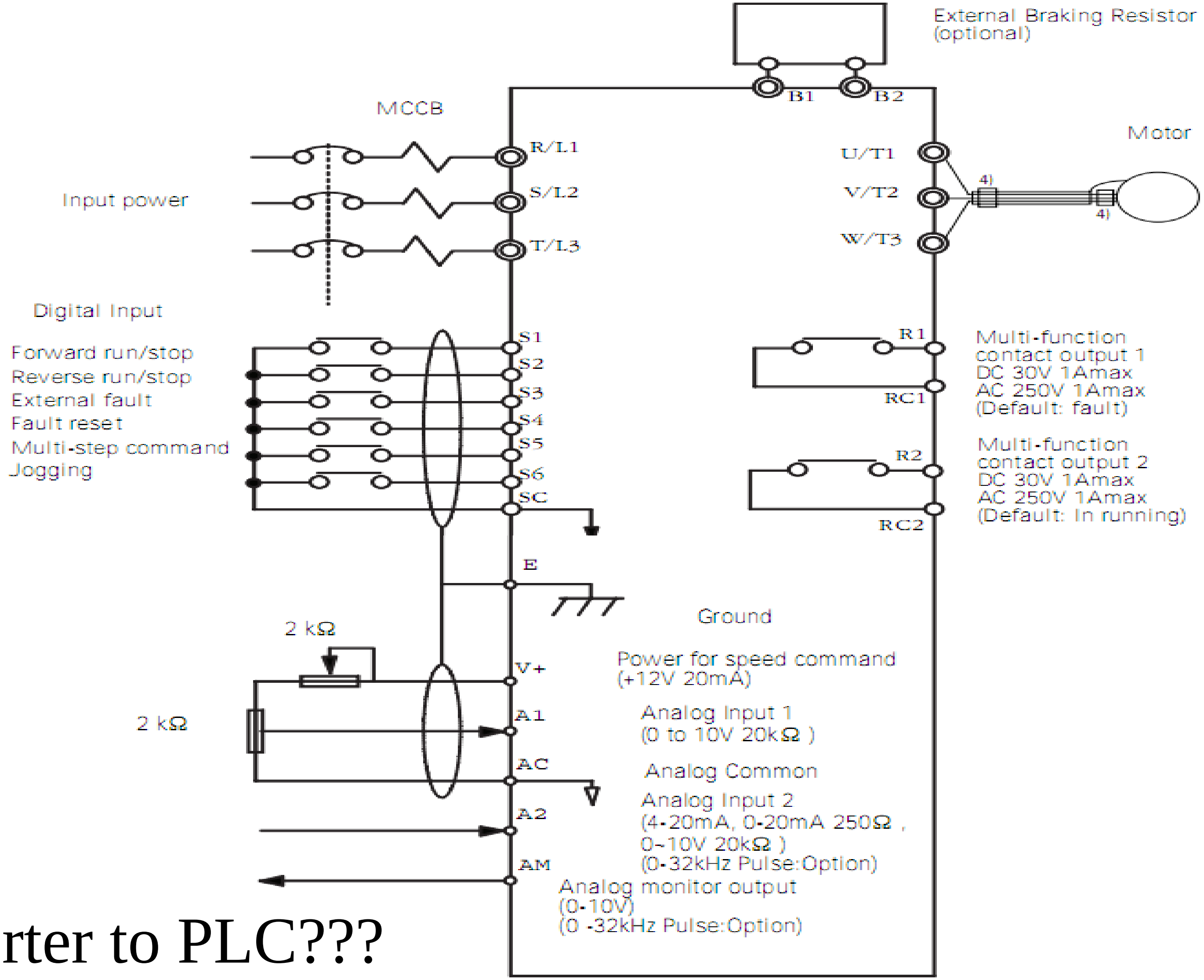
Inverter Block Diagram_M420



Connecting Inverter to PLC???

INPUT OUTPUT PLC CONNECTING

Inverter Block Diagram_FC50



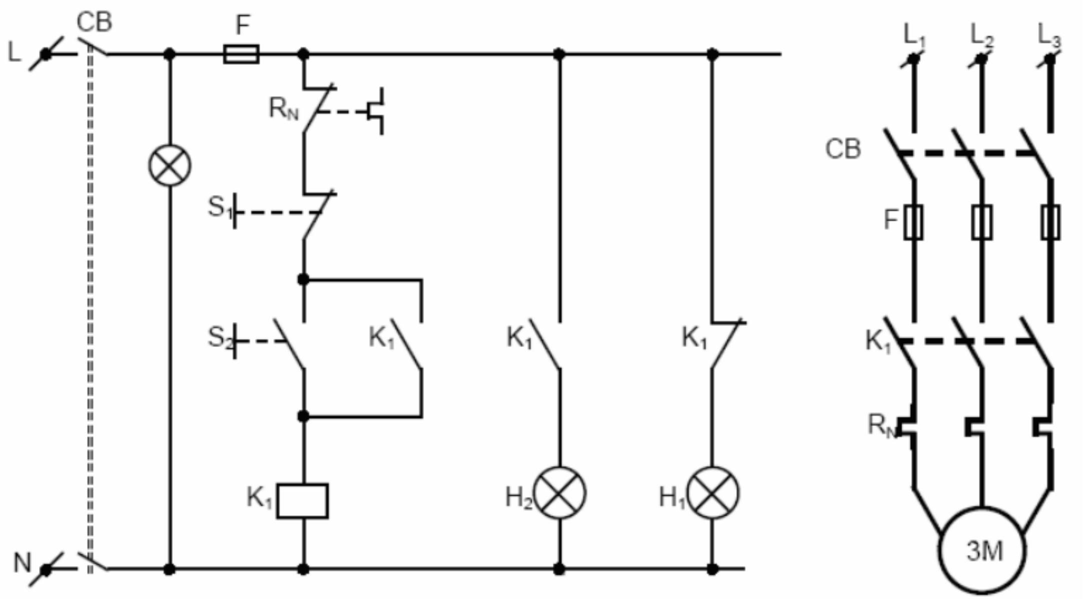
Connecting Inverter to PLC???

Connection diagram (200V/400V class 3-phase)

INPUT OUTPUT PLC CONNECTING

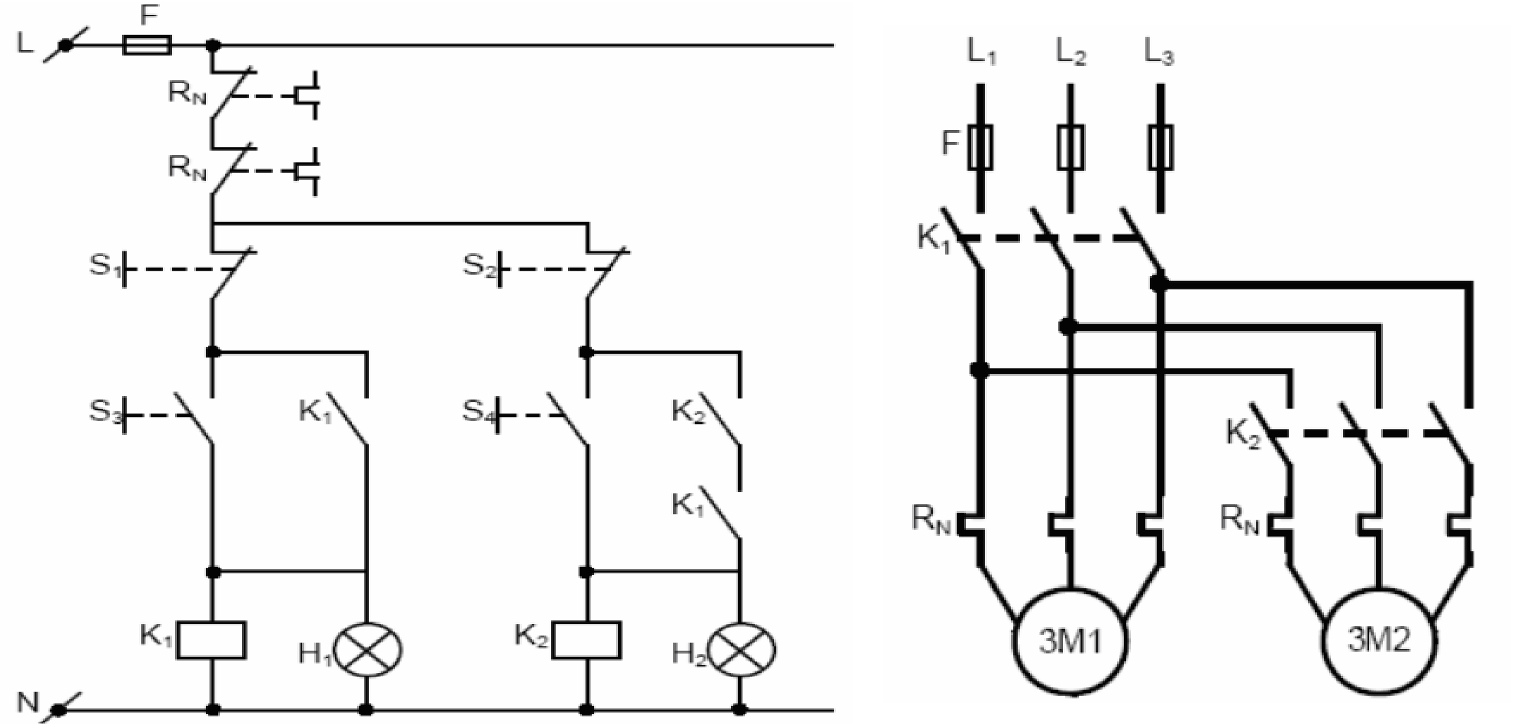
Replacing relay control circuits from Ex6 to Ex10 using PLC

Ex6



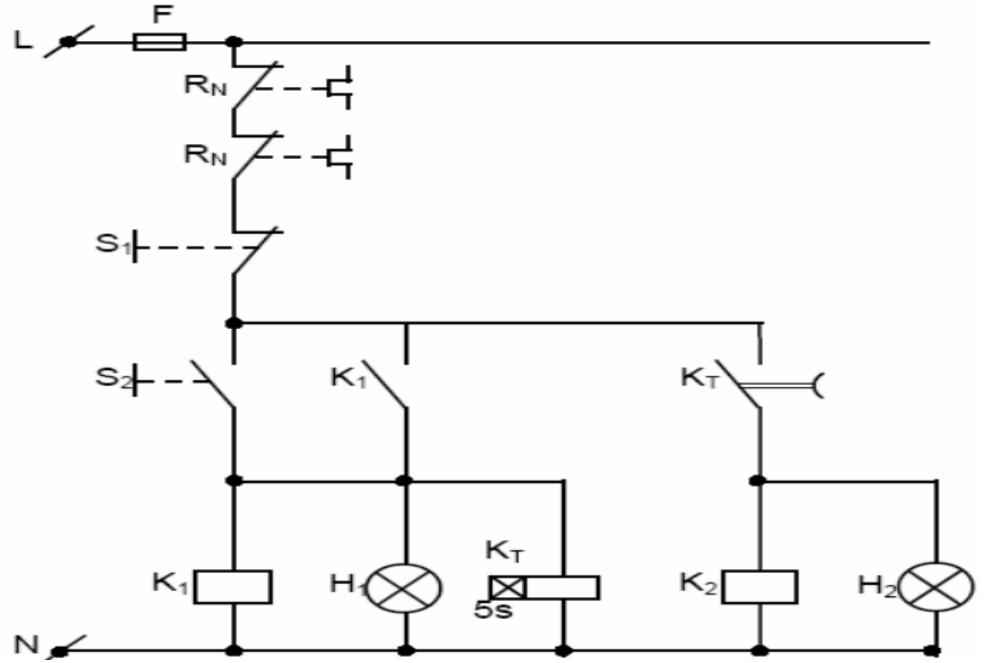
INPUT OUTPUT PLC CONNECTING

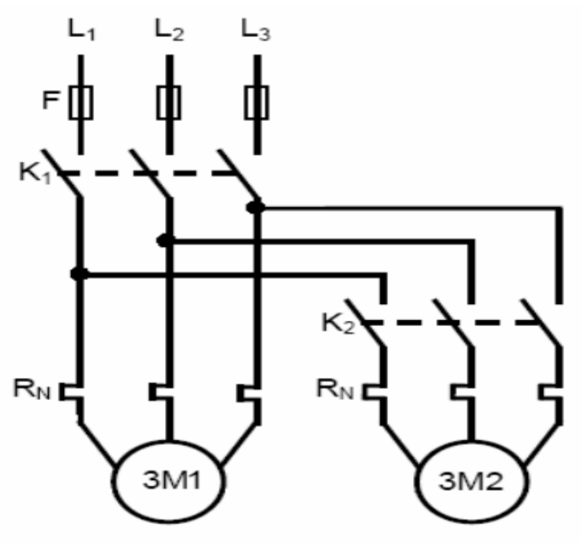
Ex7



INPUT OUTPUT PLC CONNECTING

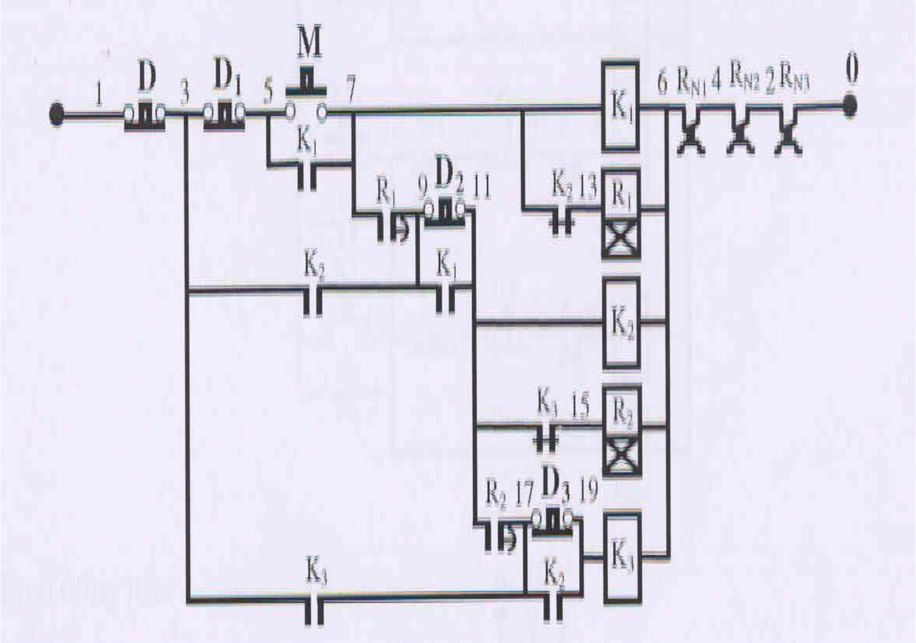
Ex8

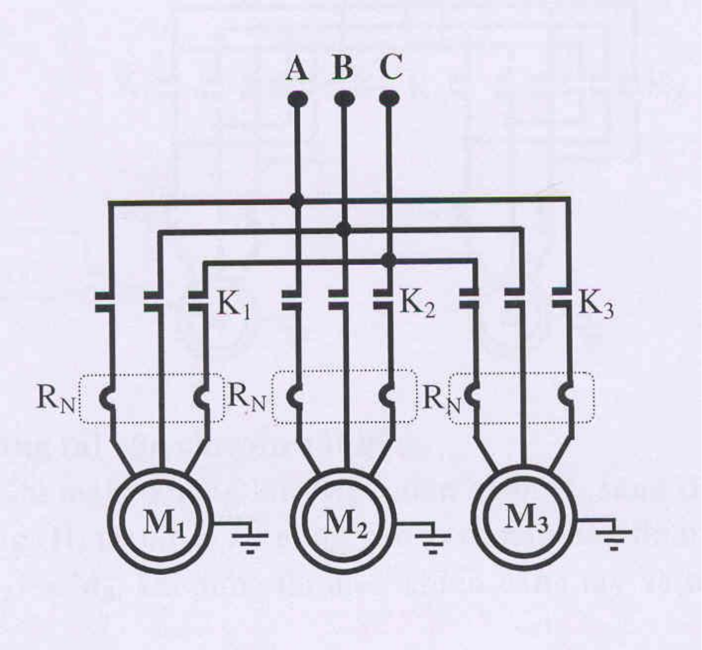




INPUT OUTPUT PLC CONNECTING

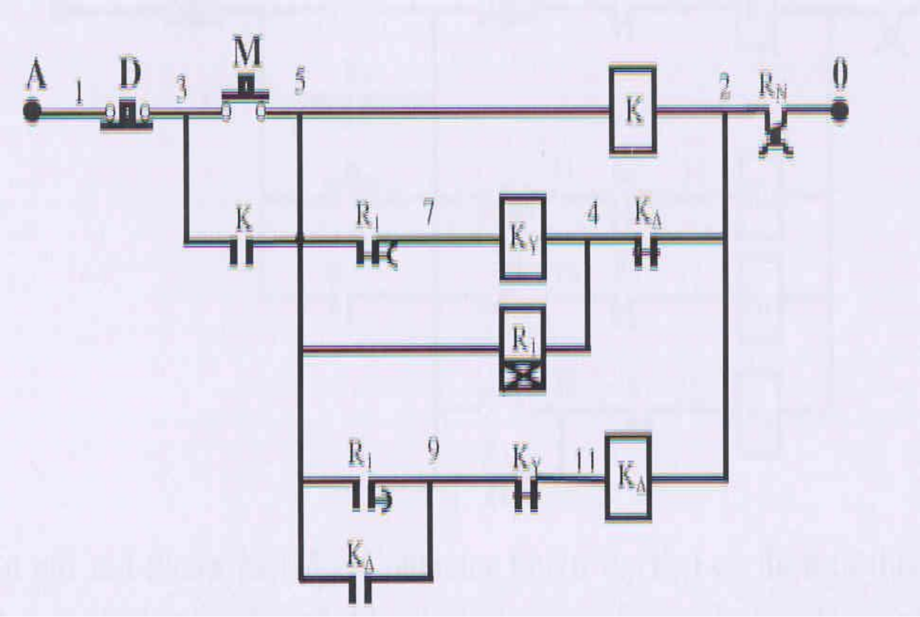
Ex9

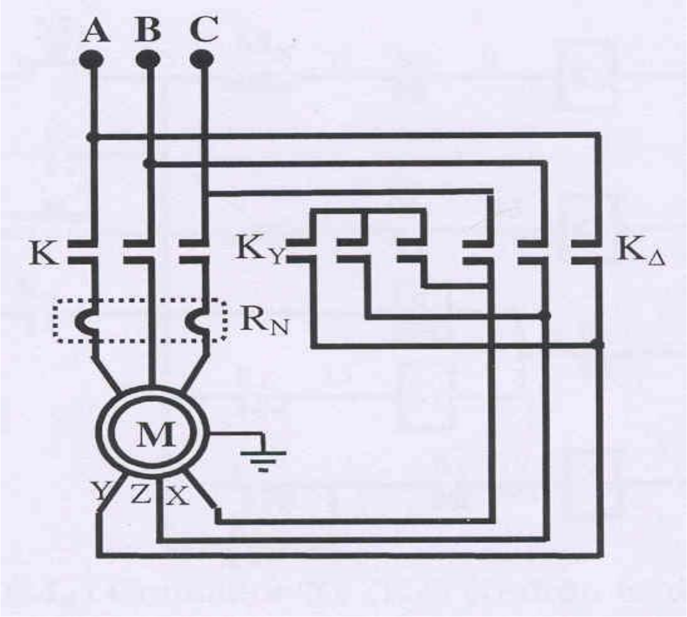




INPUT OUTPUT PLC CONNECTING

Ex10





COMPACTLOGIX CONTROLLER

COMPACTLOGIX L32E

- Bộ nhớ: 750kbytes.
- 1 port Ethernet/IP, 1 port RS-232.
- EtherNet/IP, DeviceNet.
- Relay Ladder, FBD, Structured text,

Sequential function block.

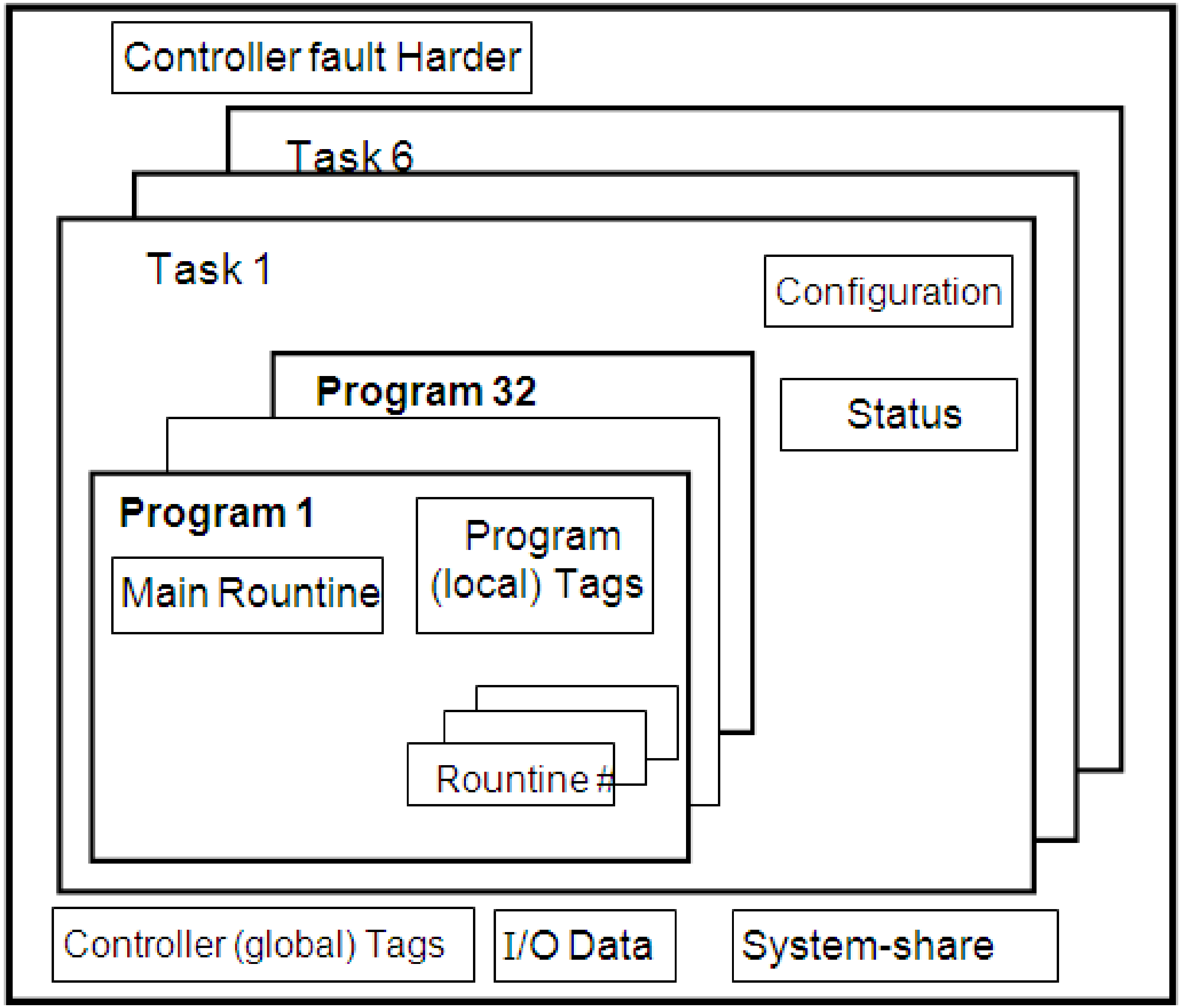
- Số module mở rộng: 16.

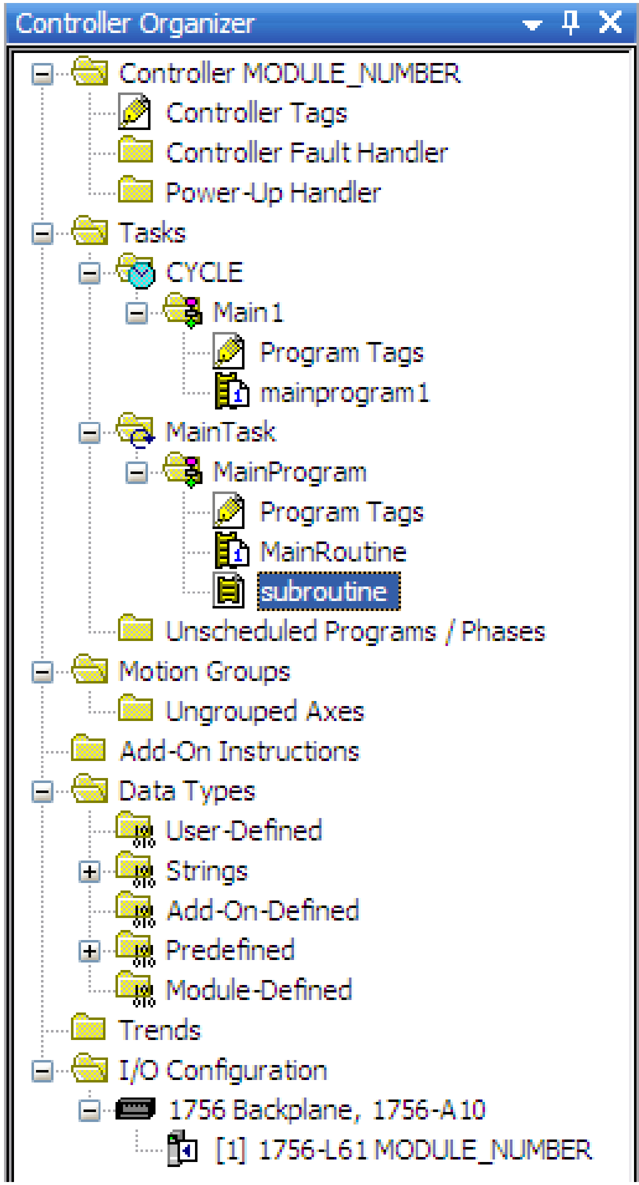
CONTROLLOGIX CONTROLLER

CONTROLLOGIX L61

- Bộ nhớ: 2MB.
- 1 port Ethernet/IP, 1 port RS-232.
- EtherNet/IP,Controlnet, DeviceNet.
- Relay Ladder, FBD, Structured text, Sequential function block.
- Số module mở rộng: 18

CONTROLLER ORGANIZER





- Controller MODULE_NUMBER
  - Controller Tags
  - Controller Fault Handler
  - Power-Up Handler
- Tasks
  - CYCLE
    - Main1
      - Program Tags
      - mainprogram1
  - MainTask
    - MainProgram
      - Program Tags
      - MainRoutine
      - subroutine
  - Unscheduled Programs / Phases
- Motion Groups
  - Ungrouped Axes
- Add-On Instructions
- Data Types
  - User-Defined
  - Strings
  - Add-On-Defined
  - Predefined
  - Module-Defined
- Trends
- I/O Configuration
  - 1756 Backplane, 1756-A10
    - [1] 1756-L61 MODULE_NUMBER

CONTROLLER ORGANIZER

Controller Organizer includes following elements

Controller fault handler is executed whenever the CPU is fault.

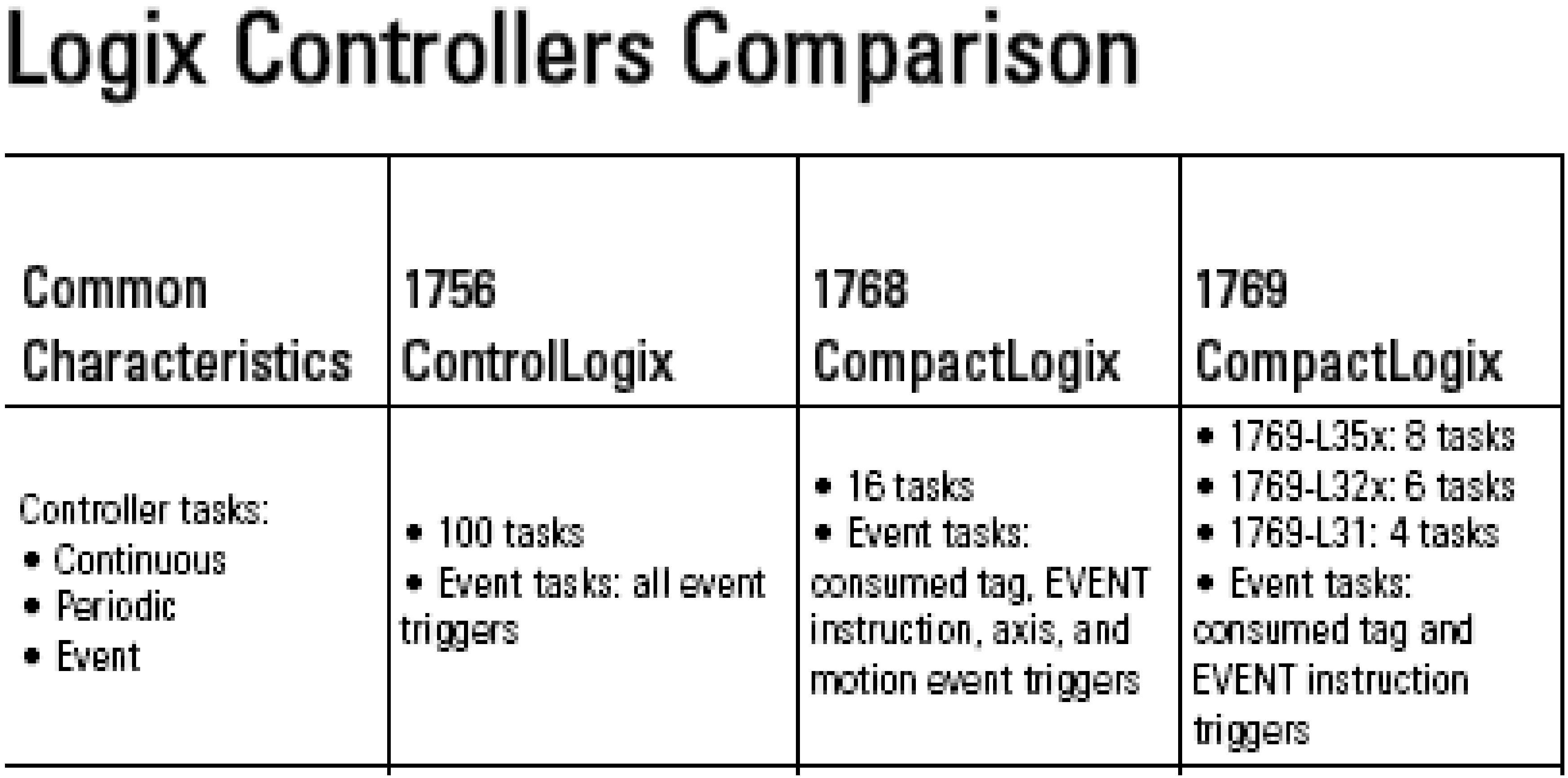
Power Up handler is executed as the CPU is powered.

Task includes three types:

- **Continuous Task** is executed all the time, a project has only a continuous task
- **Periodic Task** performs function at a specific time, whenever the time for periodic task expires.
- **Event Task** performs a function only when a specific event occurs.

Tag is a memory (data variable) in controller, includes controller tag and local tag

TASK IN CONTROLLER

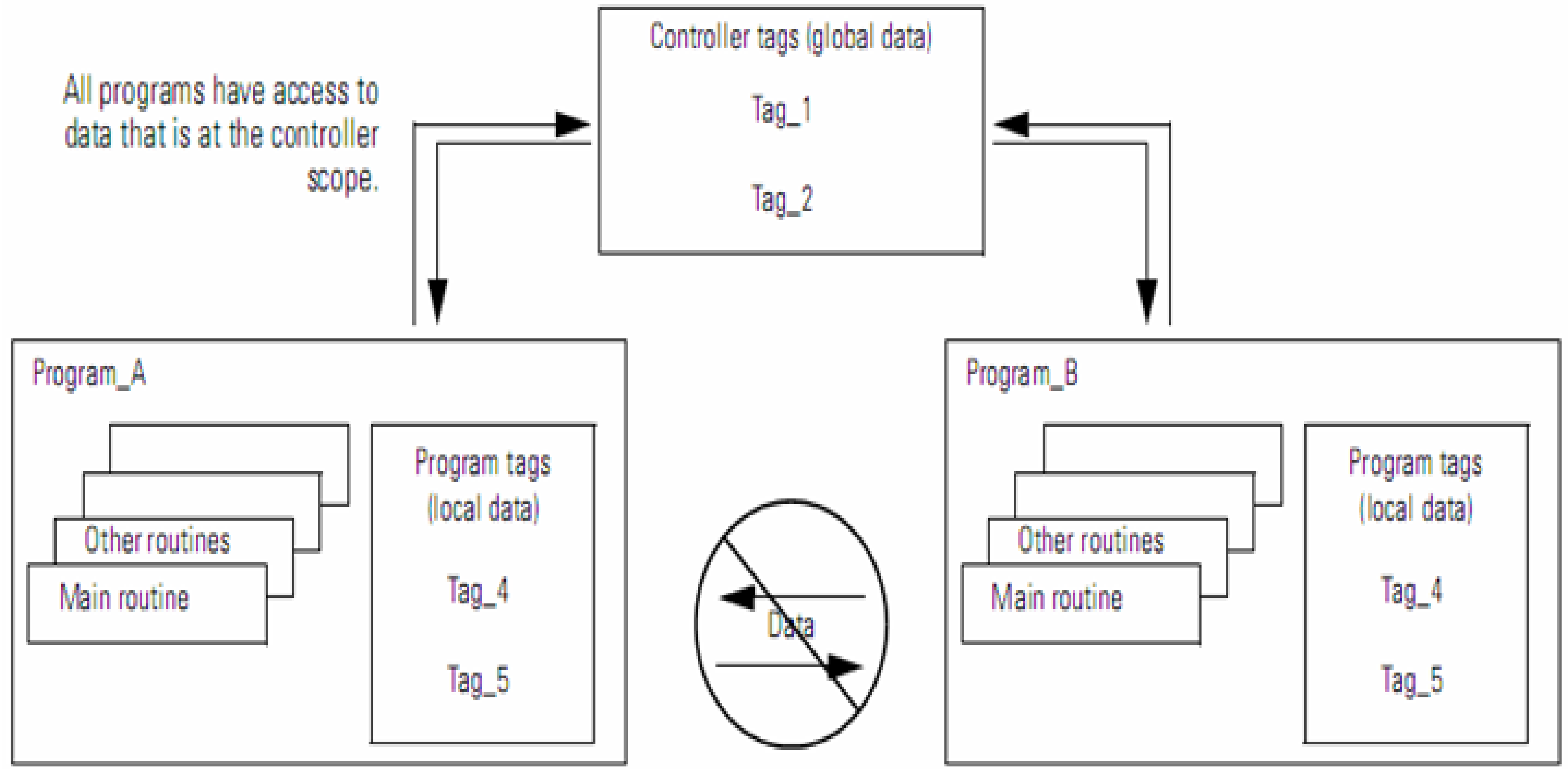


Logix Controllers Comparison

Common Characteristics	1756 ControlLogix	1768 CompactLogix	1769 CompactLogix
Controller tasks: • Continuous • Periodic • Event	• 100 tasks • Event tasks: all event triggers	• 16 tasks • Event tasks: consumed tag, EVENT instruction, axis, and motion event triggers	• 1769-L35x: 8 tasks • 1769-L32x: 6 tasks • 1769-L31: 4 tasks • Event tasks: consumed tag and EVENT instruction triggers

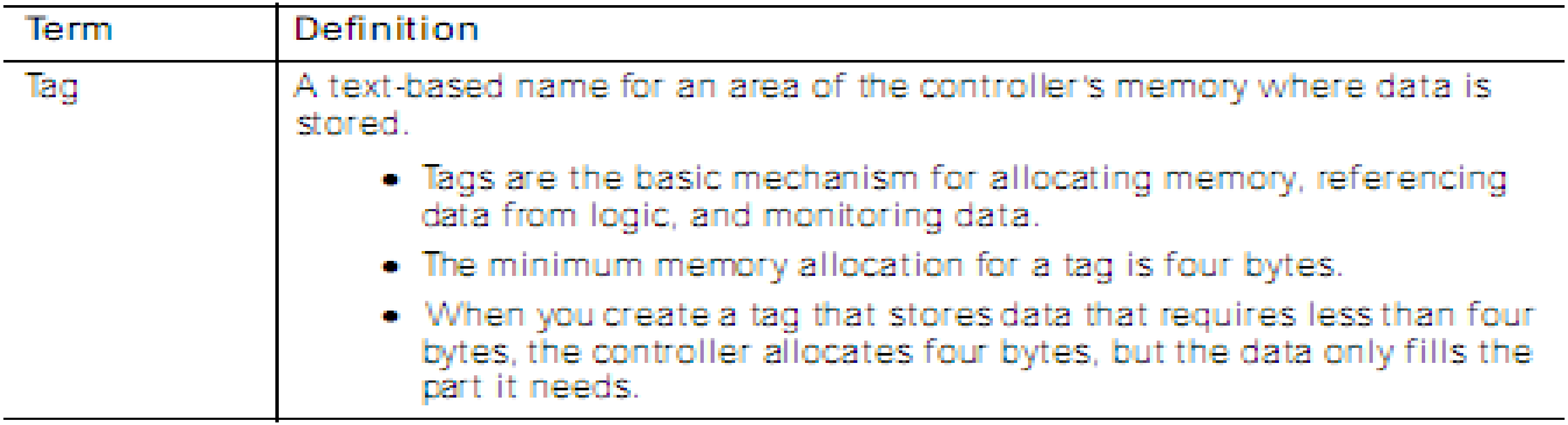
TAGS IN CONTROLLER

There are two types of tag: Controller tag(Global data) and Local tag(Program tag)

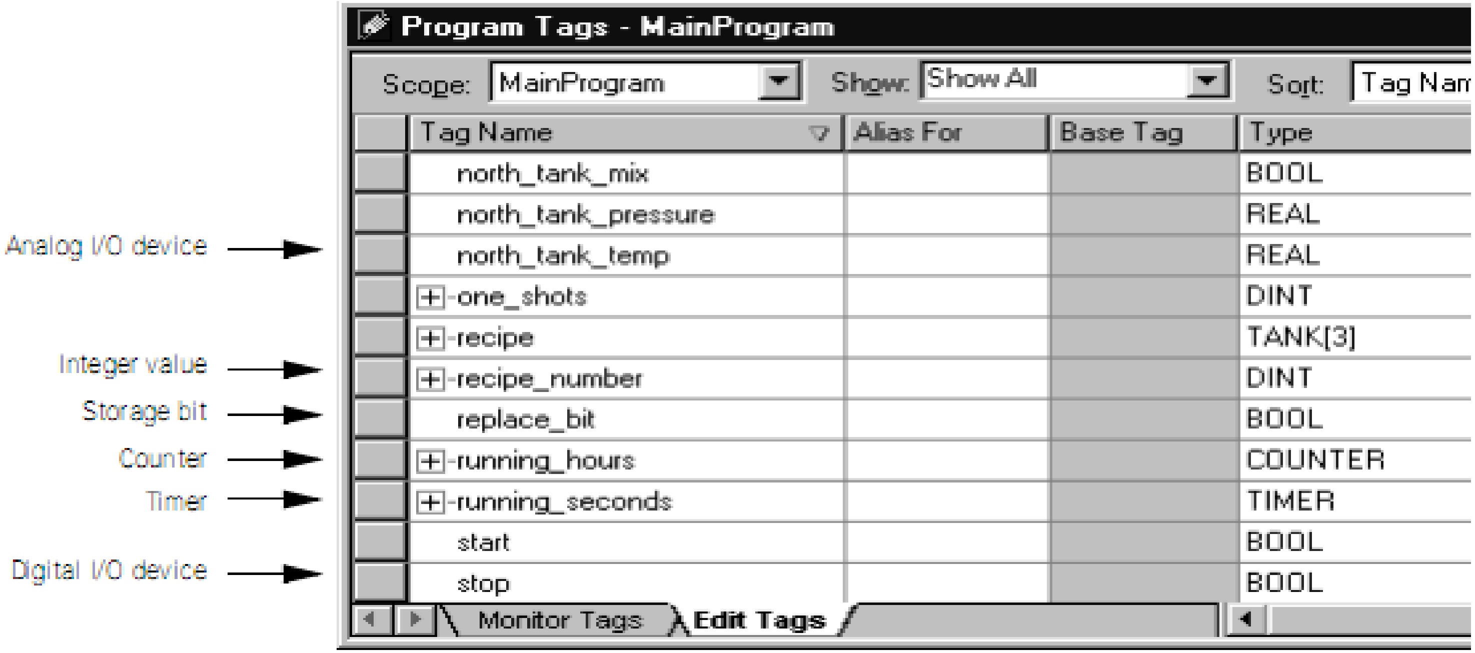

Controller tags (global data)
Tag_1
Tag_2
All programs have access to data that is at the controller scope.
Program_A
Other routines
Main routine
Program tags (local data)
Tag_4
Tag_5
Data
Program_B
Other routines
Main routine
Program tags (local data)
Tag_4
Tag_5

TAGS IN CONTROLLER

Tag is a data variable in a controller



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

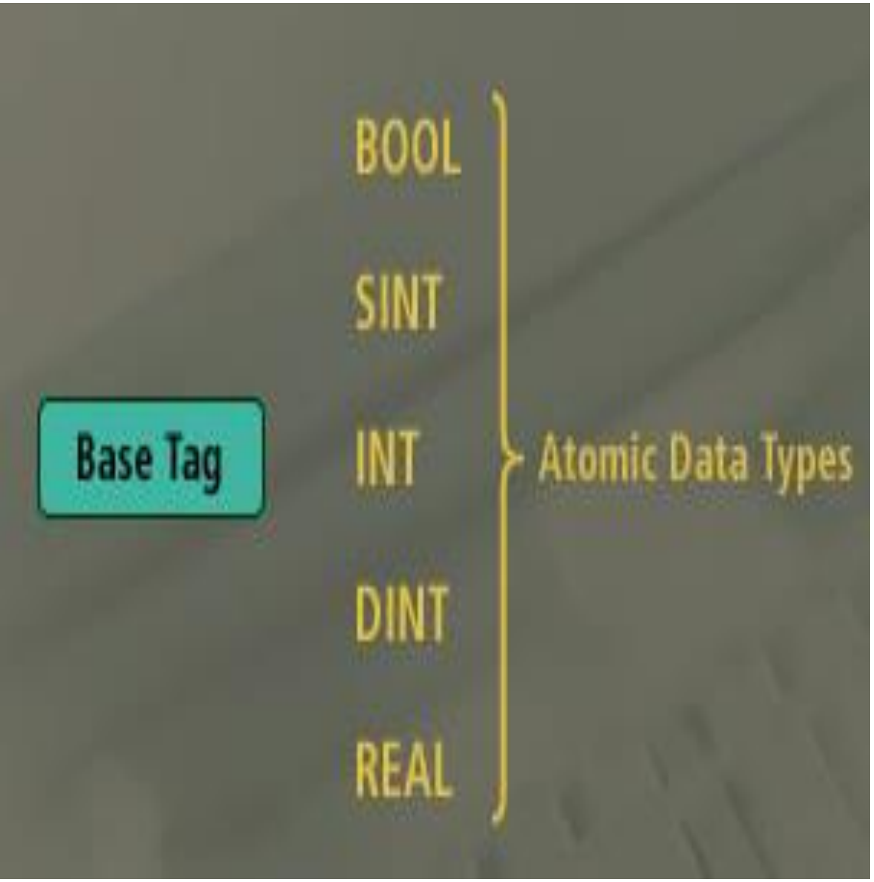


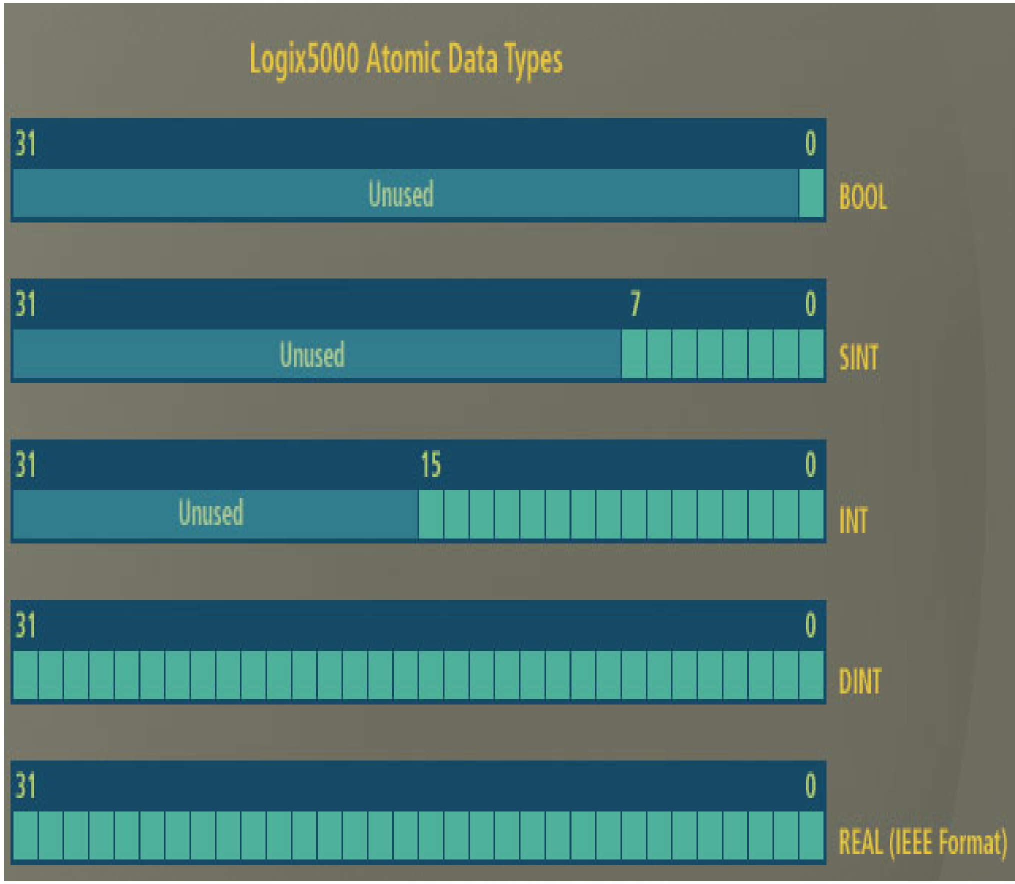
Tag Name	Alias For	Base Tag	Type
north_tank_mix			BOOL
north_tank_pressure			REAL
north_tank_temp			REAL
⊞-one_shots			DINT
⊞-recipe			TANK[3]
⊞-recipe_number			DINT
replace_bit			BOOL
⊞-running_hours			COUNTER
⊞-running_seconds			TIMER
start			BOOL
stop			BOOL

DINT
DINT
DINT
DINT
…….
DINT
DINT

TAG IN CONTROLLER

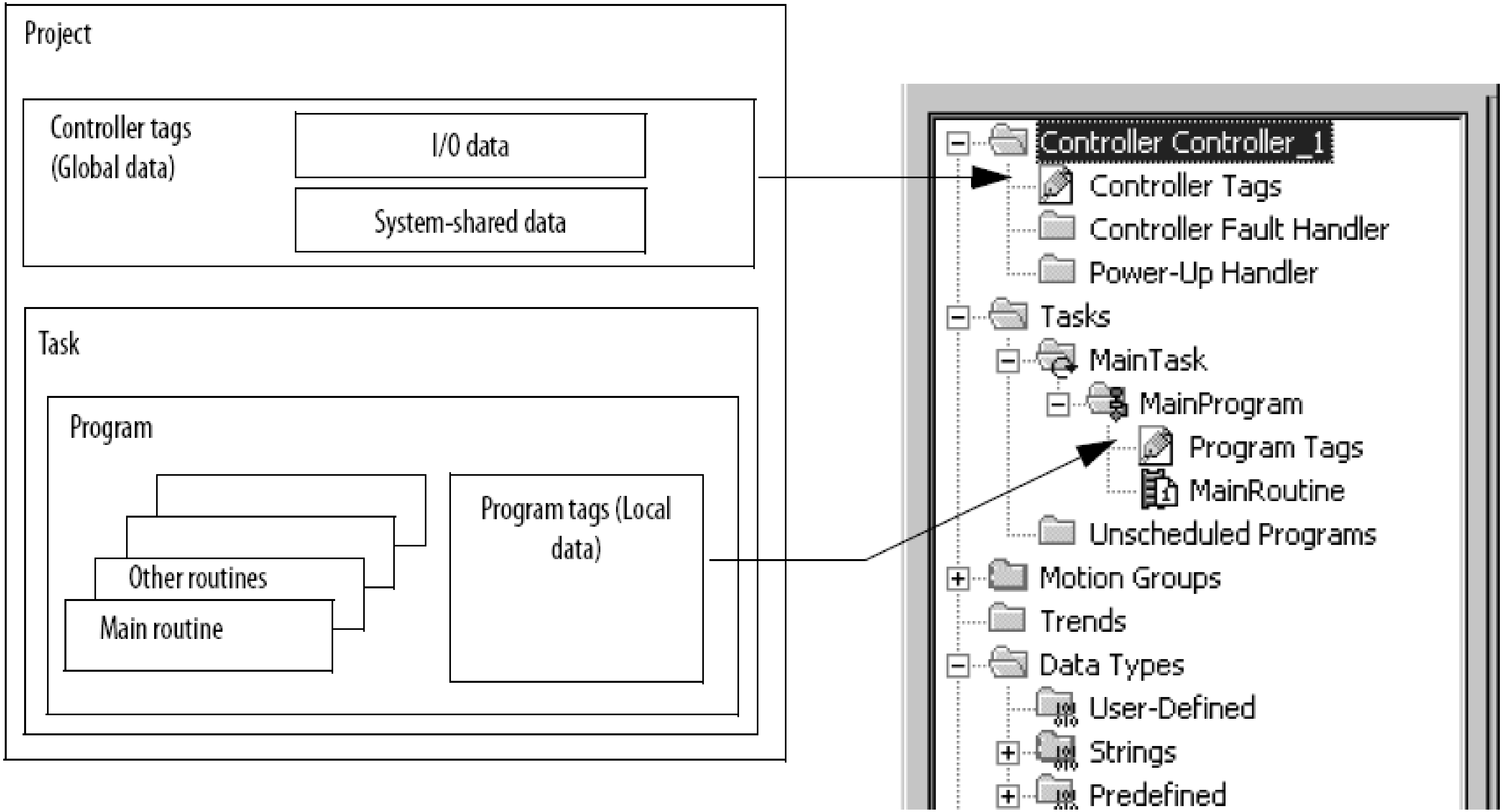
Data types of tag





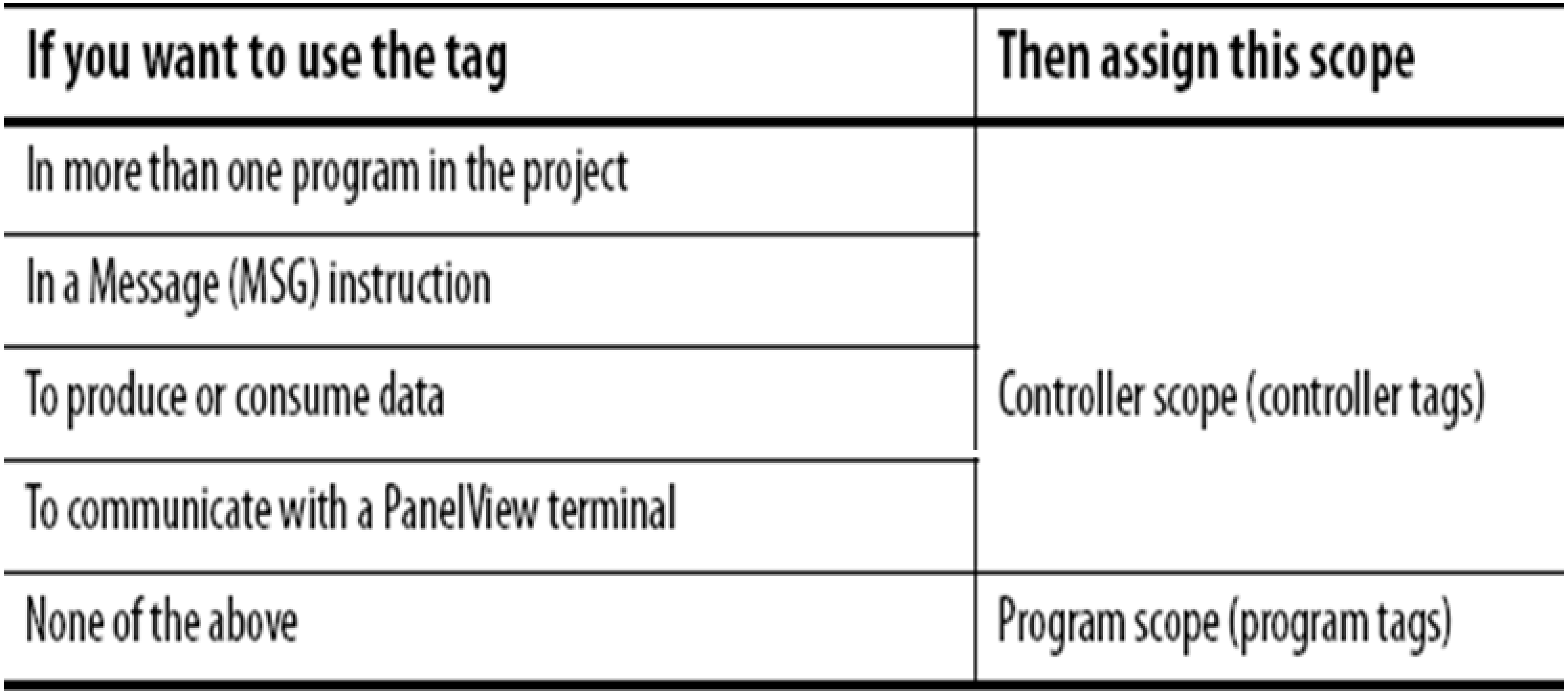
TAGS IN CONTROLLER

Controller Tags & Program Tags



TAGS IN CONTROLLER

Using Controller tags or Program tags

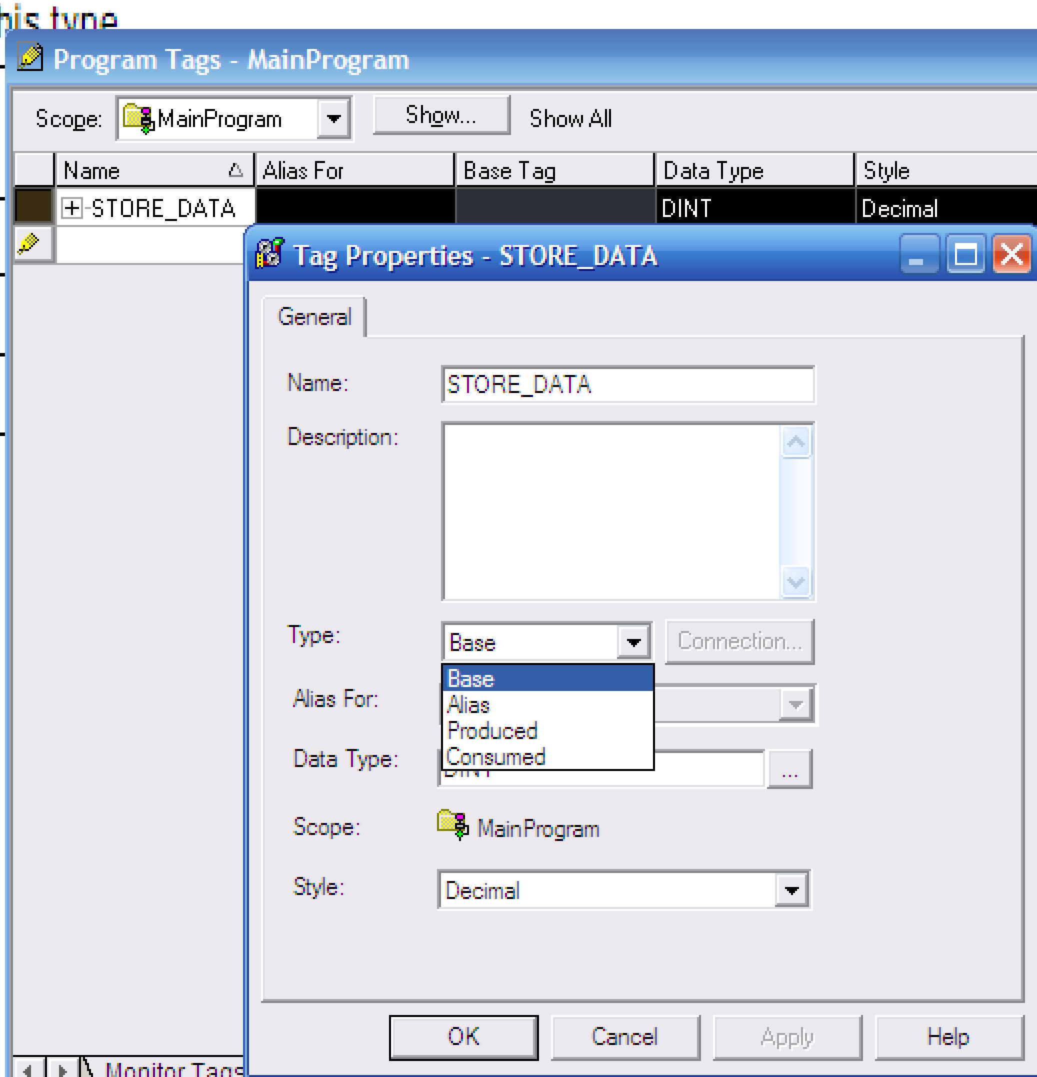


If you want to use the tag	Then assign this scope
In more than one program in the project	Controller scope (controller tags)
In a Message (MSG) instruction	
To produce or consume data	
To communicate with a PanelView terminal	
None of the above	Program scope (program tags)

TYPE TAG IN CONTROLLER

Type Tag defines how the tag operates within a project, There are four types of tag: Base, Alias, Produced and Consumed

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



Program Tags - MainProgram

Scope: MainProgram

Name	Alias For	Base Tag	Data Type	Style
STORE_DATA			DINT	Decimal

Tag Properties - STORE_DATA

General

Name: STORE_DATA

Description:

Type: Base

Alias For:

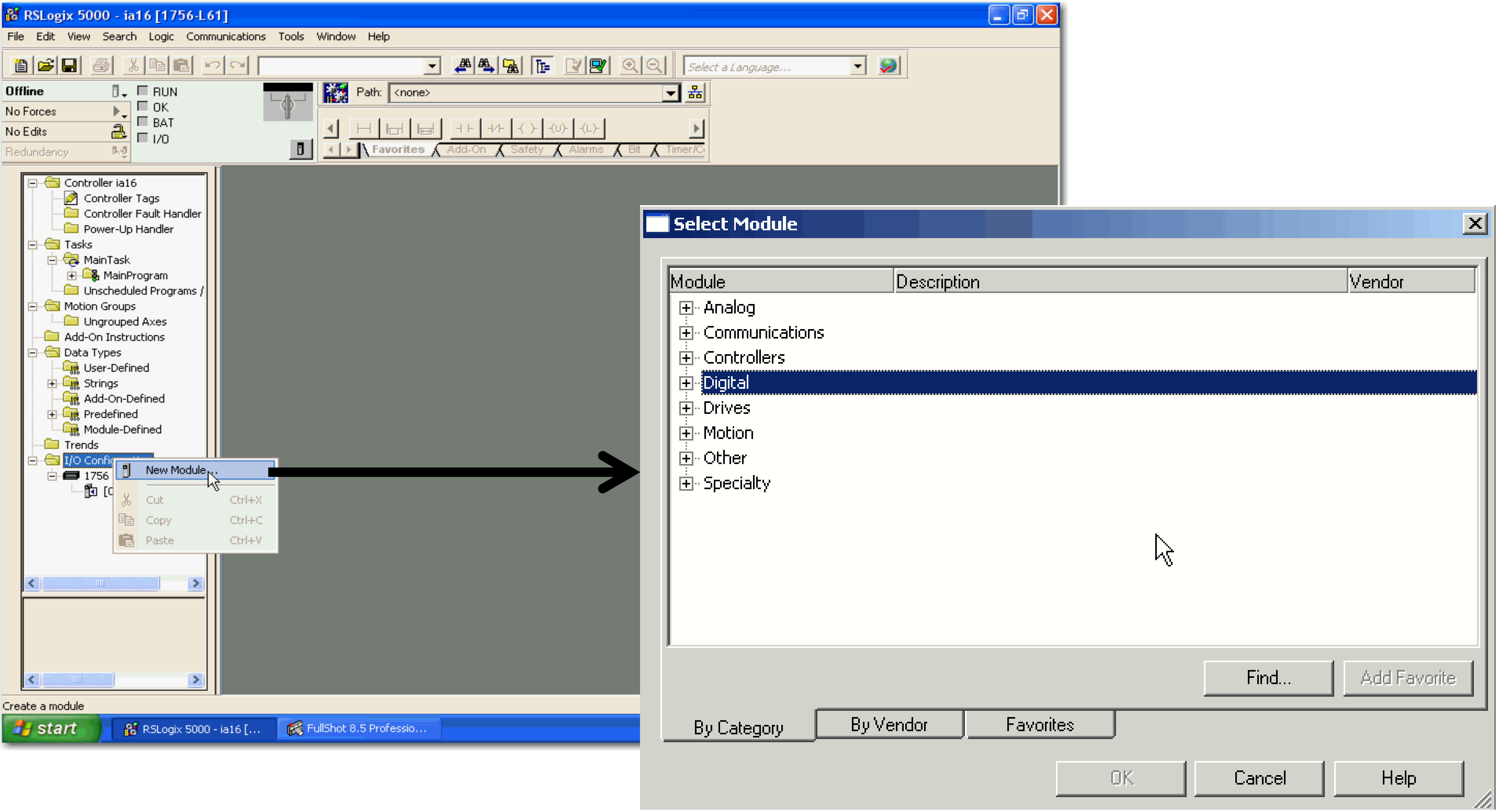
Data Type: DINT

Scope: MainProgram

Style: Decimal

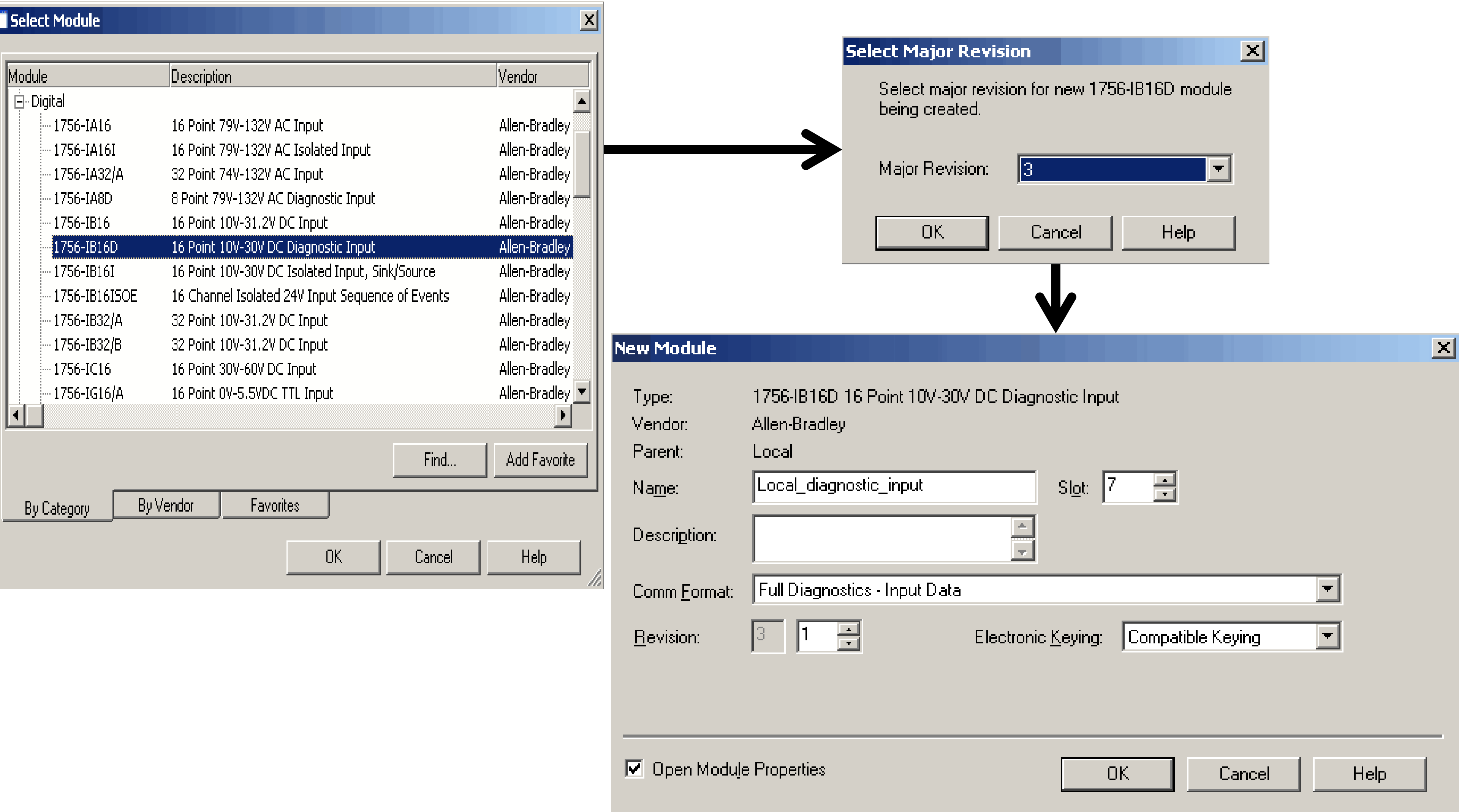
COMMUNICATION WITH I/O

Create a new Module: On the Controller Organizer, right-click I/O Configuration and choose New Module.



COMMUNICATION WITH I/O

Insert a new Module, Enter an Apropriate name, Major Revision and Electronic Keying



Select Module

Module	Description	Vendor
⊟ Digital		
1756-IA16	16 Point 79V-132V AC Input	Allen-Bradley
1756-IA16I	16 Point 79V-132V AC Isolated Input	Allen-Bradley
1756-IA32/A	32 Point 74V-132V AC Input	Allen-Bradley
1756-IA8D	8 Point 79V-132V AC Diagnostic Input	Allen-Bradley
1756-IB16	16 Point 10V-31.2V DC Input	Allen-Bradley
1756-IB16D	16 Point 10V-30V DC Diagnostic Input	Allen-Bradley
1756-IB16I	16 Point 10V-30V DC Isolated Input, Sink/Source	Allen-Bradley
1756-IB16ISOE	16 Channel Isolated 24V Input Sequence of Events	Allen-Bradley
1756-IB32/A	32 Point 10V-31.2V DC Input	Allen-Bradley
1756-IB32/B	32 Point 10V-31.2V DC Input	Allen-Bradley
1756-IC16	16 Point 30V-60V DC Input	Allen-Bradley
1756-IG16/A	16 Point 0V-5.5VDC TTL Input	Allen-Bradley

Select Major Revision

Select major revision for new 1756-IB16D module being created.

Major Revision: 3

New Module

Type: 1756-IB16D 16 Point 10V-30V DC Diagnostic Input

Vendor: Allen-Bradley

Parent: Local

Name: Local_diagnostic_input

Slot: 7

Description:

Comm Format: Full Diagnostics - Input Data

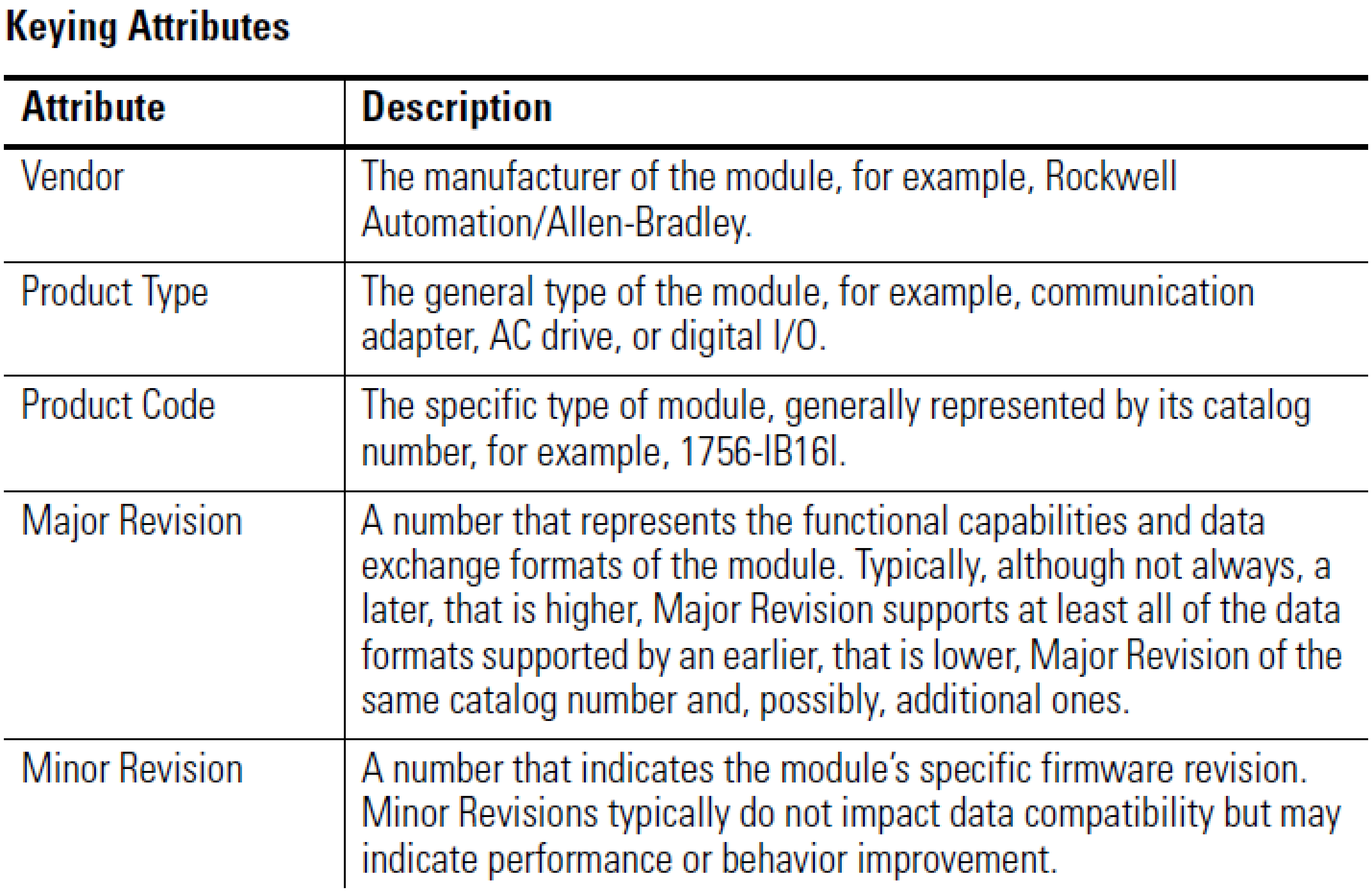
Revision: 3 1

Electronic Keying: Compatible Keying

☑ Open Module Properties

COMMUNICATION WITH I/O

Electronic Keying: Compares expected module in I/O configuration and physical module



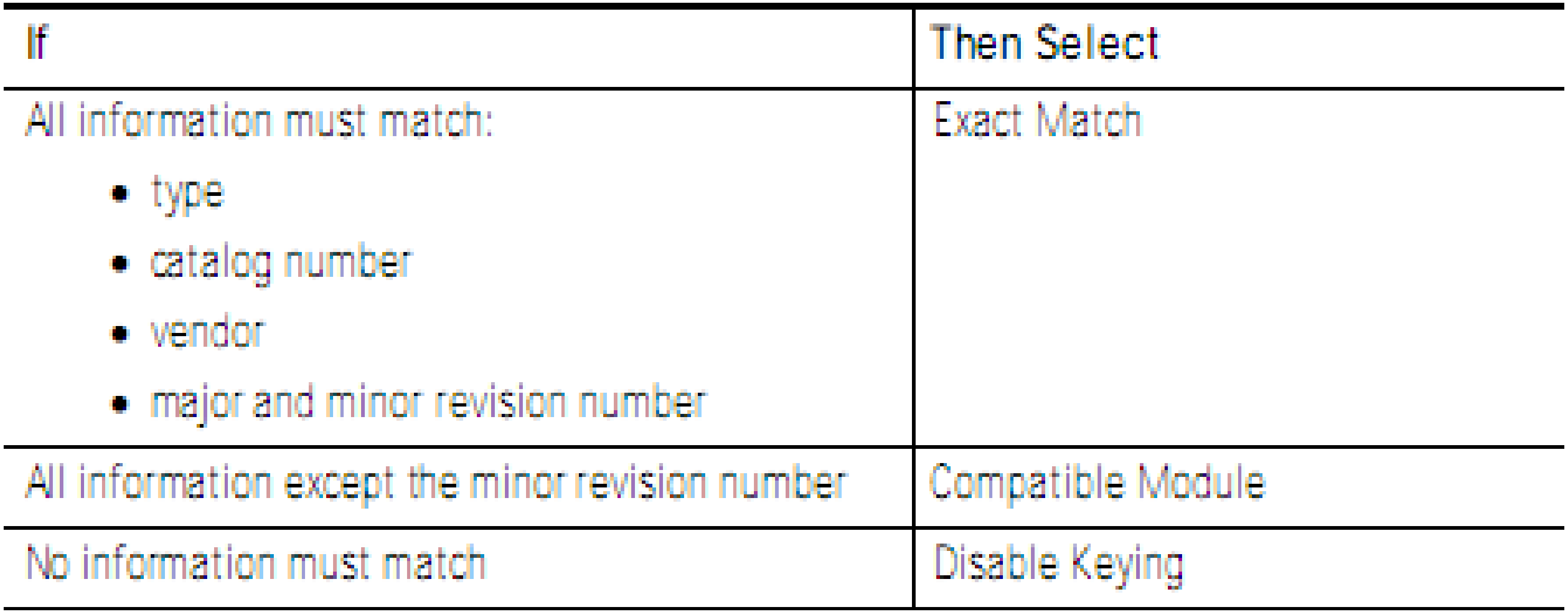
Keying Attributes

Attribute	Description
Vendor	The manufacturer of the module, for example, Rockwell Automation/Allen-Bradley.
Product Type	The general type of the module, for example, communication adapter, AC drive, or digital I/O.
Product Code	The specific type of module, generally represented by its catalog number, for example, 1756-IB16I.
Major Revision	A number that represents the functional capabilities and data exchange formats of the module. Typically, although not always, a later, that is higher, Major Revision supports at least all of the data formats supported by an earlier, that is lower, Major Revision of the same catalog number and, possibly, additional ones.
Minor Revision	A number that indicates the module's specific firmware revision. Minor Revisions typically do not impact data compatibility but may indicate performance or behavior improvement.

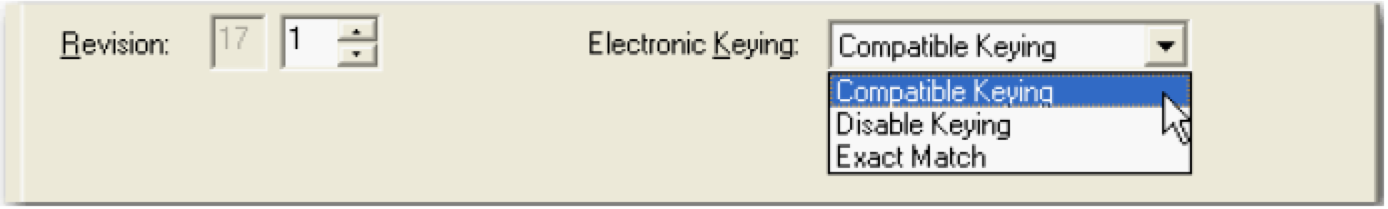
SETTING ELECTRONIC KEYING

Electronic Keying: Protect a system against the accidental placement of the wrong module in the slot

The Electronic Key determines how closely any module in a slot must match the configuration for that slot



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



Revision: 17 1

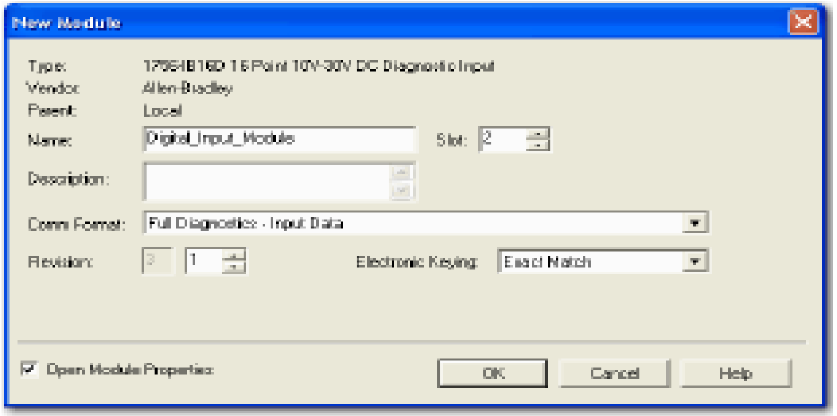
Electronic Keying: Compatible Keying

SETTING ELECTRONIC KEYING

Exact Match: All information must match

Module Configuration

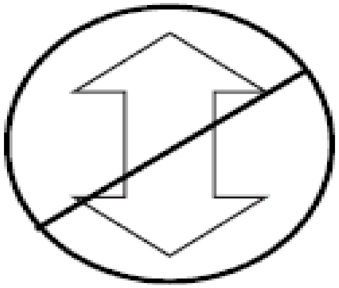
Vendor = Allen-Bradley
Product Type = Digital Input Module
Catalog Number = 1756-IB16D
Major Revision = 3
Minor Revision = 1



New Module

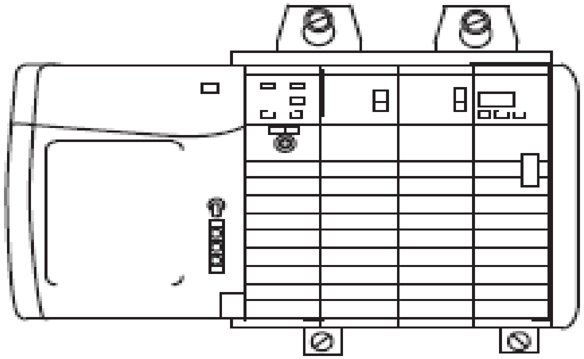
Type: 1756-IB16D 16 Point 10V-30V DC Diagnostic Input
Vendor: Allen-Bradley
Parent: Local
Name: Digital_Input_Module
Slot: 2
Description:
Comm Format: Full Diagnostics - Input Data
Revision: 3 1
Electronic Keying: Exact Match
☑ Open Module Properties

Communication is prevented



Physical Module

Vendor = Allen-Bradley
Product Type = Digital Input Module
Catalog Number = 1756-IB16D
Major Revision = 3
Minor Revision = 2



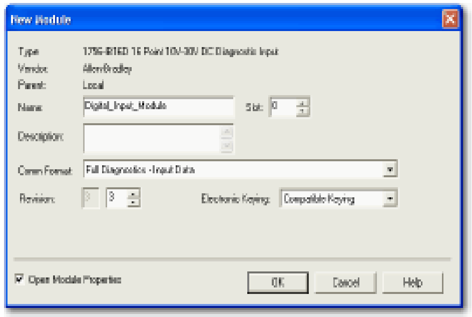
SETTING ELECTRONIC KEYING

Compatible Keying: All information excepte the minor revision number

- The module configuration is for a 1756-IB16D module with module revision 3.3. The physical module is a 1756-IB16D module with module revision 3.2. In this case, communication is prevented because the minor revision of the module is lower than expected and may not be compatible with 3.3.

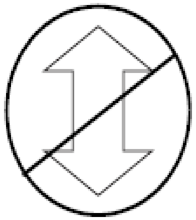
Module Configuration

Vendor = Allen-Bradley
Product Type = Digital Input Module
Catalog Number = 1756-IB16D
Major Revision = 3
Minor Revision = 3


New Module

Type: 1756-IB16D 16 Point 10V-30V DC Diagnostic Input
Vendor: Allen-Bradley
Parent: Local
Name: Digital_Input_Module
Slot: 0
Description:
Comm Format: Full Diagnostics - Input Data
Revision: 3 3
Electronic Keying: Compatible Keying

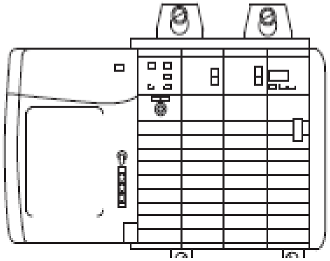
☑ Open Module Properties



Communication is prevented

Physical Module

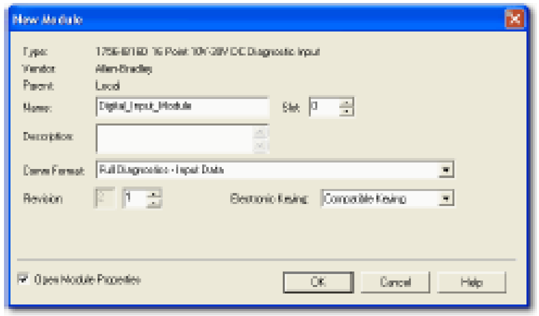
Vendor = Allen-Bradley
Product Type = Digital Input Module
Catalog Number = 1756-IB16D
Major Revision = 3
Minor Revision = 2



- The module configuration is for a 1756-IB16D module with module revision 2.1. The physical module is a 1756-IB16D module with module revision 3.2. In this case, communication is allowed because the major revision of the physical module is higher than expected and the module determines that it is compatible with the prior major revision.

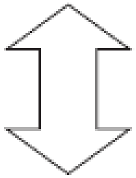
Module Configuration

Vendor = Allen-Bradley
Product Type = Digital Input Module
Catalog Number = 1756-IB16D
Major Revision = 2
Minor Revision = 1


New Module

Type: 1756-IB16D 16 Point 10V-30V DC Diagnostic Input
Vendor: Allen-Bradley
Parent: Local
Name: Digital_Input_Module
Slot: 0
Description:
Comm Format: Full Diagnostics - Input Data
Revision: 2 1
Electronic Keying: Compatible Keying

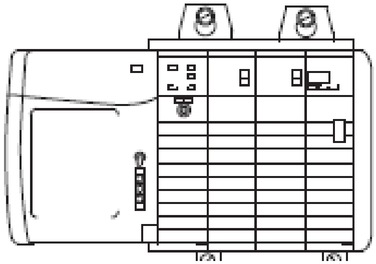
☑ Open Module Properties



Communication is allowed

Physical Module

Vendor = Allen-Bradley
Product Type = Digital Input Module
Catalog Number = 1756-IB16D
Major Revision = 3
Minor Revision = 2



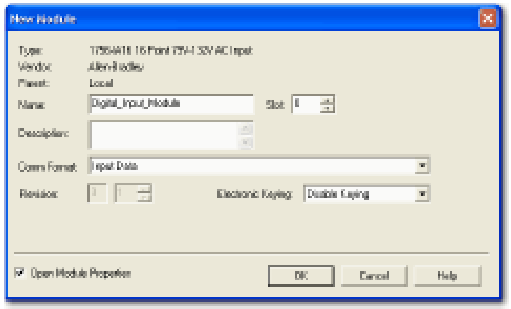
SETTING ELECTRONIC KEYING

Disable Keying:No information must match

- The module configuration is for a 1756-IA16 digital input module. The physical module is a 1756-IF16 analog input module. In this case, **communication is prevented because the analog module rejects the data formats that the digital module configuration requests**.

odule Configuration

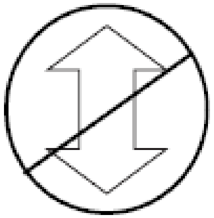
endor = Allen-Bradley
roduct Type = Digital Input
odule
atalog Number = 1756-IA16
ajor Revision = 3
inor Revision = 1


New Module

Type: 1756-IA16 16 Point 79V-132V AC Input
Vendor: Allen-Bradley
Parent: Local
Name: Digital_Input_Module   Slot: 0
Description:
Comm Format: Input Data
Revision: 3 1   Electronic Keying: Disable Keying

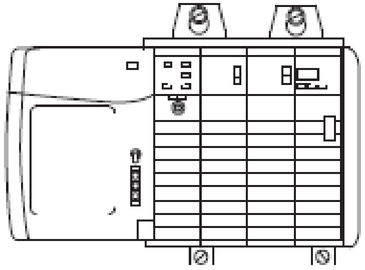
☑ Open Module Properties

Communication is prevented



hysical Module

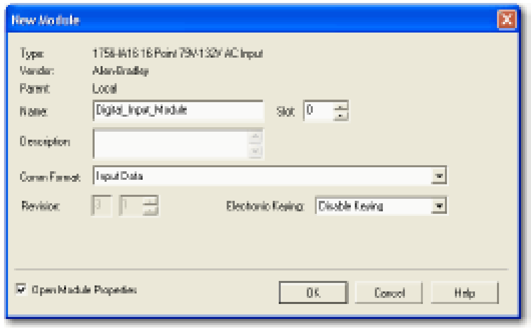
endor = Allen-Bradley
roduct Type = Analog Input
odule
atalog Number = 1756-IF16
ajor Revision = 3
inor Revision = 2



- The module configuration is for a 1756-IA16 digital input module. The physical module is a 1756-IB16 digital input module. In this case, communication is allowed because the two digital modules share common data formats.

Module Configuration

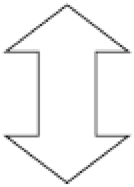
Vendor = Allen-Bradley
Product Type = Digital Input
Module
Catalog Number = 1756-IA16
Major Revision = 2
Minor Revision = 1


New Module

Type: 1756-IA16 16 Point 79V-132V AC Input
Vendor: Allen-Bradley
Parent: Local
Name: Digital_Input_Module   Slot: 0
Description:
Comm Format: Input Data
Revision: 2 1   Electronic Keying: Disable Keying

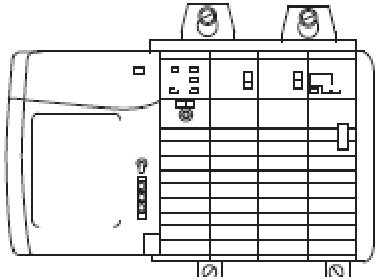
☑ Open Module Properties

Communication is allowed



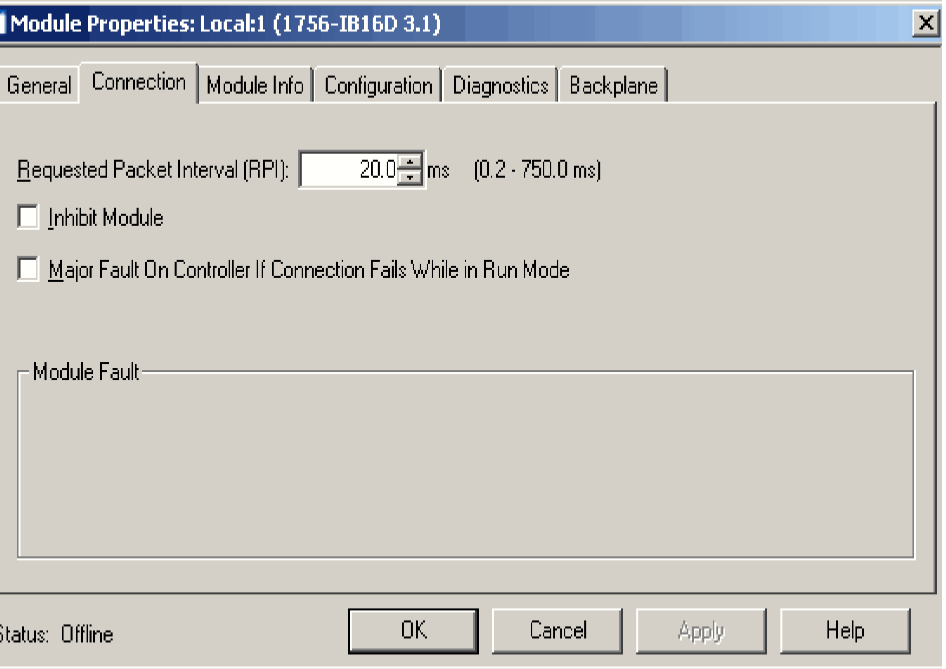
Physical Module

Vendor = Allen-Bradley
Product Type = Digital Input
Module
Catalog Number = 1756-IB16
Major Revision = 3
Minor Revision = 2



FEATURE SFECIFIC TO STANDARD INPUT MODULE

Setting RPI, COS, Diagnostics, Filter Time



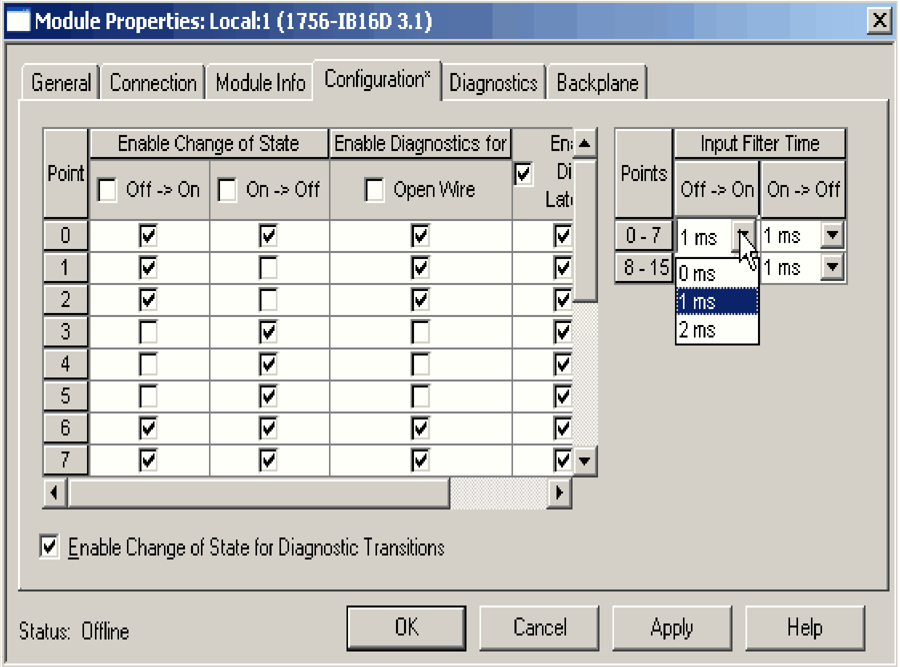
Requested Packet Interval (RPI): 20.0 ms (0.2 - 750.0 ms)

□ Inhibit Module

□ Major Fault On Controller If Connection Fails While in Run Mode

Module Fault

Status: Offline



Point	Enable Change of State		Enable Diagnostics for	En Di Lat
	□ Off -> On	□ On -> Off	□ Open Wire	☑
0	☑	☑	☑	☑
1	☑	□	☑	☑
2	☑	□	☑	☑
3	□	☑	□	☑
4	□	☑	□	☑
5	□	☑	□	☑
6	☑	☑	☑	☑
7	☑	☑	☑	☑

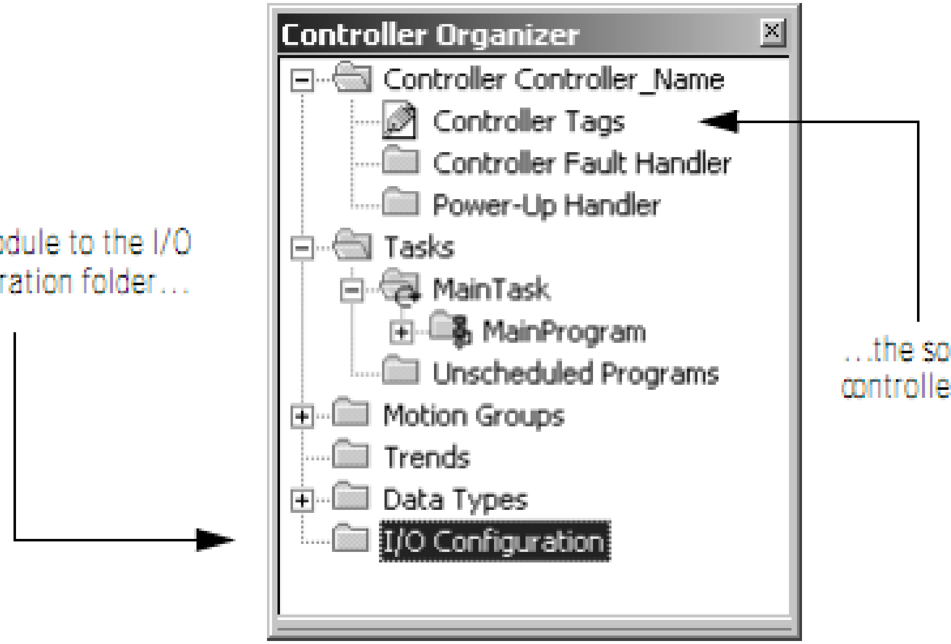
Points	Input Filter Time	
	Off -> On	On -> Off
0 - 7	1 ms	1 ms
8 - 15	0 ms 1 ms 2 ms	1 ms

☑ Enable Change of State for Diagnostic Transitions

Status: Offline

ADDRESS I/O DATA

I/O information is presented as a set of tag



When you add a module to the I/O Configuration folder…

…the software automatically creates controller-scoped tags for the module.

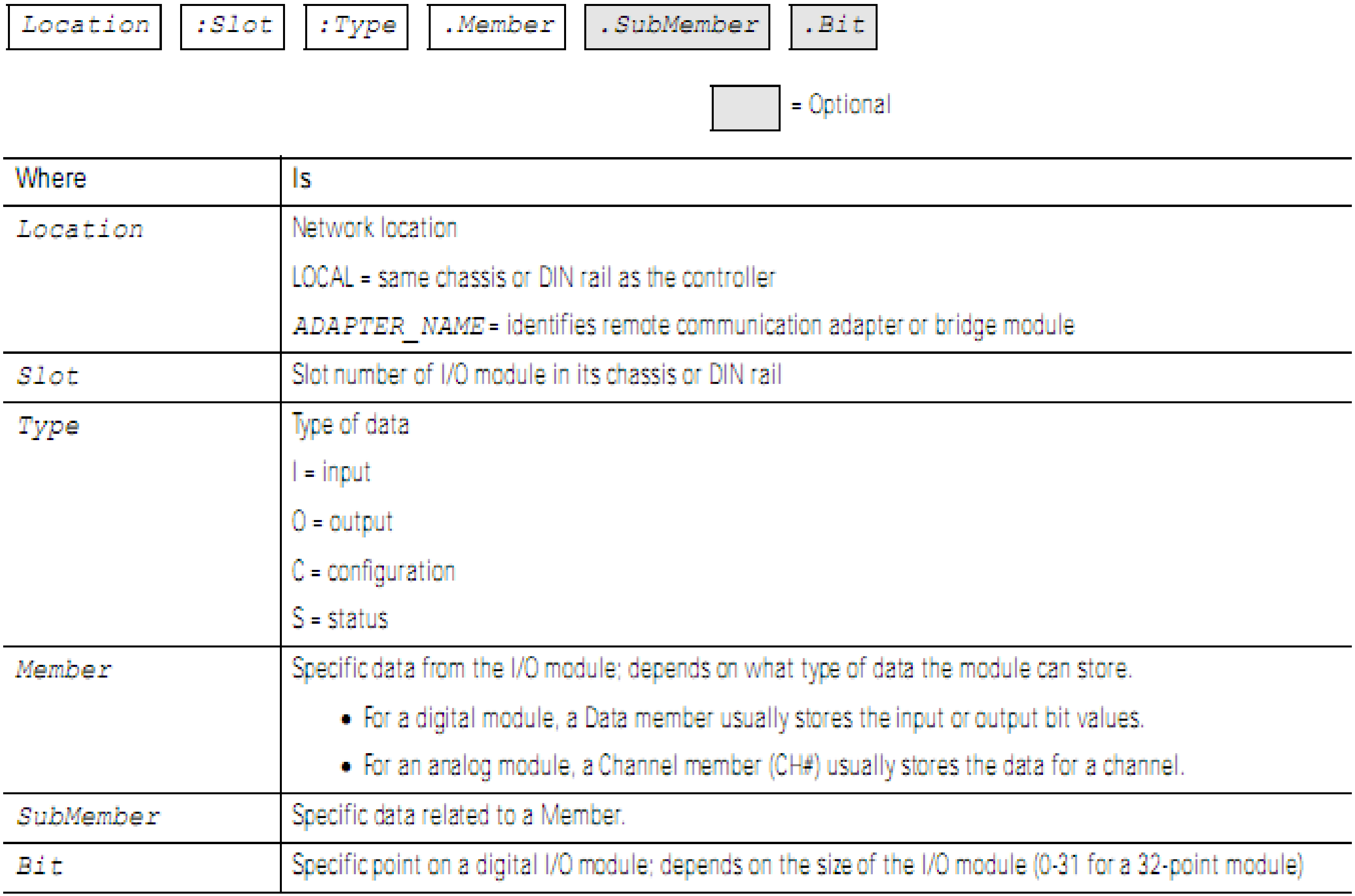
An I/O address follows this format:

`Location` `:Slot` `:Type` `.Member` `.SubMember` `.Bit`

▨ = Optional

ADDRESS I/O DATA

I/O information is presented as a set of tag



`Location` `:Slot` `:Type` `.Member` `.SubMember` `.Bit`

▨ = Optional

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

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

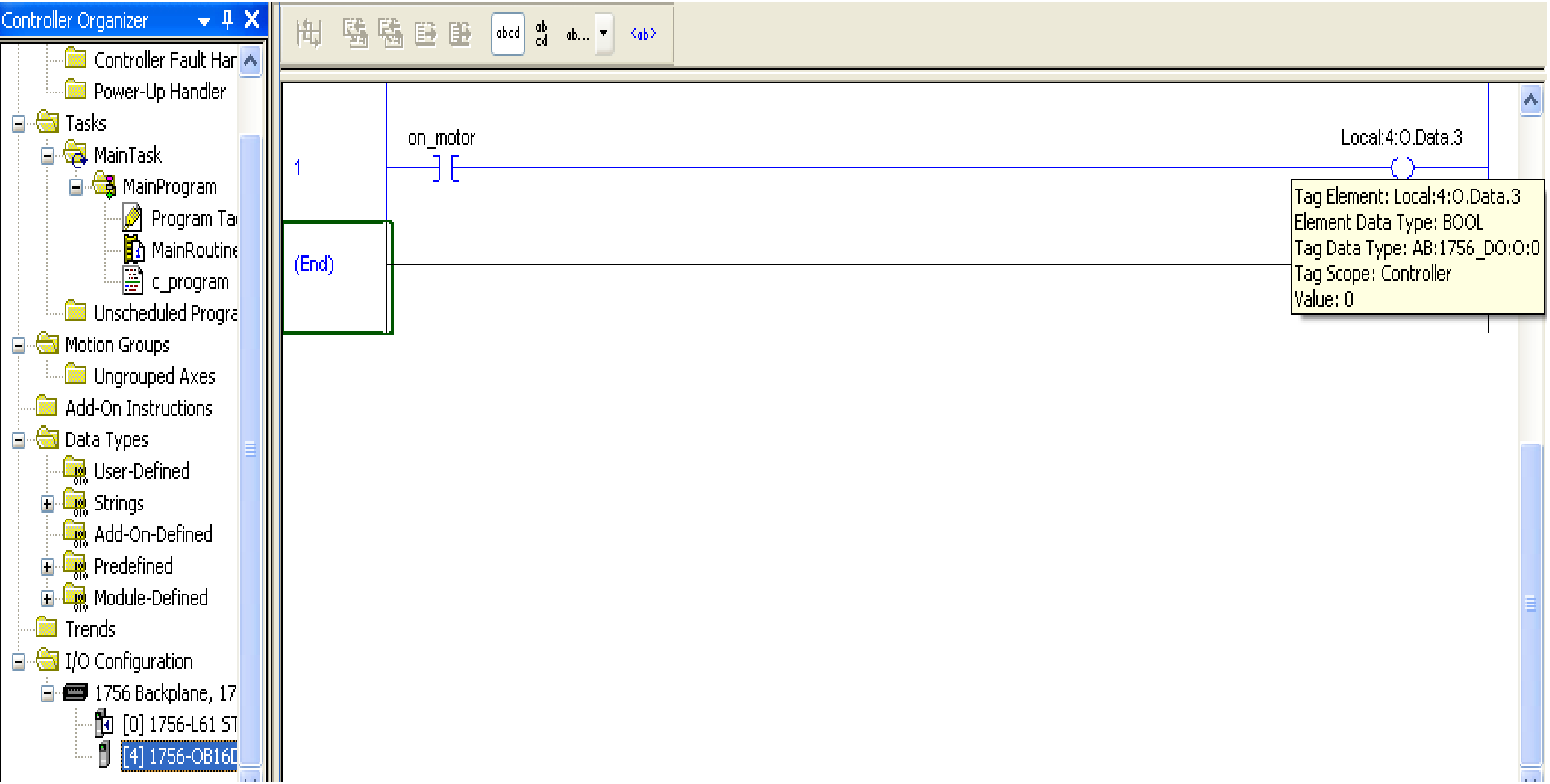
Member	Specific data from the I/O module; depends on what type of data the module can store. • For a digital module, a Data member usually stores the input or output bit values. • For an analog module, a Channel member (CH#) usually stores the data for a channel.
SubMember	Specific data related to a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

ADDRESS I/O DATA

I/O information is presented as a set of tag

Location :Slot :Type .Member .SubMember .Bit

▨ = Optional

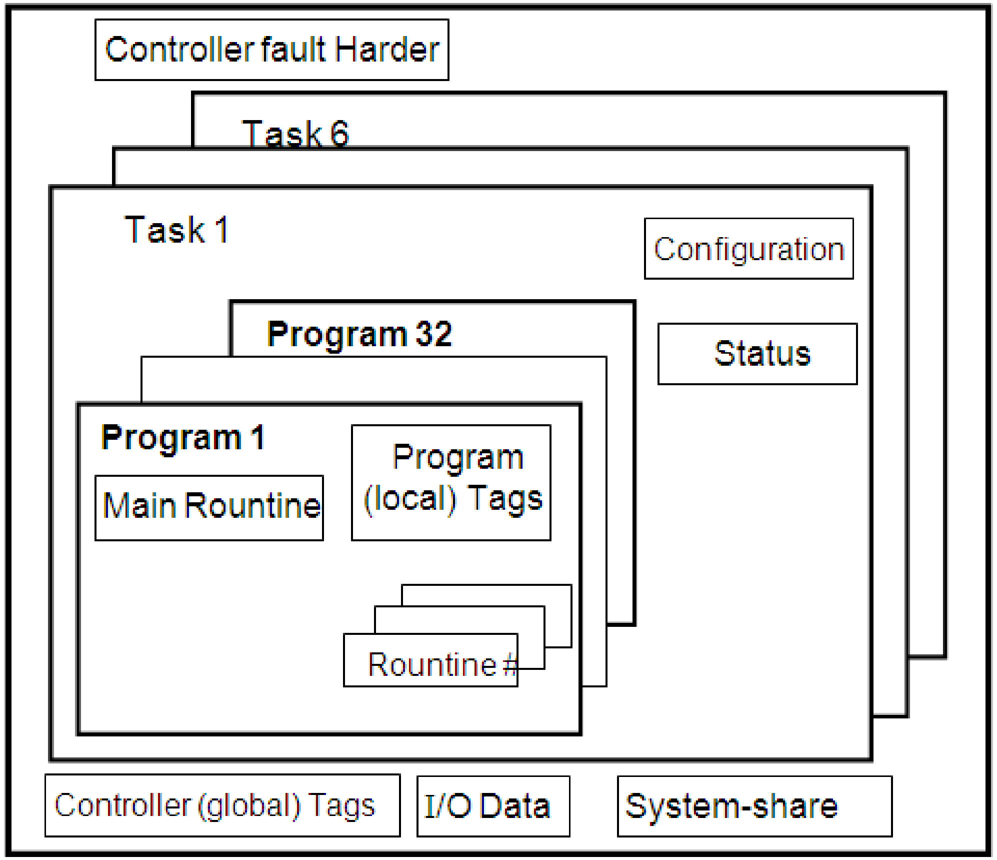


Tag Element: Local:4:O.Data.3
Element Data Type: BOOL
Tag Data Type: AB:1756_DO:O:0
Tag Scope: Controller
Value: 0

PROGRAM AND ROUTINE IN RSLOGIX

Tasks, Program and Rountine

- 32 programs in a task
- One main routine and many subroutines in a program
- Main routine is executed from program, sub is executed as called



Controller fault Harder

Task 6

Task 1

Configuration

Status

Program 32

Program 1

Main Rountine

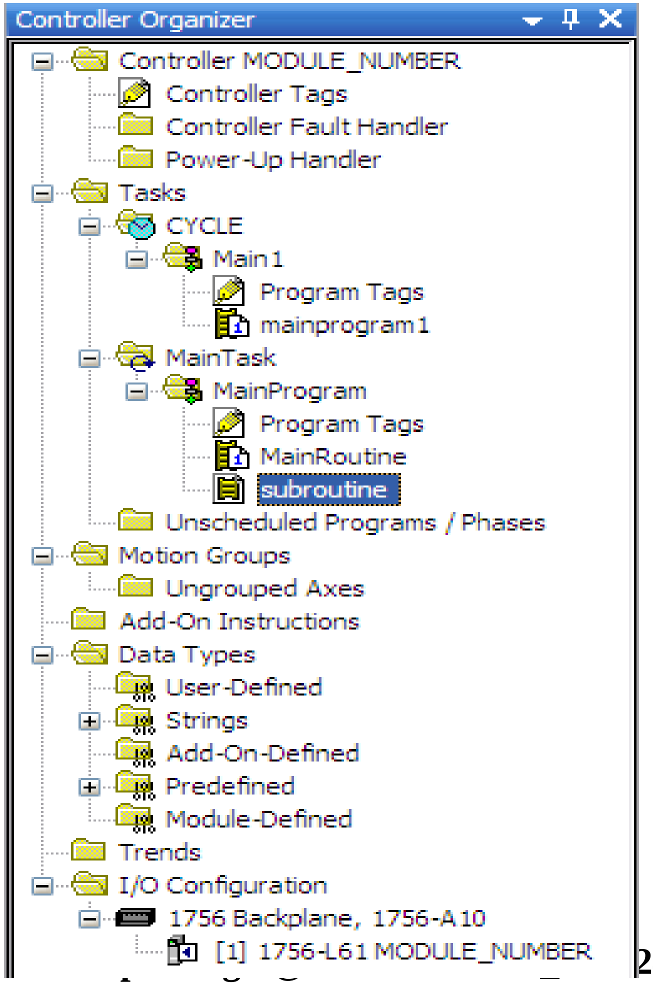
Program (local) Tags

Rountine #

Controller (global) Tags

I/O Data

System-share

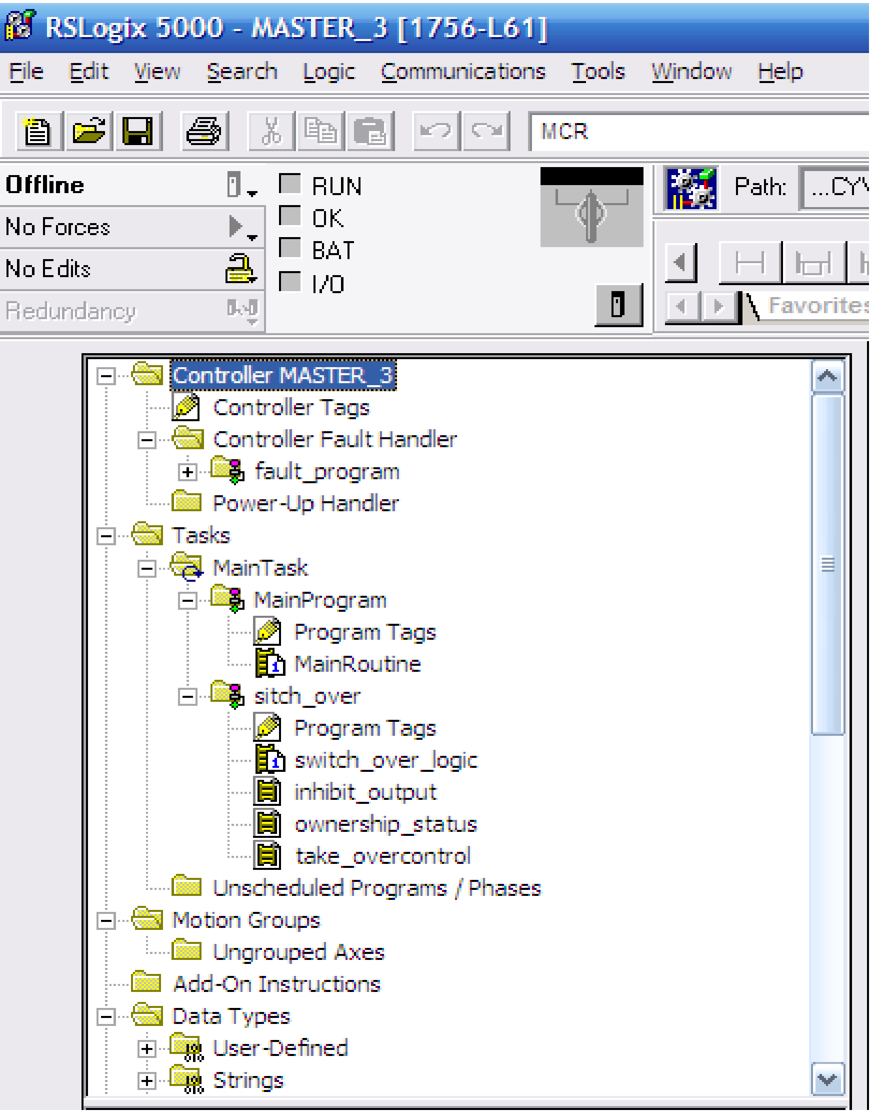


Controller Organizer

```
⊟ Controller MODULE_NUMBER
    Controller Tags
    Controller Fault Handler
    Power-Up Handler
⊟ Tasks
  ⊟ CYCLE
    ⊟ Main1
        Program Tags
        mainprogram1
  ⊟ MainTask
    ⊟ MainProgram
        Program Tags
        MainRoutine
        subroutine
    Unscheduled Programs / Phases
⊟ Motion Groups
    Ungrouped Axes
  Add-On Instructions
⊟ Data Types
    User-Defined
  ⊞ Strings
    Add-On-Defined
  ⊞ Predefined
    Module-Defined
  Trends
⊟ I/O Configuration
  ⊟ 1756 Backplane, 1756-A10
      [1] 1756-L61 MODULE_NUMBER
```

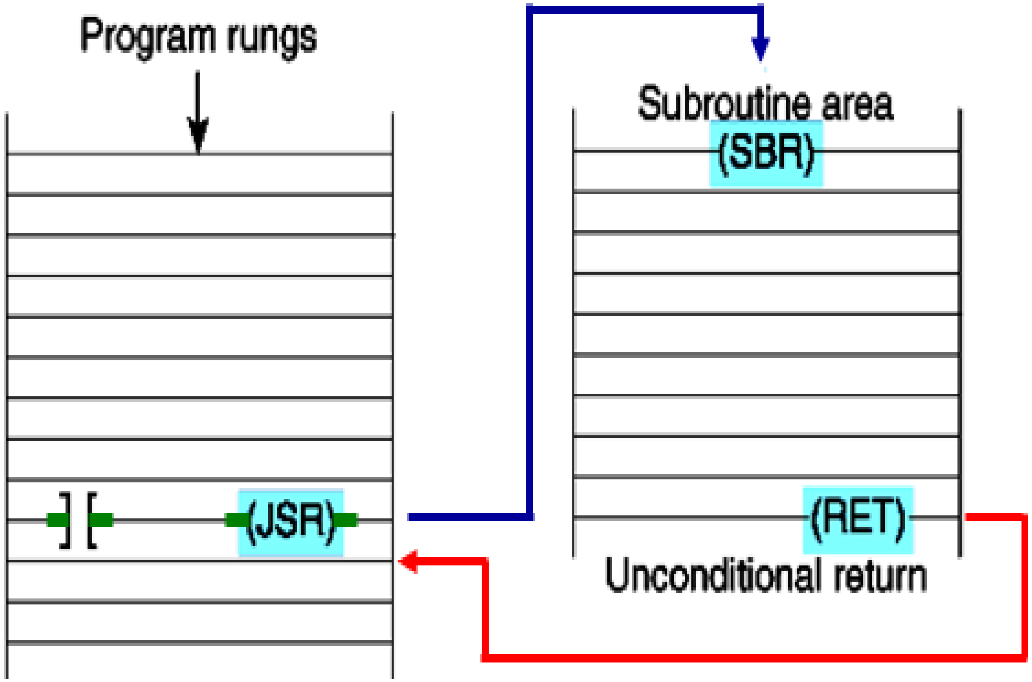
SUBROUTINE

A Subroutine is called by another routine



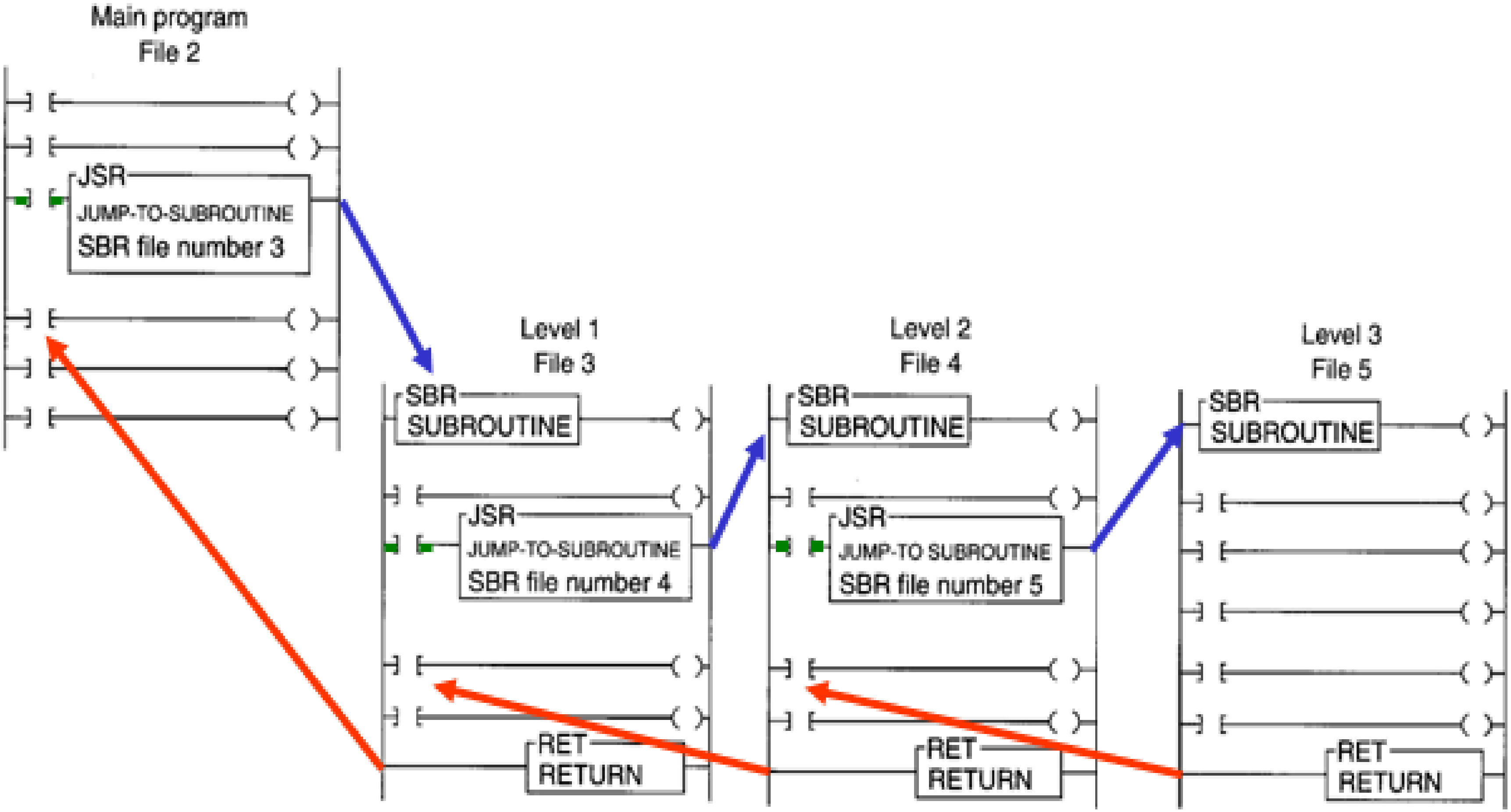
RSLogix 5000 - MASTER_3 [1756-L61]
File Edit View Search Logic Communications Tools Window Help
MCR
Offline
No Forces
No Edits
Redundancy
RUN
OK
BAT
I/O
Path: ...CYV
Favorites

- Controller MASTER_3
  - Controller Tags
  - Controller Fault Handler
    - fault_program
  - Power-Up Handler
- Tasks
  - MainTask
    - MainProgram
      - Program Tags
      - MainRoutine
    - sitch_over
      - Program Tags
      - switch_over_logic
      - inhibit_output
      - ownership_status
      - take_overcontrol
  - Unscheduled Programs / Phases
- Motion Groups
  - Ungrouped Axes
- Add-On Instructions
- Data Types
  - User-Defined
  - Strings



SUBROUTINE

Nested Subroutine

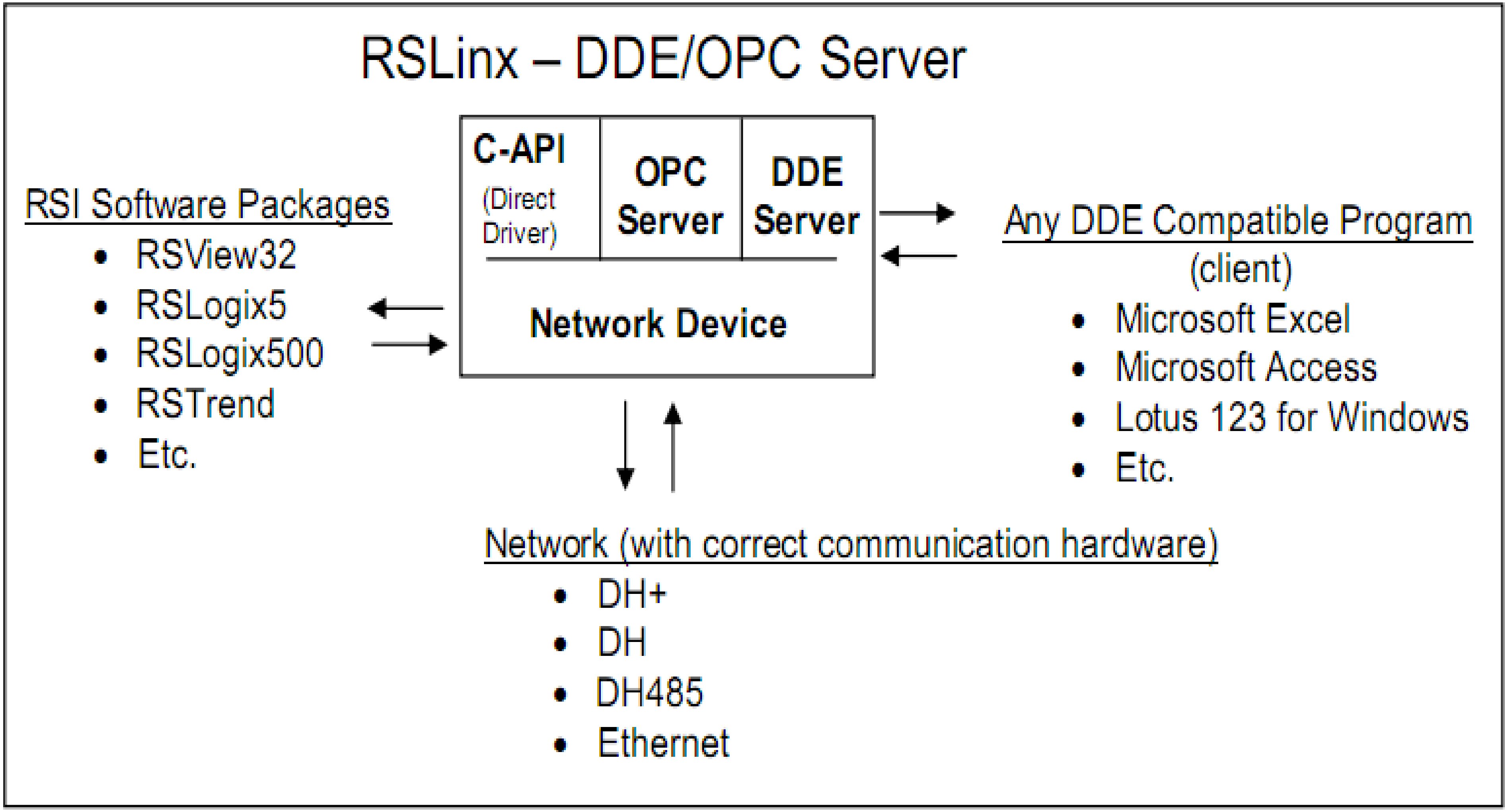


ROCKWELL SOFTWARE

- Rslogix 5000: Programming for CompactLogix and ControlLogix.
- Rslink: Communicating between RSLogix 5000 and controllers.
- Rsnetwork for Devicenet: Configuring Devicenet Network
- Rsnetwork for Controlnet: Configuring Controlnet Network
- RSview32, Factory Talk: Designing Scada Systems

ROCKWELL SOFTWARE

RSLINX



RSLinx – DDE/OPC Server

C-API (Direct Driver) | OPC Server | DDE Server

Network Device

RSI Software Packages

- RSView32
- RSLogix5
- RSLogix500
- RSTrend
- Etc.

Any DDE Compatible Program (client)

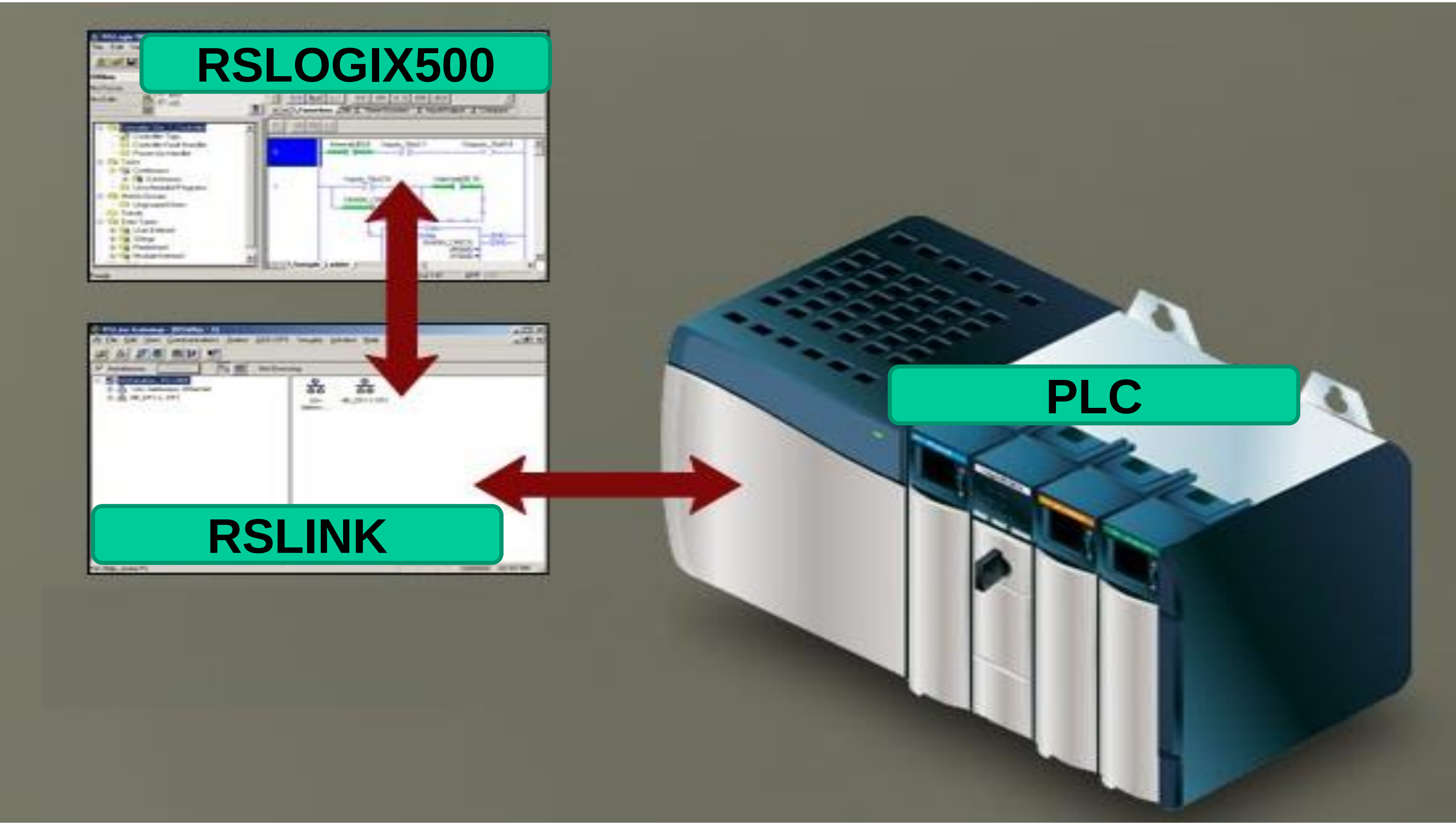
- Microsoft Excel
- Microsoft Access
- Lotus 123 for Windows
- Etc.

Network (with correct communication hardware)

- DH+
- DH
- DH485
- Ethernet

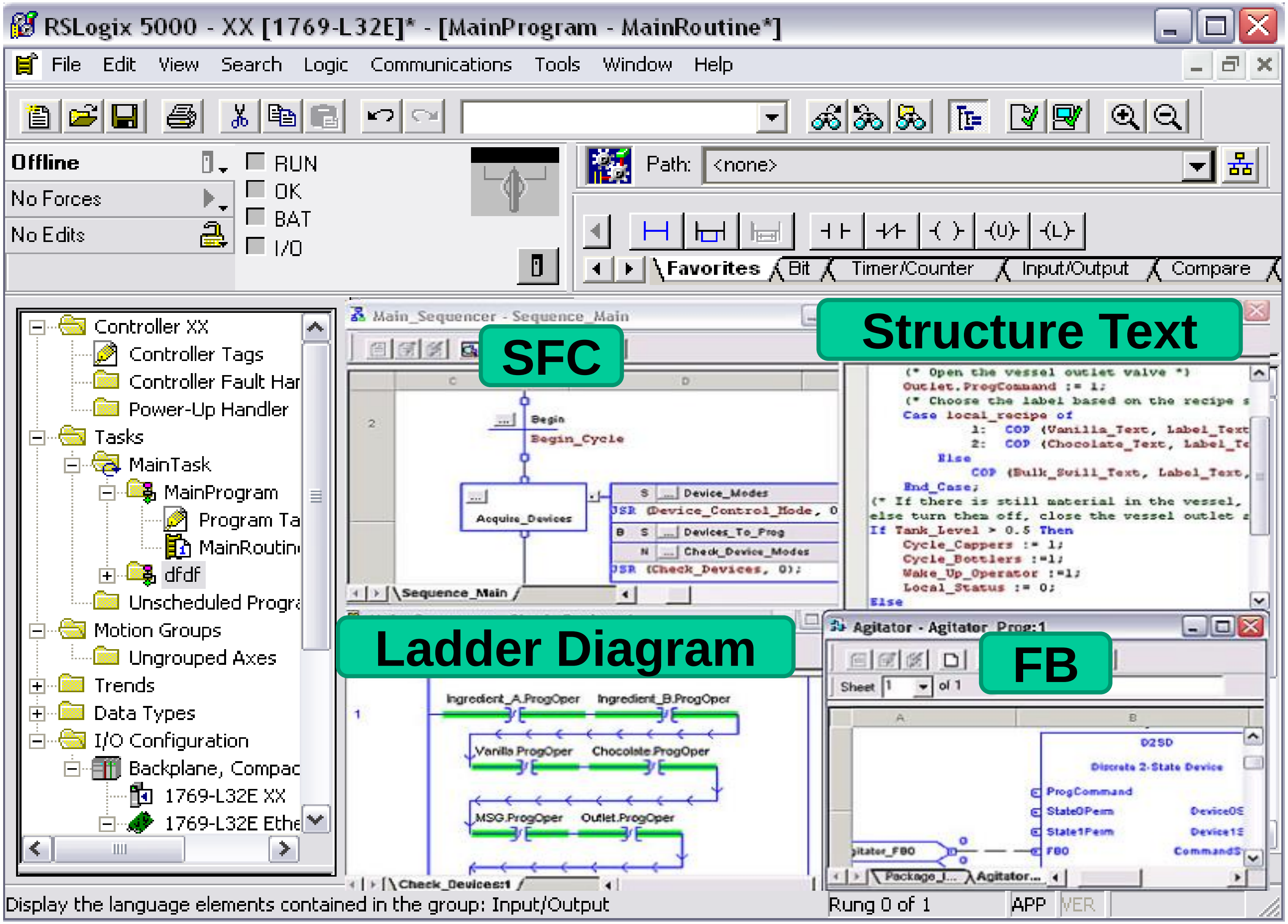
ROCKWELL SOFTWARE

RSlink



PROGRAMMING LANGUAGE

SFC, Structure Text, Ladder and FBD



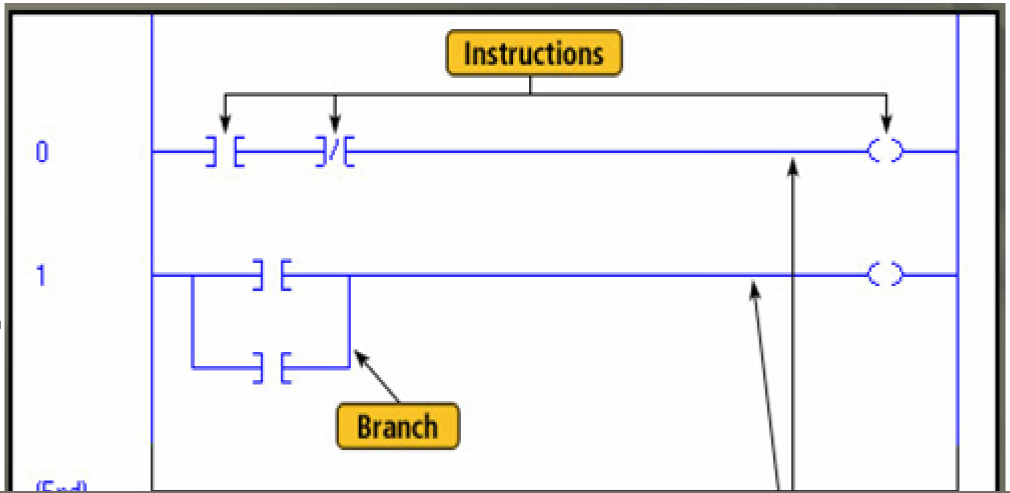
```
        (* Open the vessel outlet valve *)
        Outlet.ProgCommand := 1;
        (* Choose the label based on the recipe s
        Case local_recipe of
                1:   COP (Vanilla_Text, Label_Text
                2:   COP (Chocolate_Text, Label_Te
            Else
                COP (Bulk_Swill_Text, Label_Text,
        End_Case;
(* If there is still material in the vessel,
else turn them off, close the vessel outlet a
If Tank_Level > 0.5 Then
        Cycle_Cappers := 1;
        Cycle_Bottlers :=1;
        Wake_Up_Operator :=1;
        Local_Status := 0;
Else
```

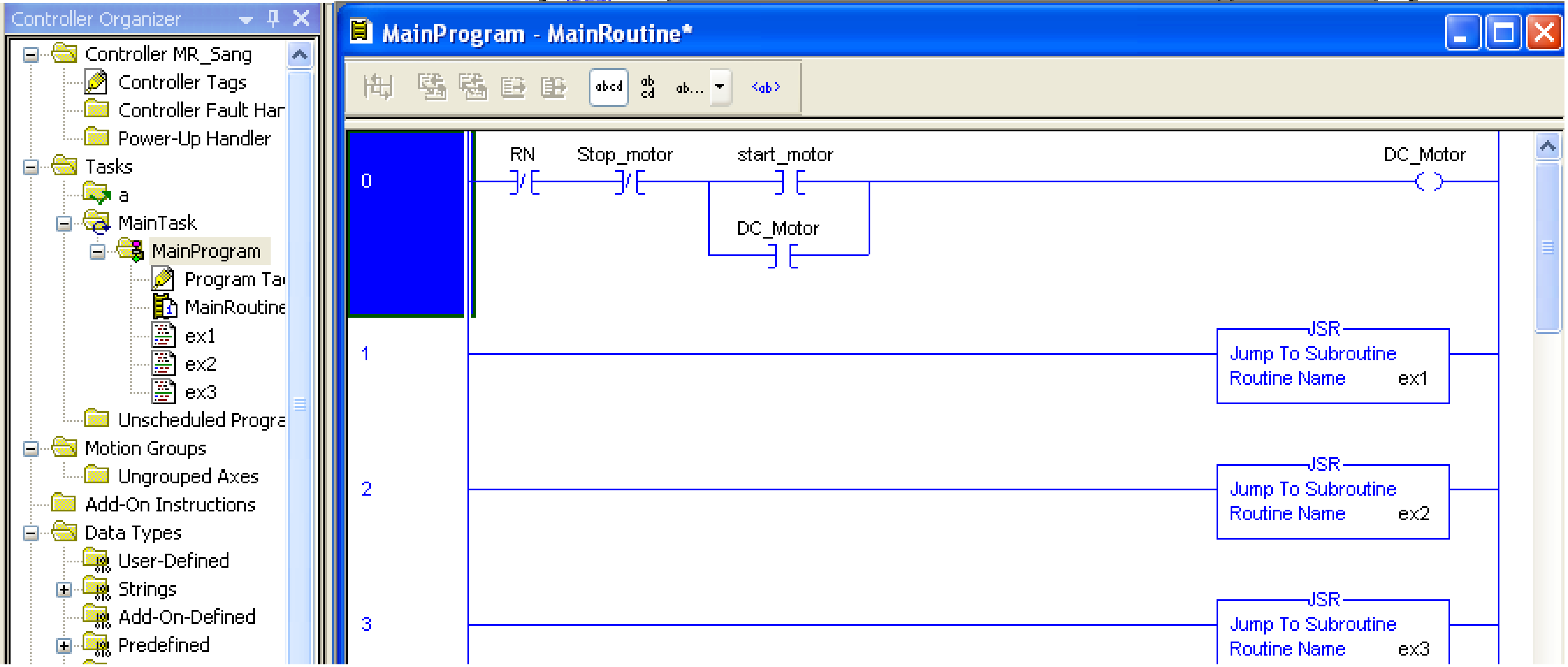
Display the language elements contained in the group: Input/Output

LAD

Ladder logic

- ➢ Rungs.
- ➢ Instructions.
- ➢ Branches.

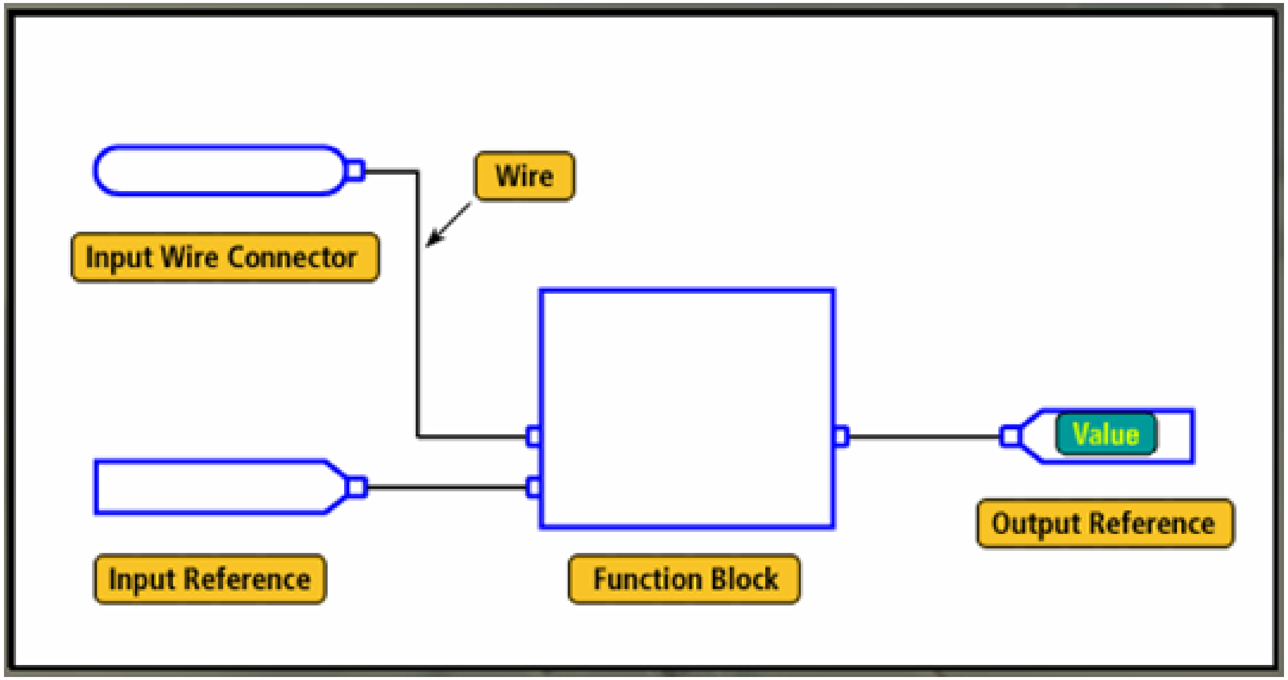




FBD

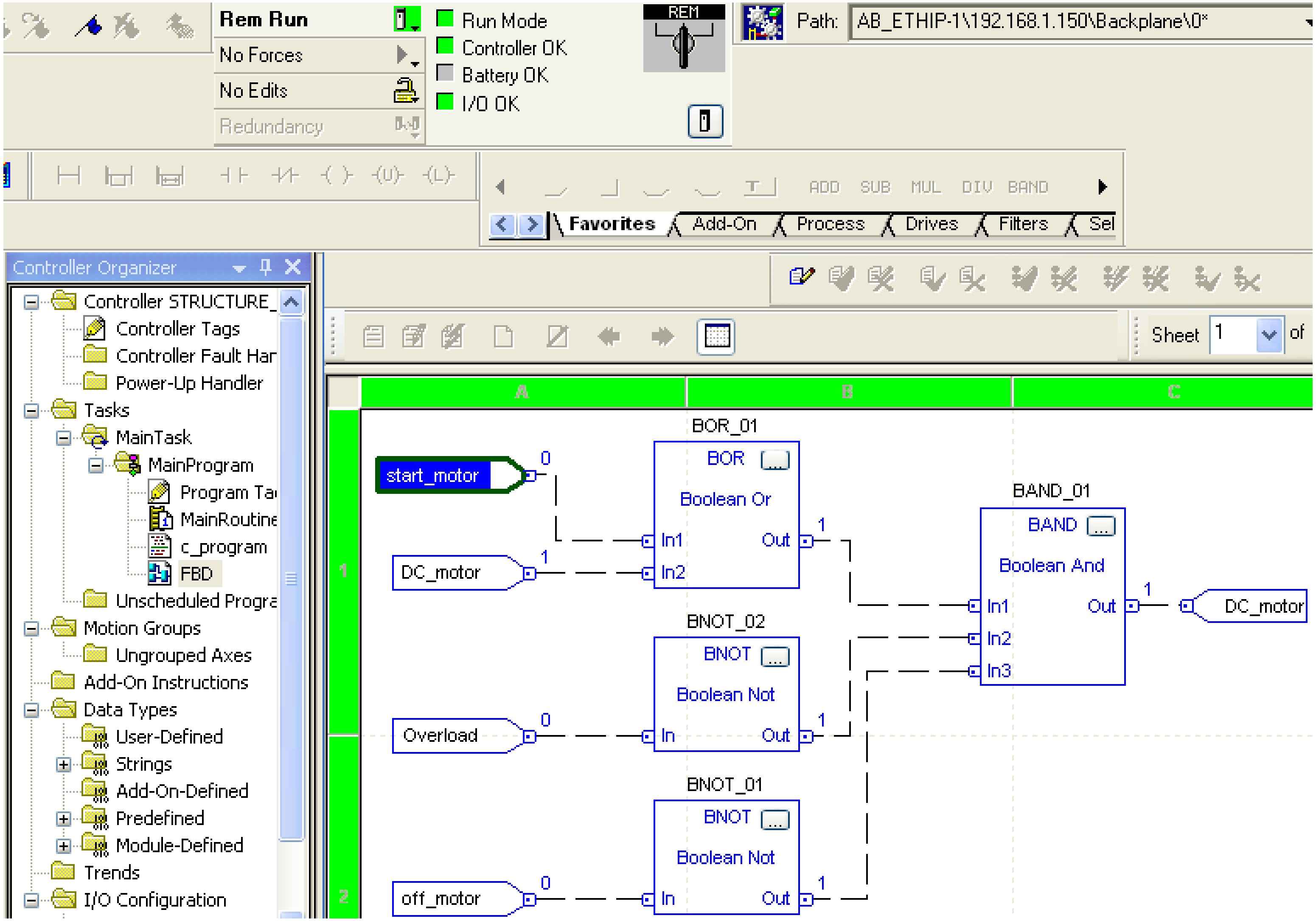
Function Block Diagram:

- Function Block.
- Input Reference.
- Output Reference.
- Wire.



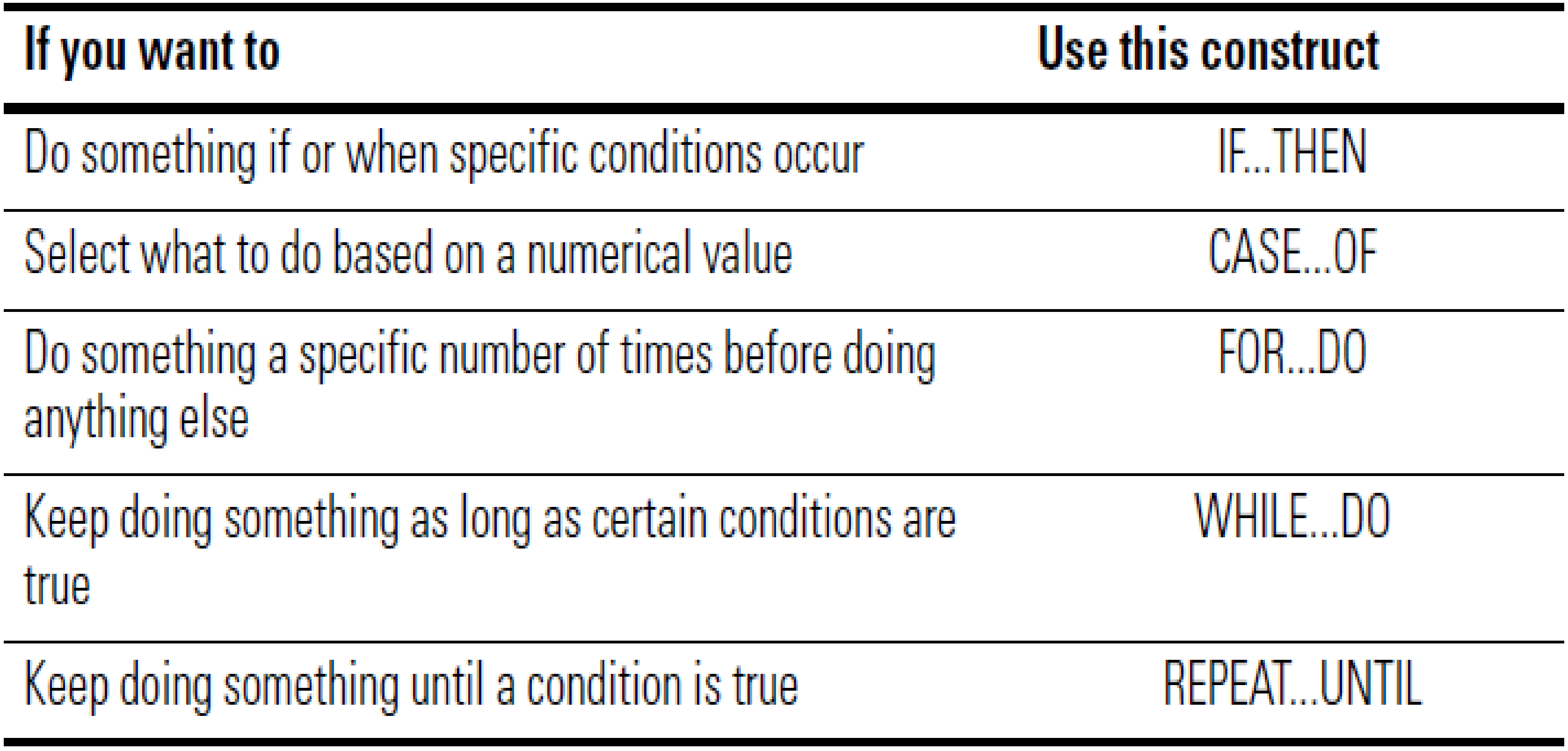
FBD

Start Stop Motor Control



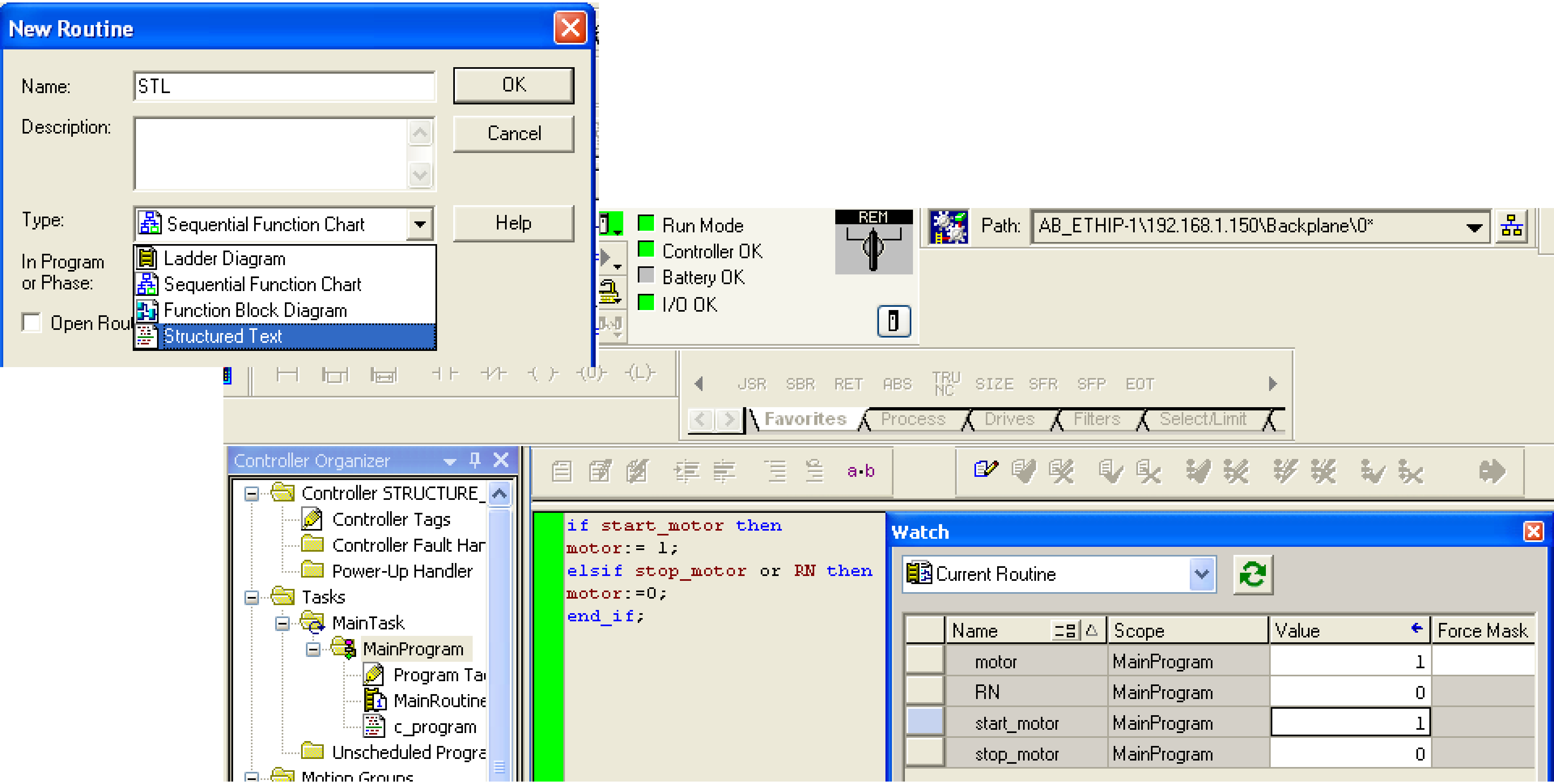
STRUCTURED TEXT

ST Construct



If you want to	Use this construct
Do something if or when specific conditions occur	IF...THEN
Select what to do based on a numerical value	CASE...OF
Do something a specific number of times before doing anything else	FOR...DO
Keep doing something as long as certain conditions are true	WHILE...DO
Keep doing something until a condition is true	REPEAT...UNTIL

STRUCTURED TEXT



New Routine

Name: STL

Description:

Type: Sequential Function Chart

- Ladder Diagram
- Sequential Function Chart
- Function Block Diagram
- Structured Text

In Program or Phase:

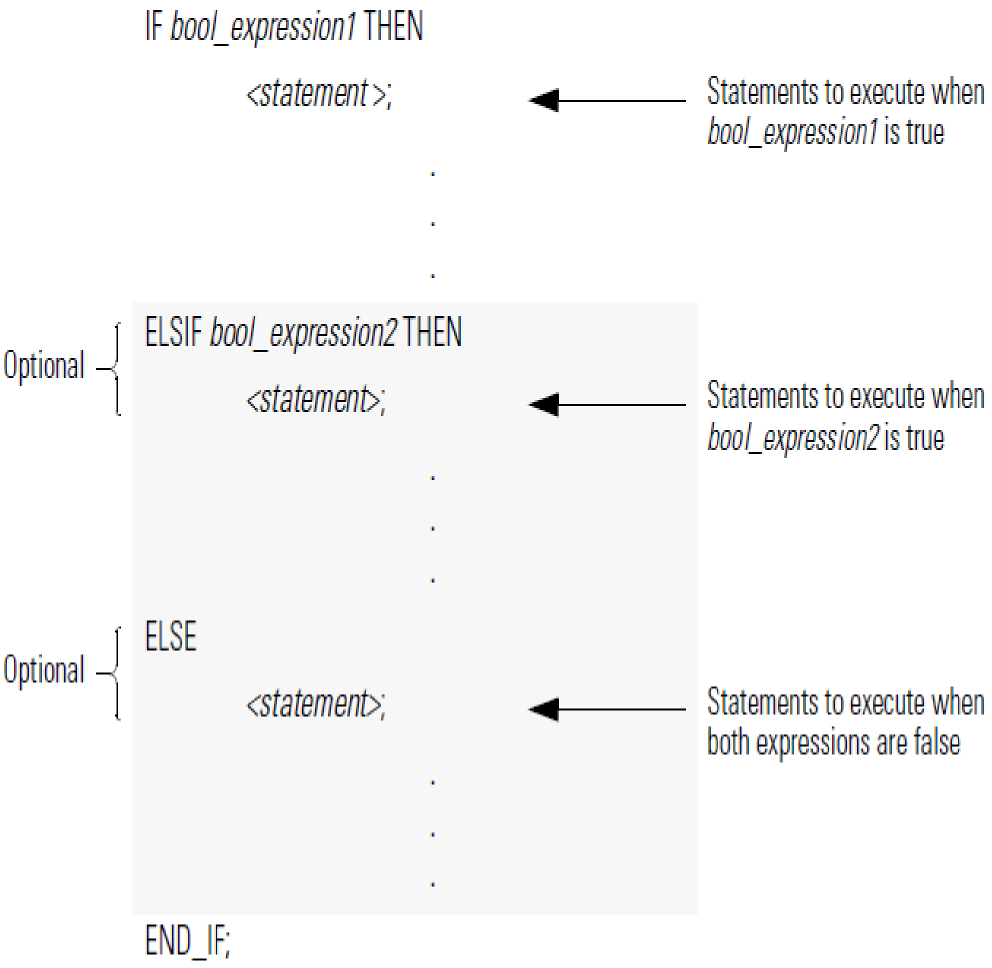
□ Open Rou

```
if start_motor then
motor:= 1;
elsif stop_motor or RN then
motor:=0;
end_if;
```

Name	Scope	Value	Force Mask
motor	MainProgram	1	
RN	MainProgram	0	
start_motor	MainProgram	1	
stop_motor	MainProgram	0	

STRUCTURED TEXT

IF.. THEN Construct

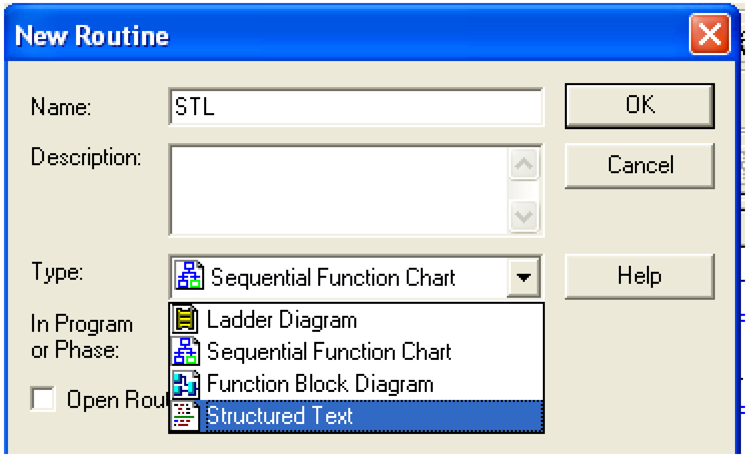


```
IF bool_expression1 THEN
    <statement>;
        .
        .
        .
ELSIF bool_expression2 THEN
    <statement>;
        .
        .
        .
ELSE
    <statement>;
        .
        .
        .
END_IF;
```

Statements to execute when *bool_expression1* is true

Optional: ELSIF … Statements to execute when *bool_expression2* is true

Optional: ELSE … Statements to execute when both expressions are false



New Routine

Name: STL

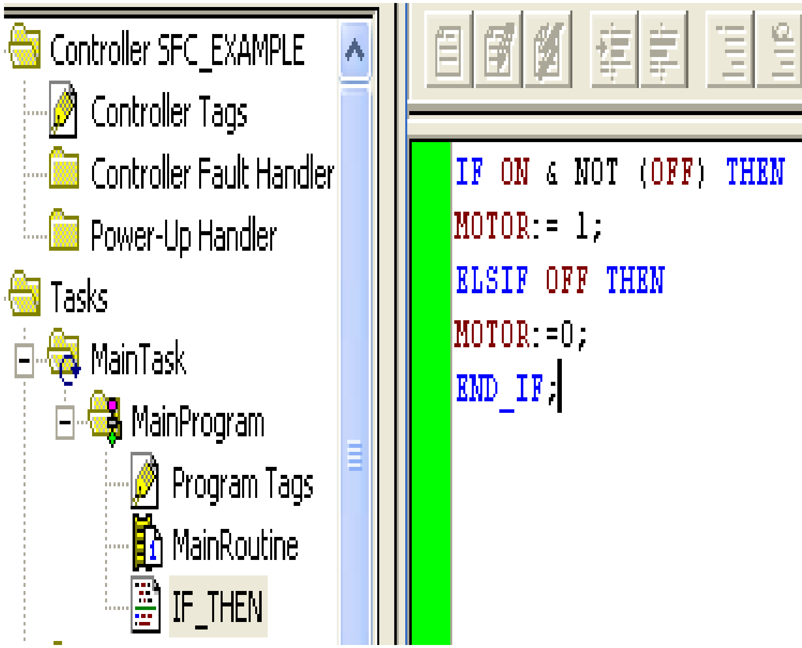
Description:

Type: Sequential Function Chart

- Ladder Diagram
- Sequential Function Chart
- Function Block Diagram
- Structured Text

In Program or Phase:

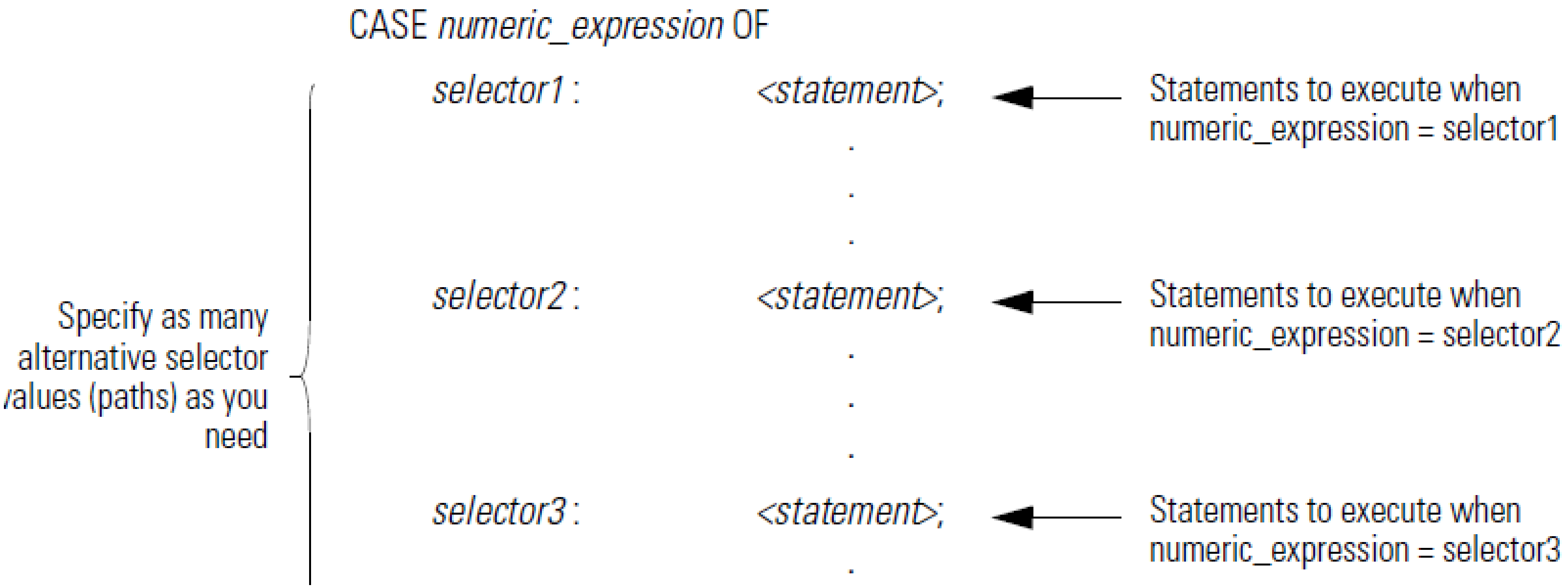
□ Open Rou



```
IF ON & NOT (OFF) THEN
MOTOR:= 1;
ELSIF OFF THEN
MOTOR:=0;
END_IF;
```

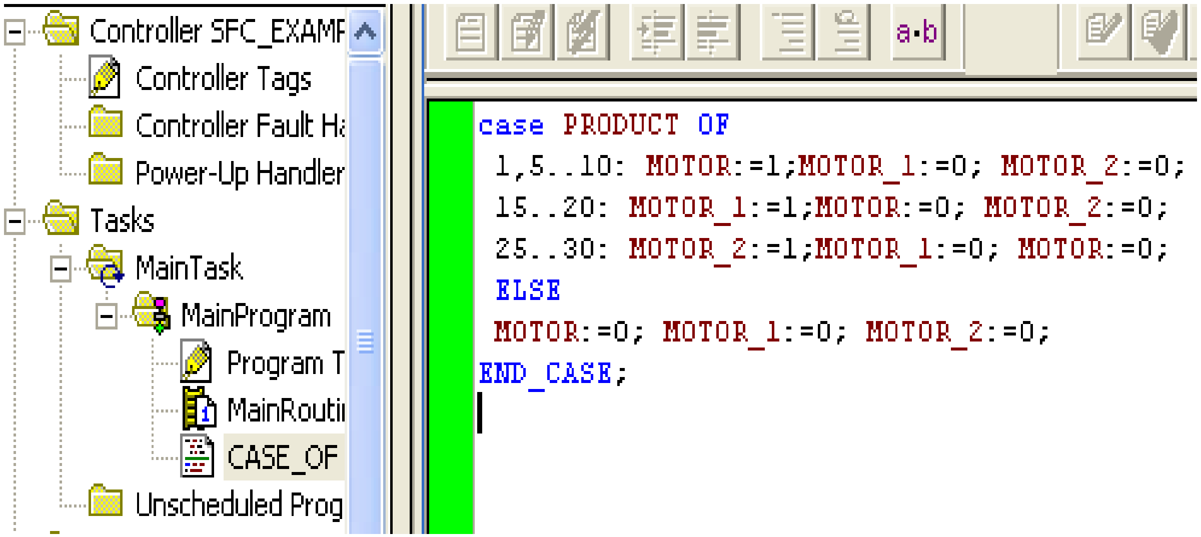
STRUCTURED TEXT

CASE...OF Construct



```
CASE numeric_expression OF
    selector1 :        <statement>;    Statements to execute when
                                       numeric_expression = selector1
    selector2 :        <statement>;    Statements to execute when
                                       numeric_expression = selector2
    selector3 :        <statement>;    Statements to execute when
                                       numeric_expression = selector3
```

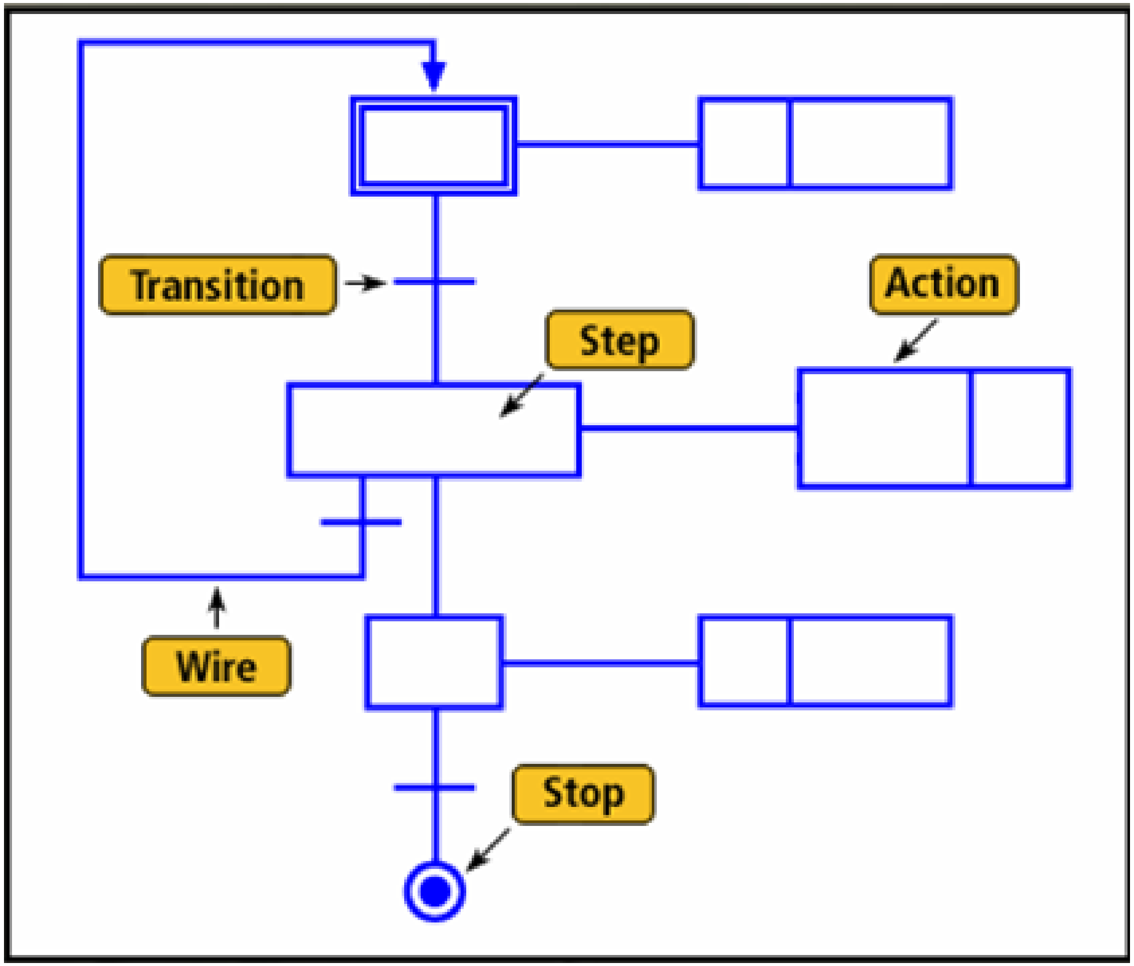
Specify as many alternative selector values (paths) as you need



```
case PRODUCT OF
 1,5..10: MOTOR:=1;MOTOR_1:=0; MOTOR_2:=0;
 15..20: MOTOR_1:=1;MOTOR:=0; MOTOR_2:=0;
 25..30: MOTOR_2:=1;MOTOR_1:=0; MOTOR:=0;
 ELSE
 MOTOR:=0; MOTOR_1:=0; MOTOR_2:=0;
END_CASE;
```

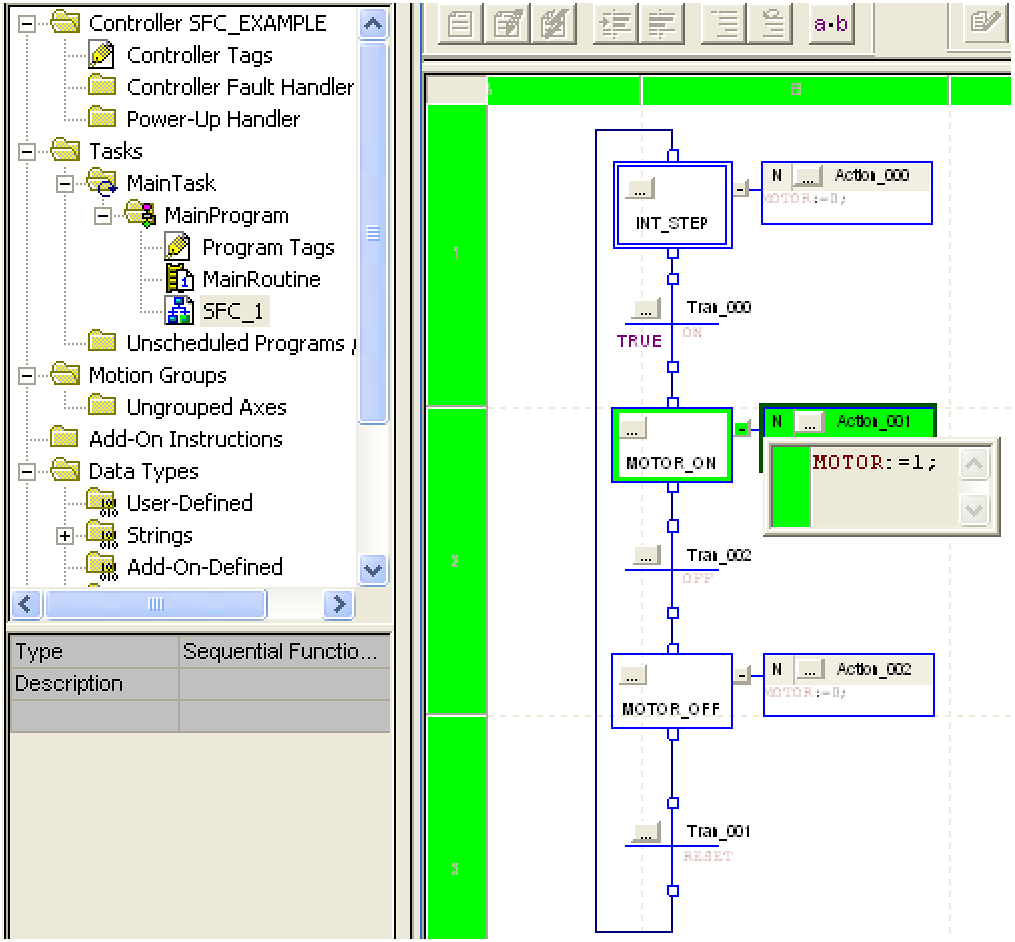
SFC

Sequential Function Chart (SFC):



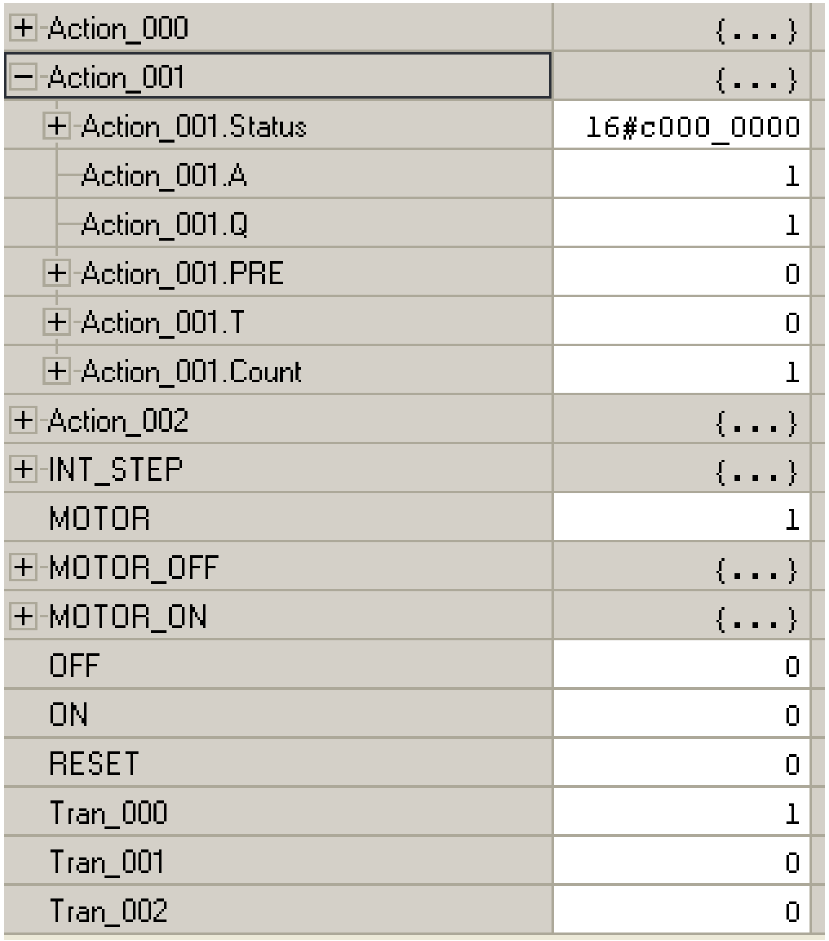
SFC

Start Stop Motor Control



Type	Sequential Functio...
Description	

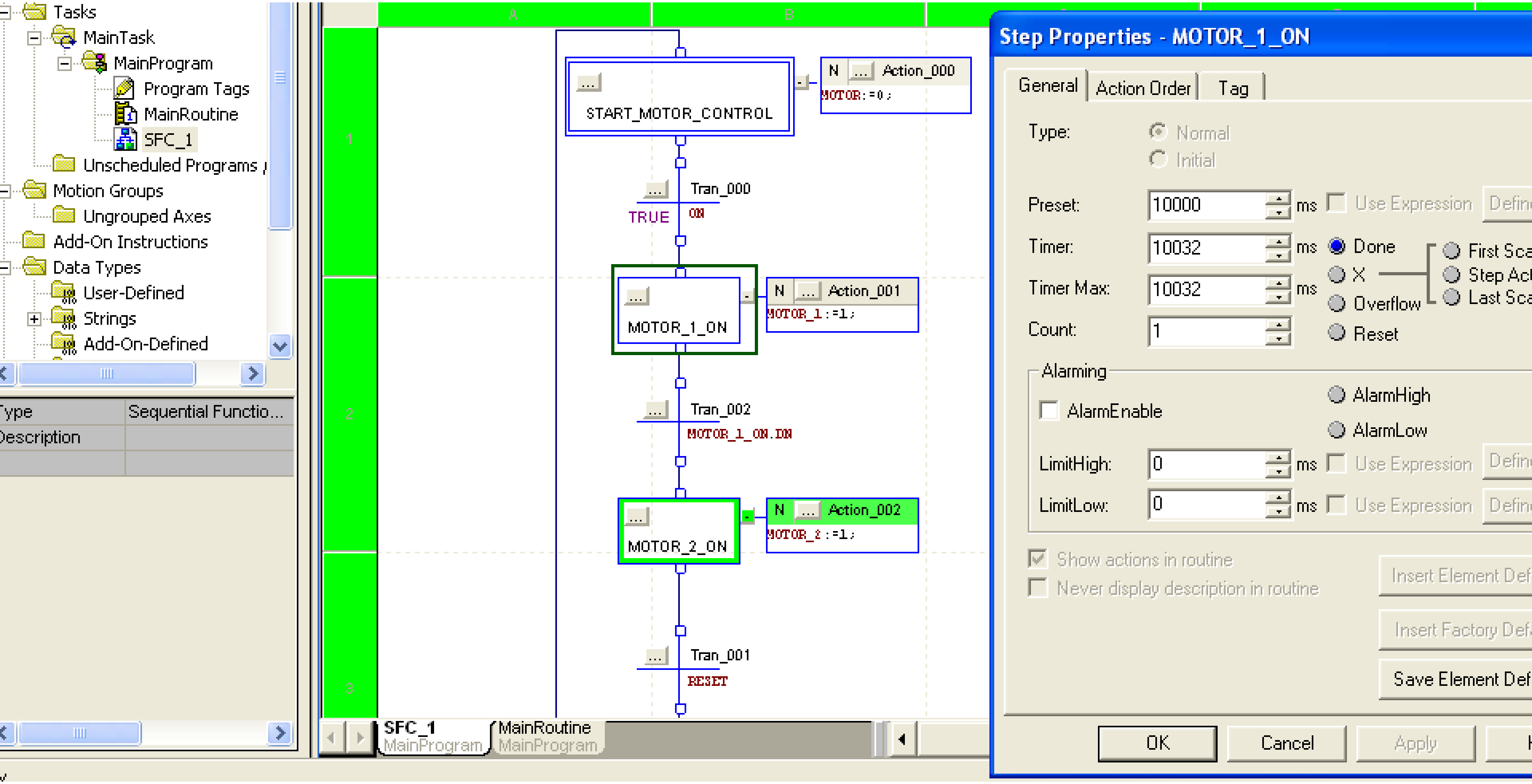
```
Action_000: MOTOR:=0;
Tran_000: ON
Action_001: MOTOR:=1;
Tran_002: OFF
Action_002: MOTOR:=0;
Tran_001: RESET
```



⊞ Action_000	{...}
⊟ Action_001	{...}
⊞ Action_001.Status	16#c000_0000
Action_001.A	1
Action_001.Q	1
⊞ Action_001.PRE	0
⊞ Action_001.T	0
⊞ Action_001.Count	1
⊞ Action_002	{...}
⊞ INT_STEP	{...}
MOTOR	1
⊞ MOTOR_OFF	{...}
⊞ MOTOR_ON	{...}
OFF	0
ON	0
RESET	0
Tran_000	1
Tran_001	0
Tran_002	0

SFC

Sequential Motor Starter



Type | Sequential Functio...
Description |

```
MOTOR:=0;
ON
MOTOR_1:=1;
MOTOR_1_ON.DN
MOTOR_2:=1;
RESET
```

Type: ● Normal ○ Initial

Preset: 10000 ms □ Use Expression Defin

Timer: 10032 ms ● Done

Timer Max: 10032 ms ○ X

Count: 1 ○ Overflow ○ Reset

Alarming

□ AlarmEnable ○ AlarmHigh ○ AlarmLow

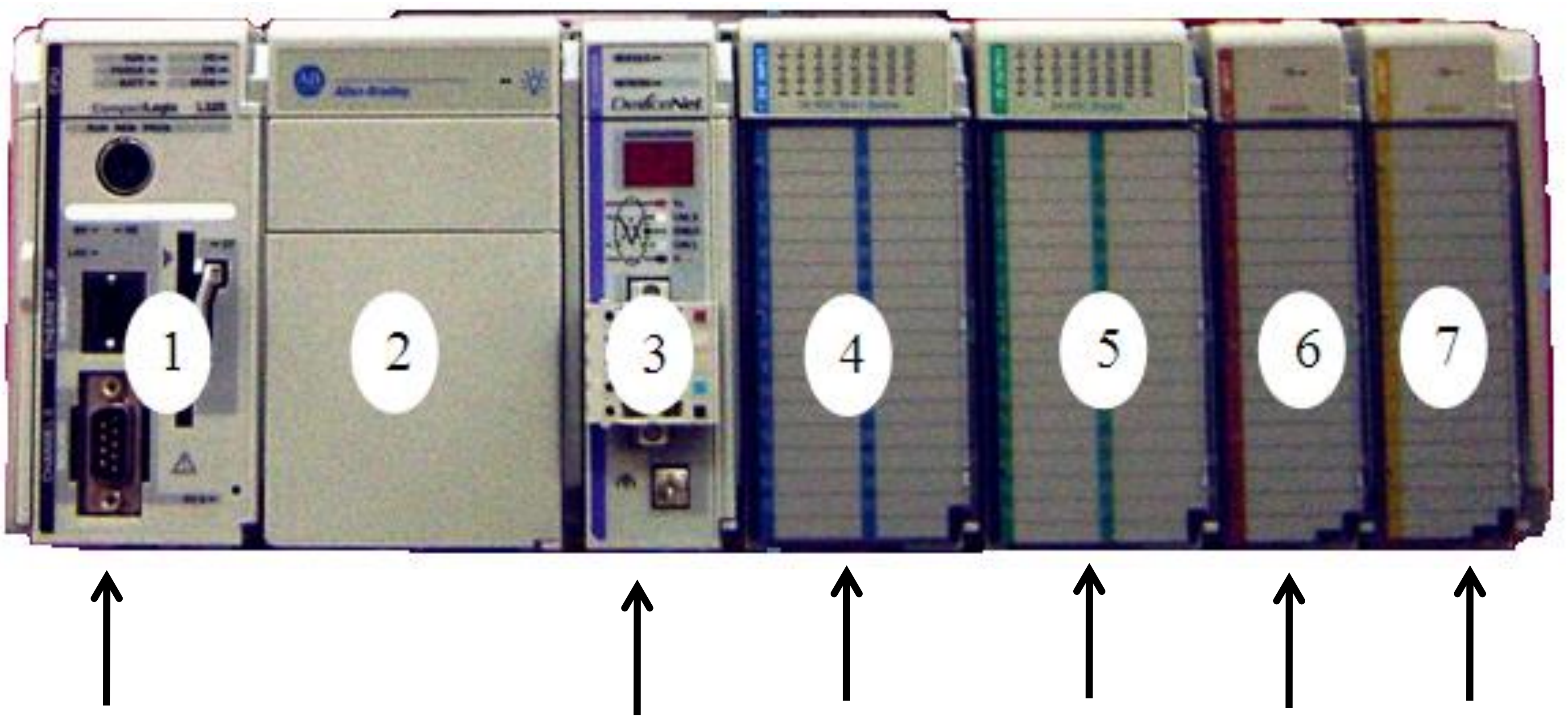
LimitHigh: 0 ms □ Use Expression Defin

LimitLow: 0 ms □ Use Expression Defin

☑ Show actions in routine

□ Never display description in routine

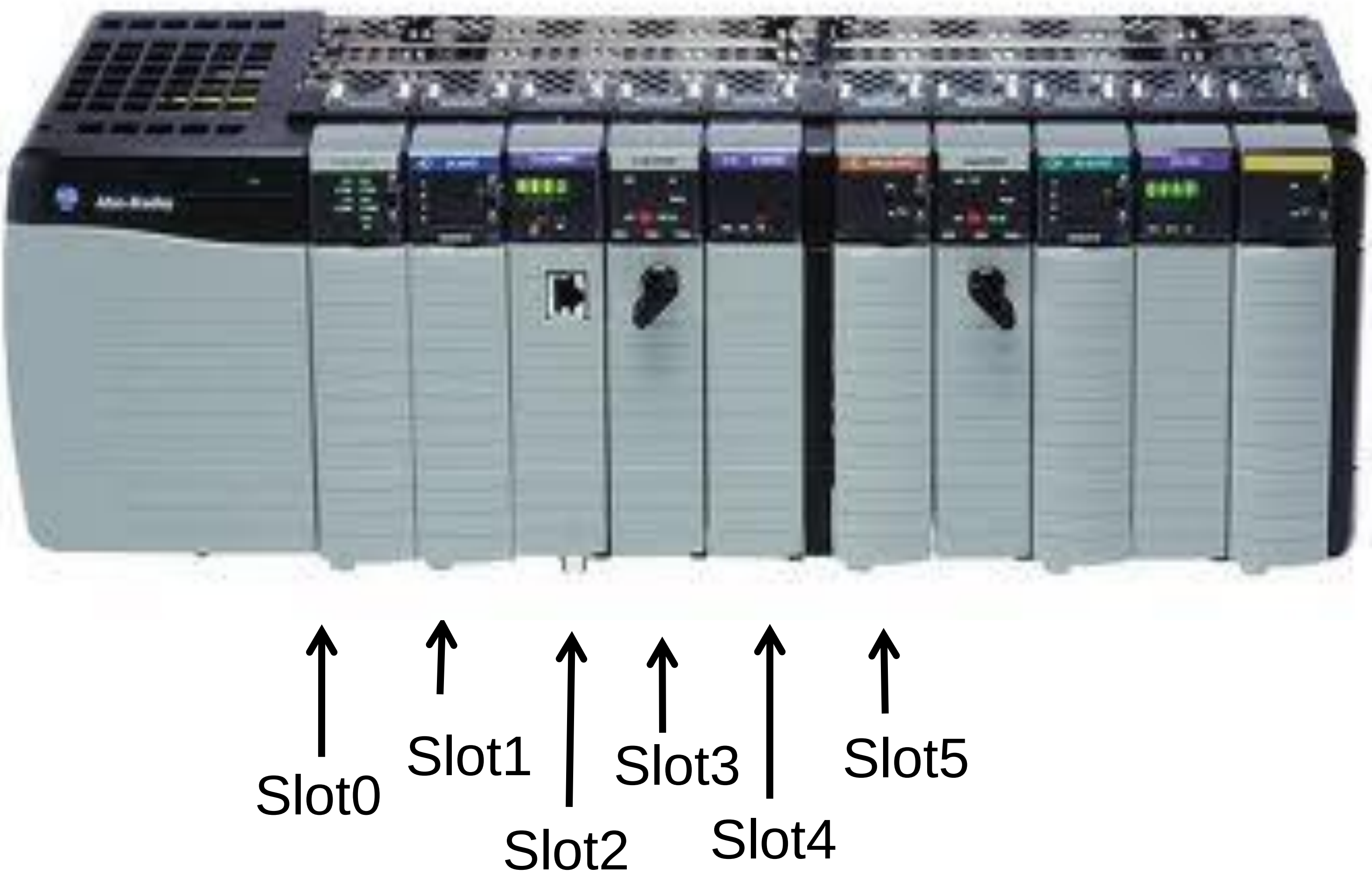
COMPACTLOGIX TRAINING KIT



Slot0 Slot1 Slot2 Slot3 Slot4 Slot5

Except the CPU, all modules can be changed their position

CONTROLLOGIX TRAINING KIT



CPUs and modules can be placed in any slot of chassis

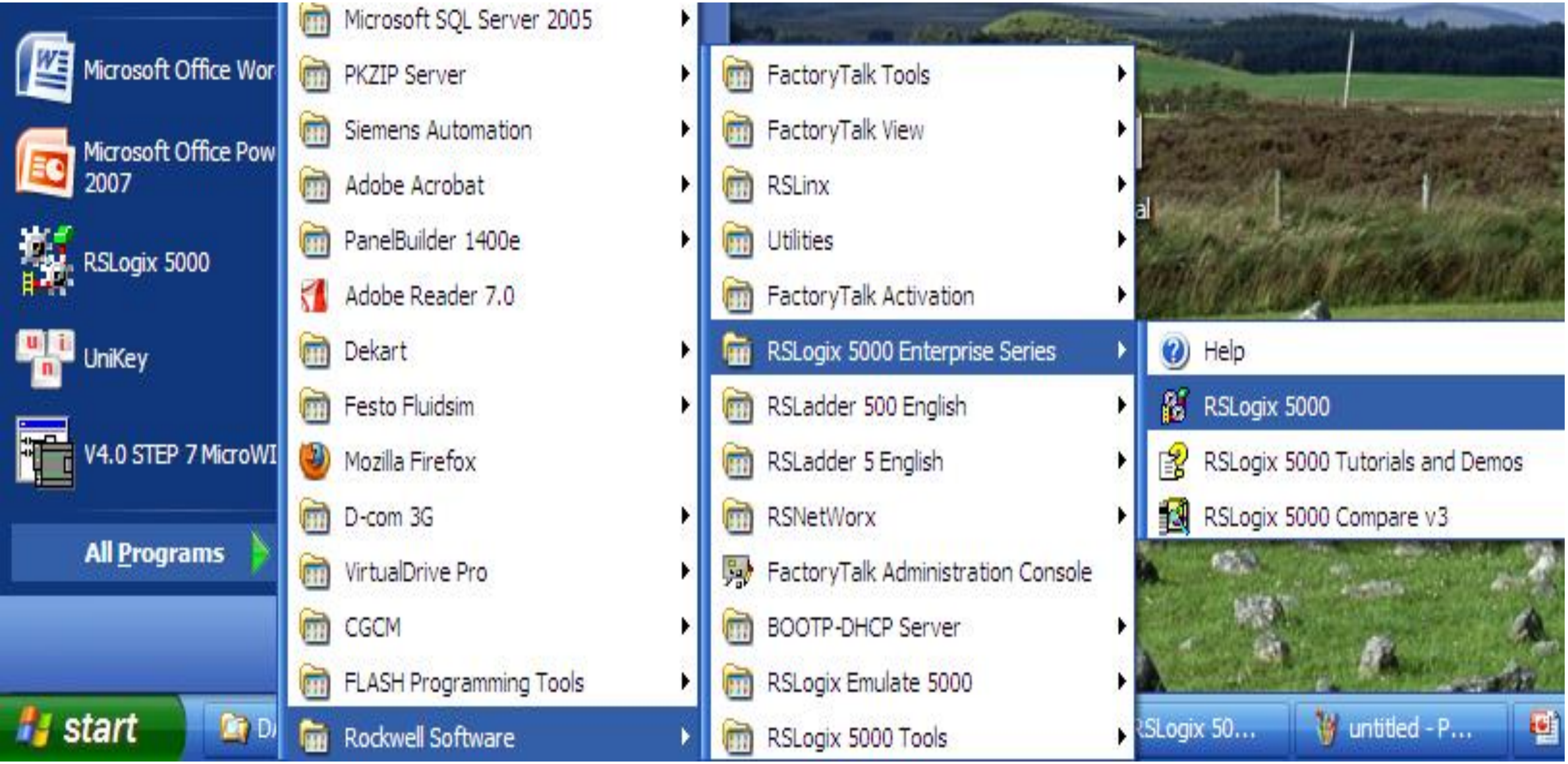
PLC PROGRAMMING

Working with a project

1. Connecting hardware
2. Configuring CPU and I/O module by Rslogix 5000
3. Create Tags(Program Tags or Controller Tags)
4. *Alias Tags to represent another tag*
5. Write logic: LAD, FBD, ST, SFC
6. Download to CPU by Rslinx via Rs232 or Ethernet
7. Run and check

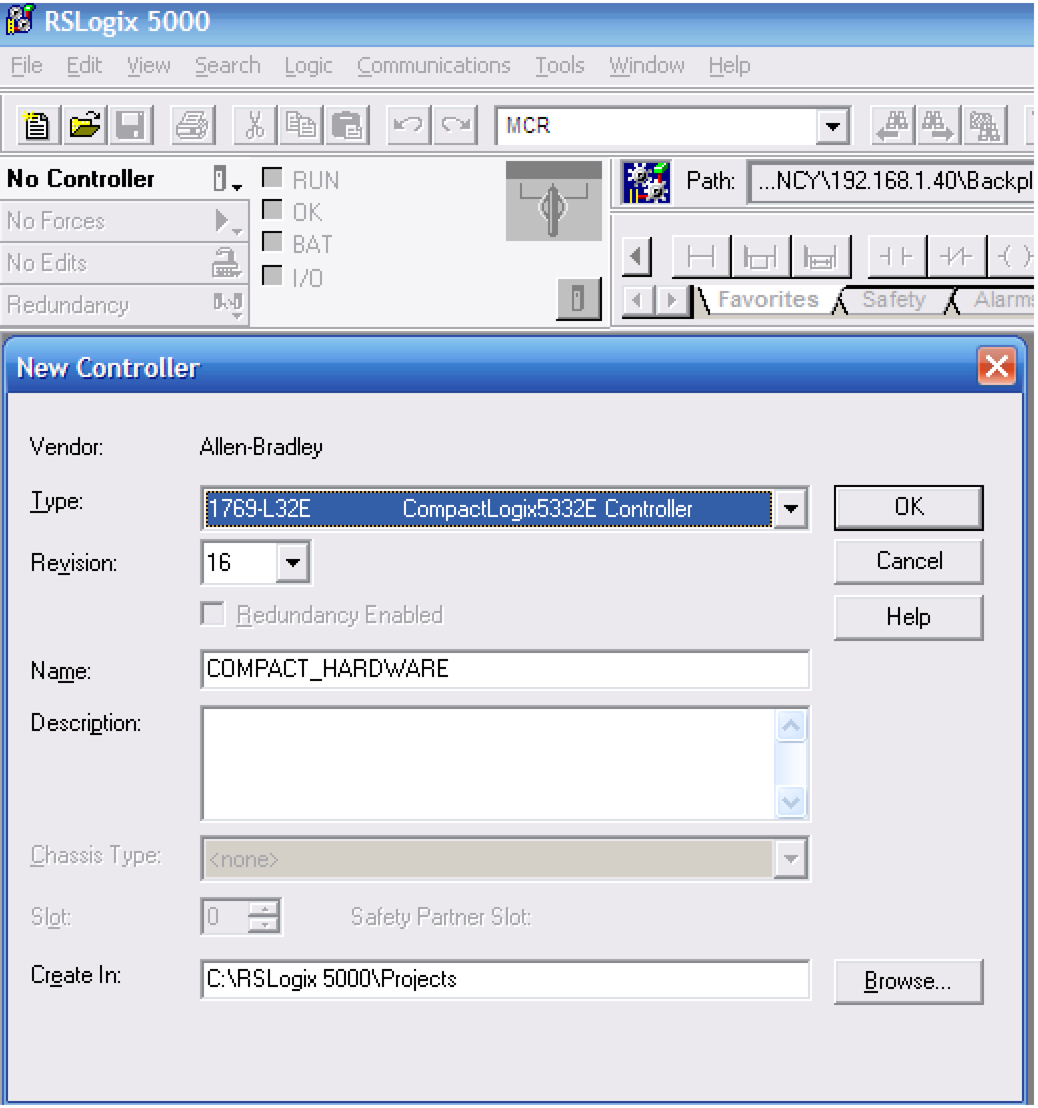
PLC PROGRAMMING

Open Rslogix 5000



PLC PROGRAMMING

Configure hardware for commpactLogix


RSLogix 5000
File Edit View Search Logic Communications Tools Window Help
MCR
No Controller
No Forces
No Edits
Redundancy
RUN
OK
BAT
I/O
Path: ...NCY\192.168.1.40\Backpl
Favorites
Safety
Alarms

New Controller

Vendor: Allen-Bradley

Type: 1769-L32E CompactLogix5332E Controller

Revision: 16

□ Redundancy Enabled

Name: COMPACT_HARDWARE

Description:

Chassis Type: <none>

Slot: 0 Safety Partner Slot:

Create In: C:\RSLogix 5000\Projects

Open Rslogix 500, Create a new project, slelect a appropriate CPU and Revision, enter project name and save.

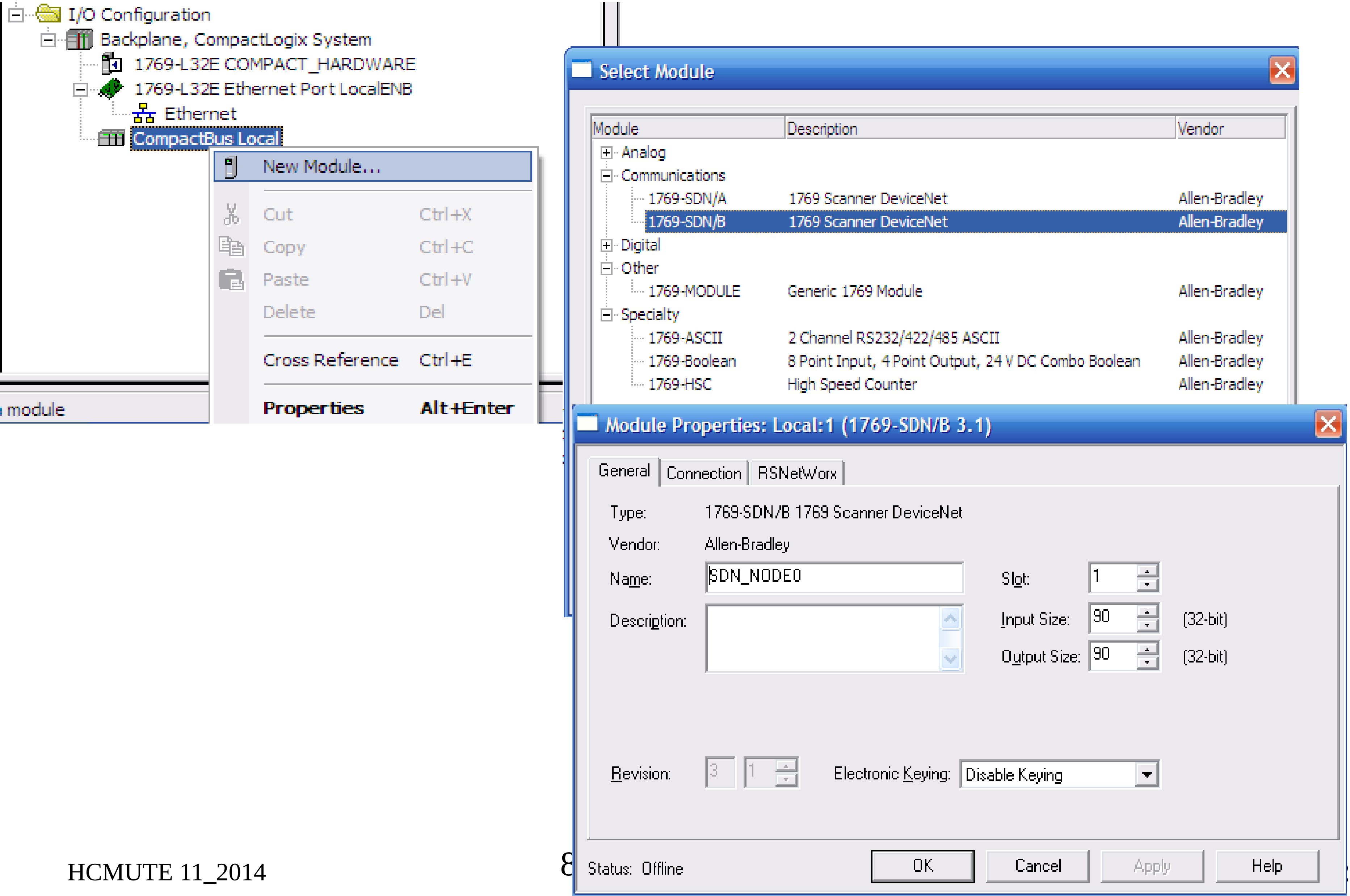
Notice:

CPU type must be matched with real CPU.

For controllogix, CPU can be placed in any slot of chassis

PLC PROGRAMMING

Configure hardware for commpactLogix: Adding Dnet module



Module	Description	Vendor
Analog		
Communications		
1769-SDN/A	1769 Scanner DeviceNet	Allen-Bradley
1769-SDN/B	1769 Scanner DeviceNet	Allen-Bradley
Digital		
Other		
1769-MODULE	Generic 1769 Module	Allen-Bradley
Specialty		
1769-ASCII	2 Channel RS232/422/485 ASCII	Allen-Bradley
1769-Boolean	8 Point Input, 4 Point Output, 24 V DC Combo Boolean	Allen-Bradley
1769-HSC	High Speed Counter	Allen-Bradley

Type: 1769-SDN/B 1769 Scanner DeviceNet

Vendor: Allen-Bradley

Name: SDN_NODE0

Slot: 1

Description:

Input Size: 90 (32-bit)

Output Size: 90 (32-bit)

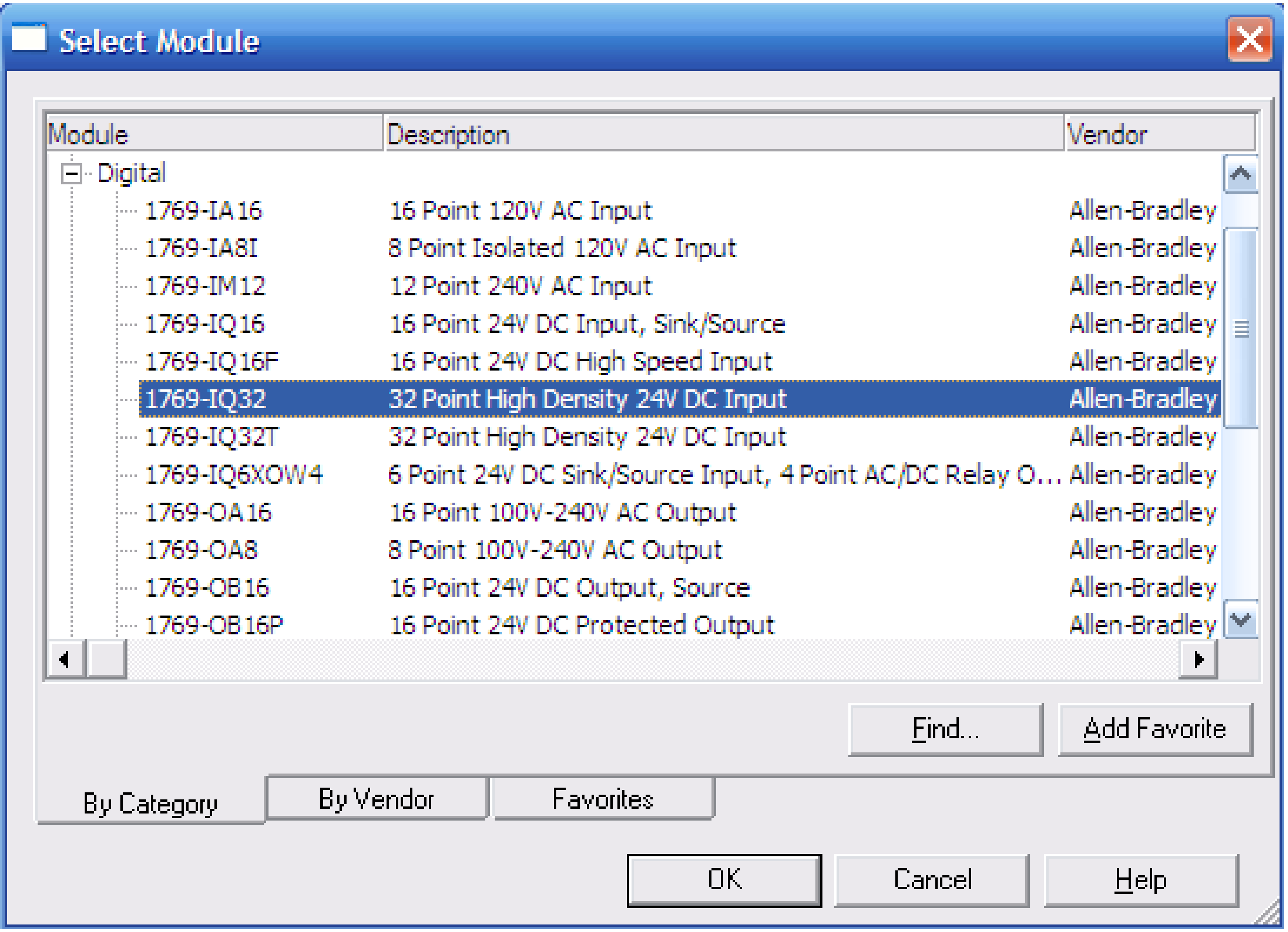
Revision: 3 1

Electronic Keying: Disable Keying

Status: Offline

PLC PROGRAMMING

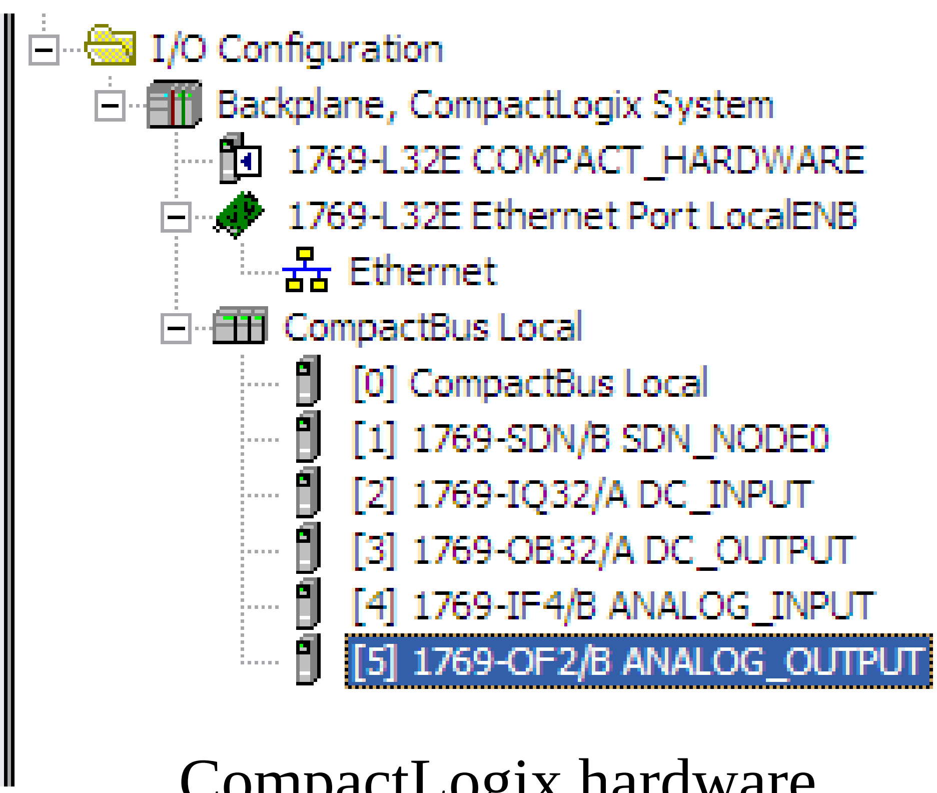
Configure hardware for commpactLogix: Adding Input module



Module	Description	Vendor
Digital		
1769-IA16	16 Point 120V AC Input	Allen-Bradley
1769-IA8I	8 Point Isolated 120V AC Input	Allen-Bradley
1769-IM12	12 Point 240V AC Input	Allen-Bradley
1769-IQ16	16 Point 24V DC Input, Sink/Source	Allen-Bradley
1769-IQ16F	16 Point 24V DC High Speed Input	Allen-Bradley
1769-IQ32	32 Point High Density 24V DC Input	Allen-Bradley
1769-IQ32T	32 Point High Density 24V DC Input	Allen-Bradley
1769-IQ6XOW4	6 Point 24V DC Sink/Source Input, 4 Point AC/DC Relay O...	Allen-Bradley
1769-OA16	16 Point 100V-240V AC Output	Allen-Bradley
1769-OA8	8 Point 100V-240V AC Output	Allen-Bradley
1769-OB16	16 Point 24V DC Output, Source	Allen-Bradley
1769-OB16P	16 Point 24V DC Protected Output	Allen-Bradley

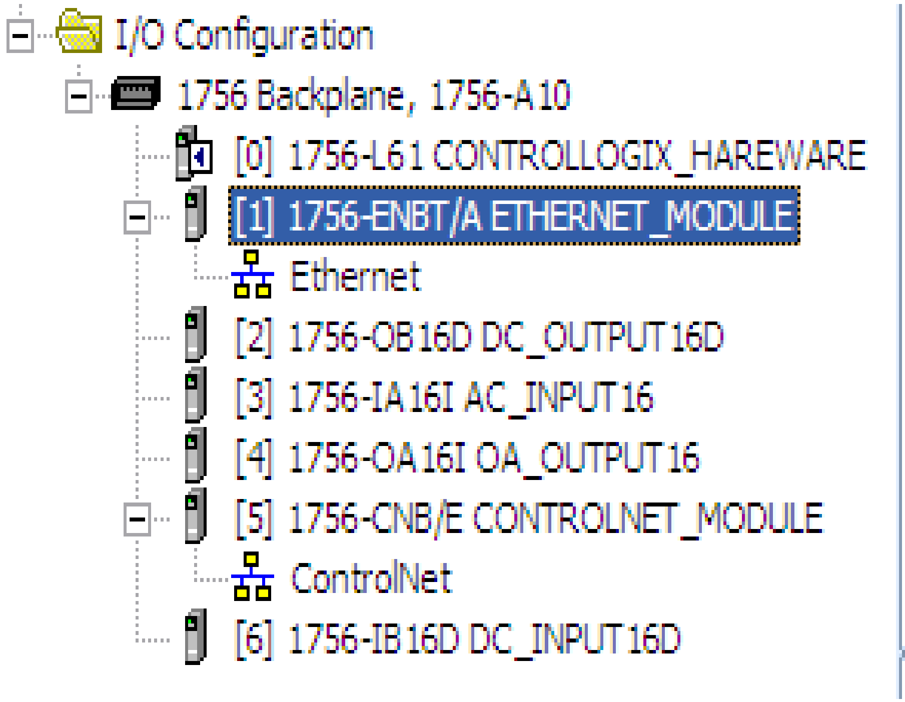
PROGRAMMING

Configure hardware for commpactLogix: Similar to others modules



```
I/O Configuration
  Backplane, CompactLogix System
    1769-L32E COMPACT_HARDWARE
    1769-L32E Ethernet Port LocalENB
      Ethernet
    CompactBus Local
      [0] CompactBus Local
      [1] 1769-SDN/B SDN_NODE0
      [2] 1769-IQ32/A DC_INPUT
      [3] 1769-OB32/A DC_OUTPUT
      [4] 1769-IF4/B ANALOG_INPUT
      [5] 1769-OF2/B ANALOG_OUTPUT
```

CompactLogix hardware



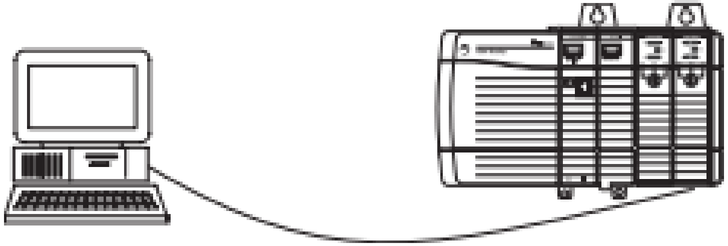
```
I/O Configuration
  1756 Backplane, 1756-A10
    [0] 1756-L61 CONTROLLOGIX_HAREWARE
    [1] 1756-ENBT/A ETHERNET_MODULE
      Ethernet
    [2] 1756-OB16D DC_OUTPUT16D
    [3] 1756-IA16I AC_INPUT16
    [4] 1756-OA16I OA_OUTPUT16
    [5] 1756-CNB/E CONTROLNET_MODULE
      ControlNet
    [6] 1756-IB16D DC_INPUT16D
```

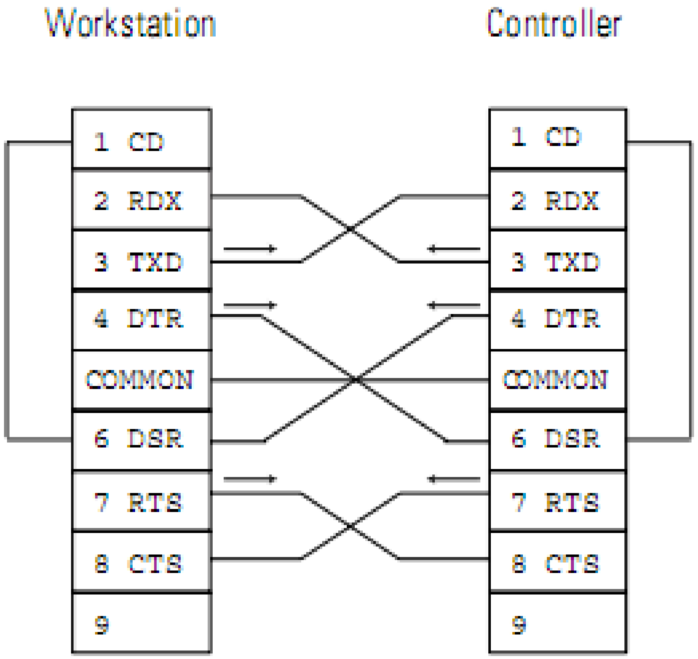
ControlLogix hardware

EX11: Participants configure hardware for compactLogix and ControlLogix Controller.

CONNECT **PC** TO **CPU**

Directly connect to the CPU via the serial port



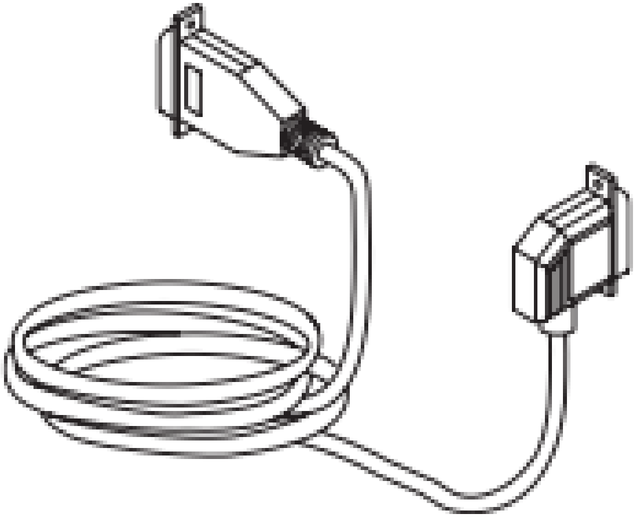


Workstation

1 CD
2 RDX
3 TXD
4 DTR
COMMON
6 DSR
7 RTS
8 CTS
9

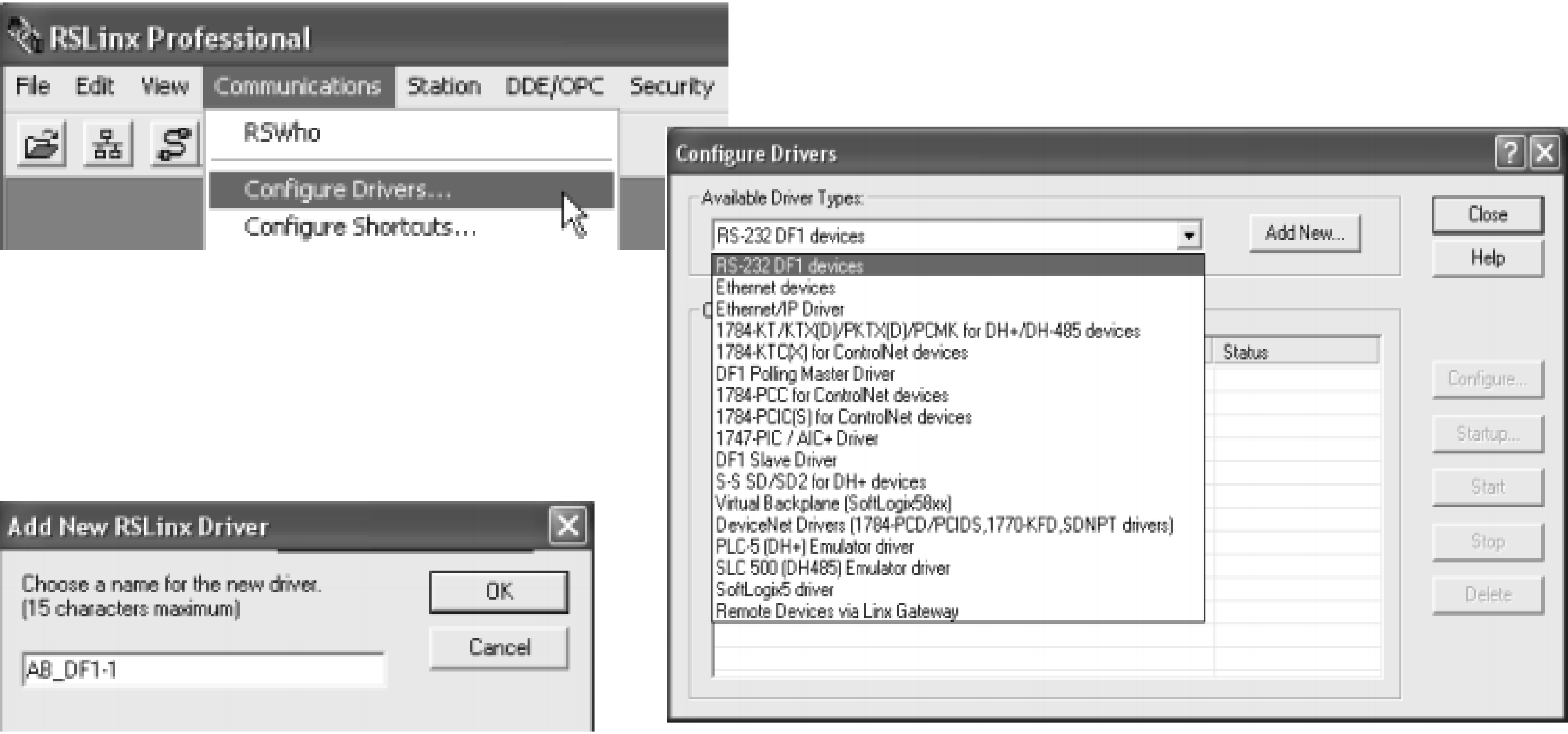
Controller

1 CD
2 RDX
3 TXD
4 DTR
COMMON
6 DSR
7 RTS
8 CTS
9



CONNECT **PC** TO **CPU**

Configure the serial driver via RSlinx



Available Driver Types: RS-232 DF1 devices

- RS-232 DF1 devices
- Ethernet devices
- Ethernet/IP Driver
- 1784-KT/KTX(D)/PKTX(D)/PCMK for DH+/DH-485 devices
- 1784-KTC(X) for ControlNet devices
- DF1 Polling Master Driver
- 1784-PCC for ControlNet devices
- 1784-PCIC(S) for ControlNet devices
- 1747-PIC / AIC+ Driver
- DF1 Slave Driver
- S-S SD/SD2 for DH+ devices
- Virtual Backplane (SoftLogix58xx)
- DeviceNet Drivers (1784-PCD/PCIDS,1770-KFD,SDNPT drivers)
- PLC-5 (DH+) Emulator driver
- SLC 500 (DH485) Emulator driver
- SoftLogix5 driver
- Remote Devices via Linx Gateway

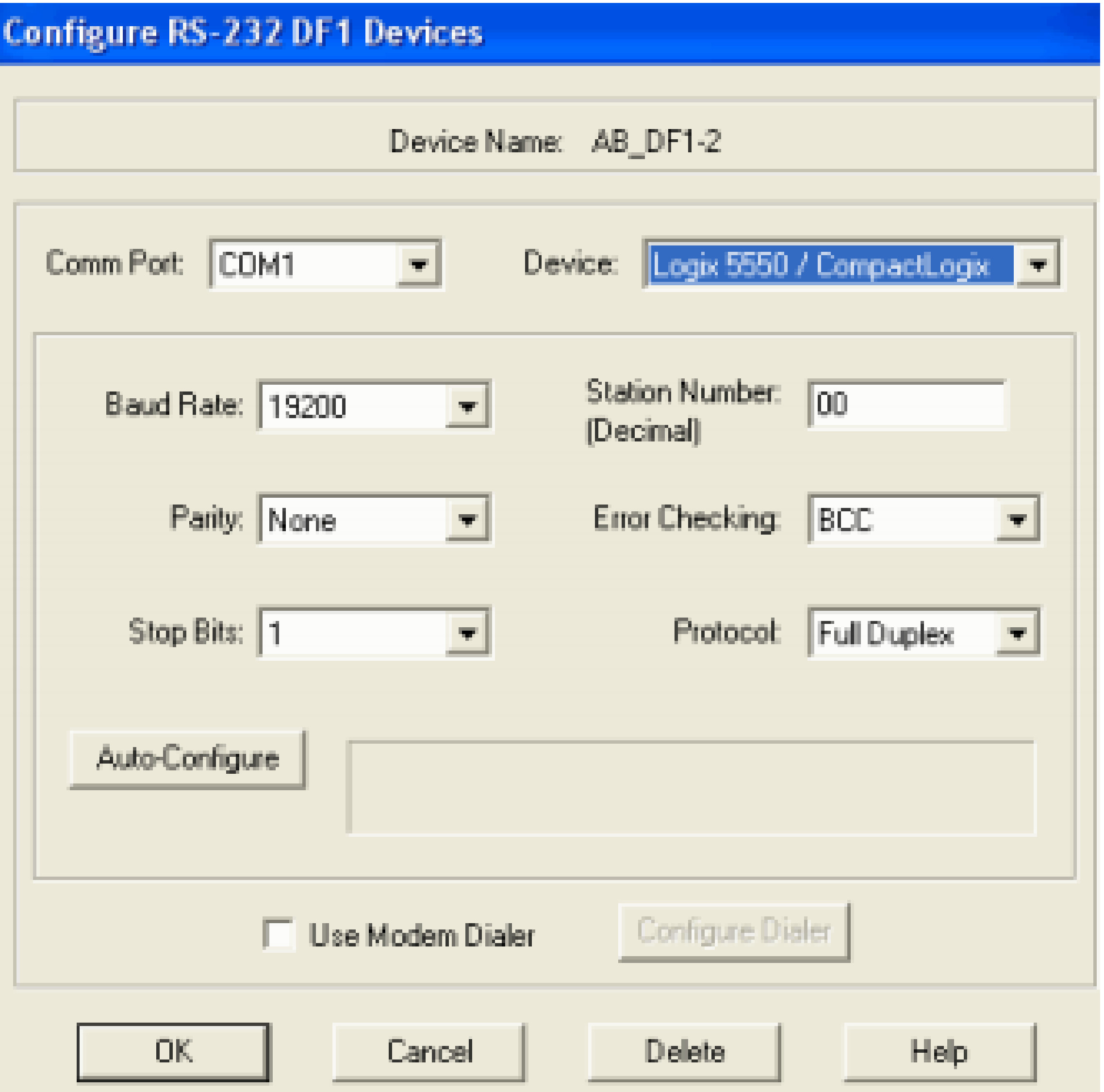
Choose a name for the new driver.
(15 characters maximum)

AB_DF1-1

From communication tab in Rslink, choose configure Driver, Rs232 DF1 devices, enter an appropriate name

CONNECT **PC** TO **CPU**

Configure the serial driver via RSlinx


Configure RS-232 DF1 Devices

Device Name: AB_DF1-2

Comm Port: COM1

Device: Logix 5550 / CompactLogix

Baud Rate: 19200

Station Number (Decimal): 00

Parity: None

Error Checking: BCC

Stop Bits: 1

Protocol: Full Duplex

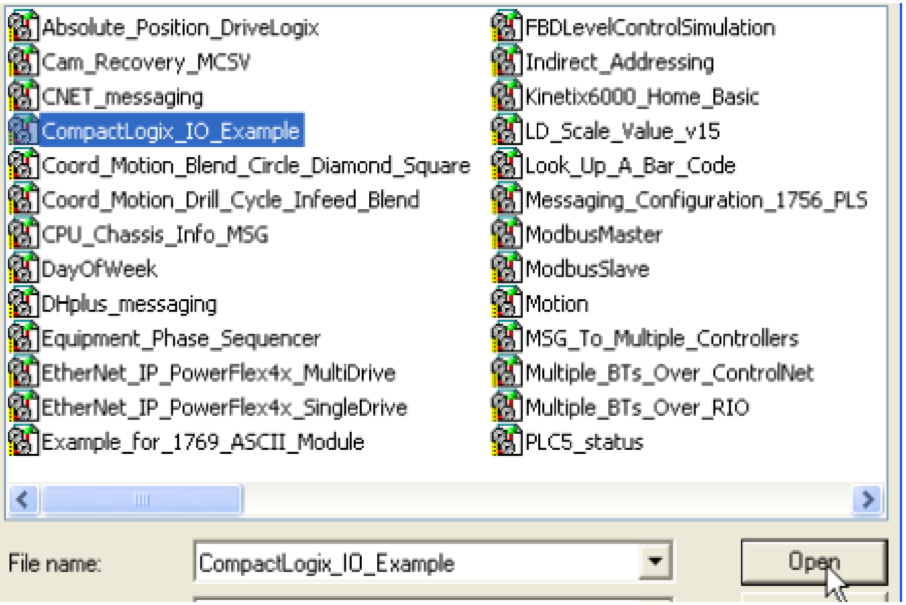
Setup parameters for

Configure RS 232 DF1 Devices

dialogs

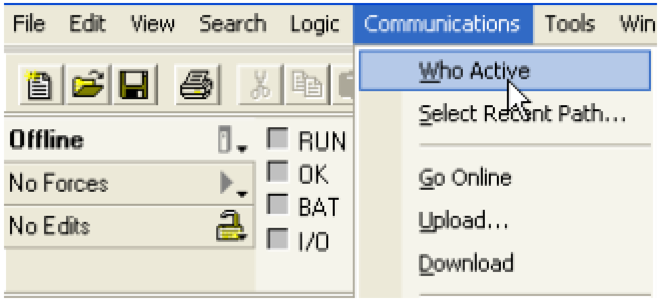
CONNECT **PC** TO **CPU**

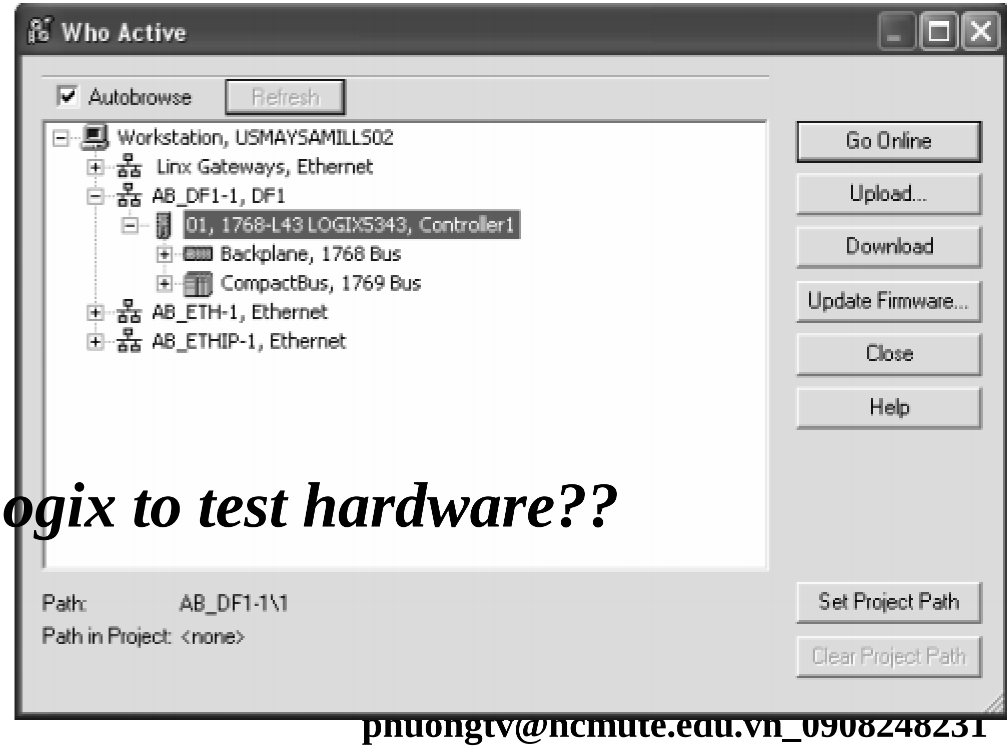
Select the Controller Path to download to the CPU: Open a project, choose Who Active then choose CPU to download



Absolute_Position_DriveLogix
Cam_Recovery_MCSV
CNET_messaging
CompactLogix_IO_Example
Coord_Motion_Blend_Circle_Diamond_Square
Coord_Motion_Drill_Cycle_Infeed_Blend
CPU_Chassis_Info_MSG
DayOfWeek
DHplus_messaging
Equipment_Phase_Sequencer
EtherNet_IP_PowerFlex4x_MultiDrive
EtherNet_IP_PowerFlex4x_SingleDrive
Example_for_1769_ASCII_Module
FBDLevelControlSimulation
Indirect_Addressing
Kinetix6000_Home_Basic
LD_Scale_Value_v15
Look_Up_A_Bar_Code
Messaging_Configuration_1756_PLS
ModbusMaster
ModbusSlave
Motion
MSG_To_Multiple_Controllers
Multiple_BTs_Over_ControlNet
Multiple_BTs_Over_RIO
PLC5_status

File name: CompactLogix_IO_Example





Who Active

☑ Autobrowse

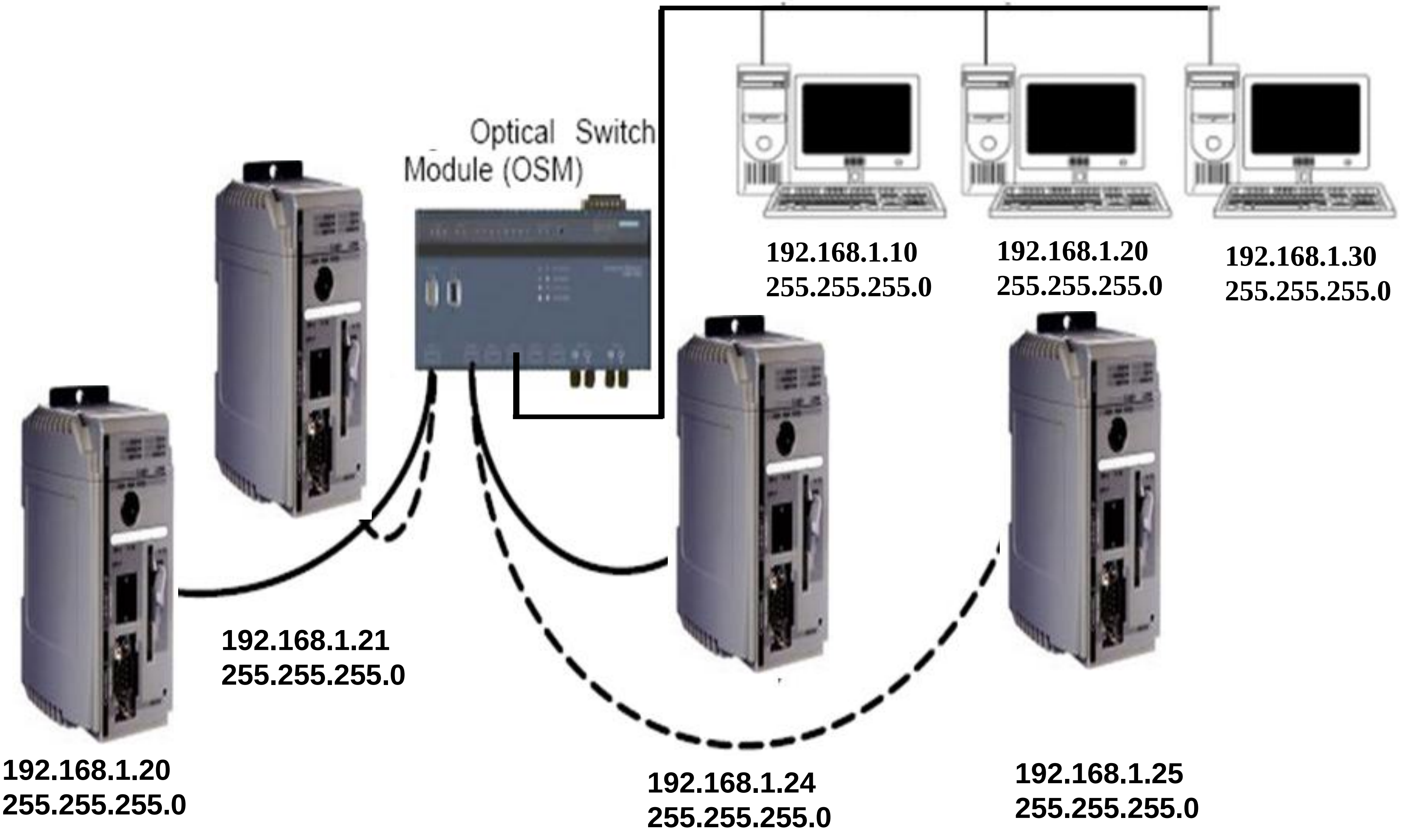
Workstation, USMAYSAMILLS02
Linx Gateways, Ethernet
AB_DF1-1, DF1
01, 1768-L43 LOGIX5343, Controller1
Backplane, 1768 Bus
CompactBus, 1769 Bus
AB_ETH-1, Ethernet
AB_ETHIP-1, Ethernet

Path: AB_DF1-1\1
Path in Project: <none>

Participants download to ComactLogix to test hardware??

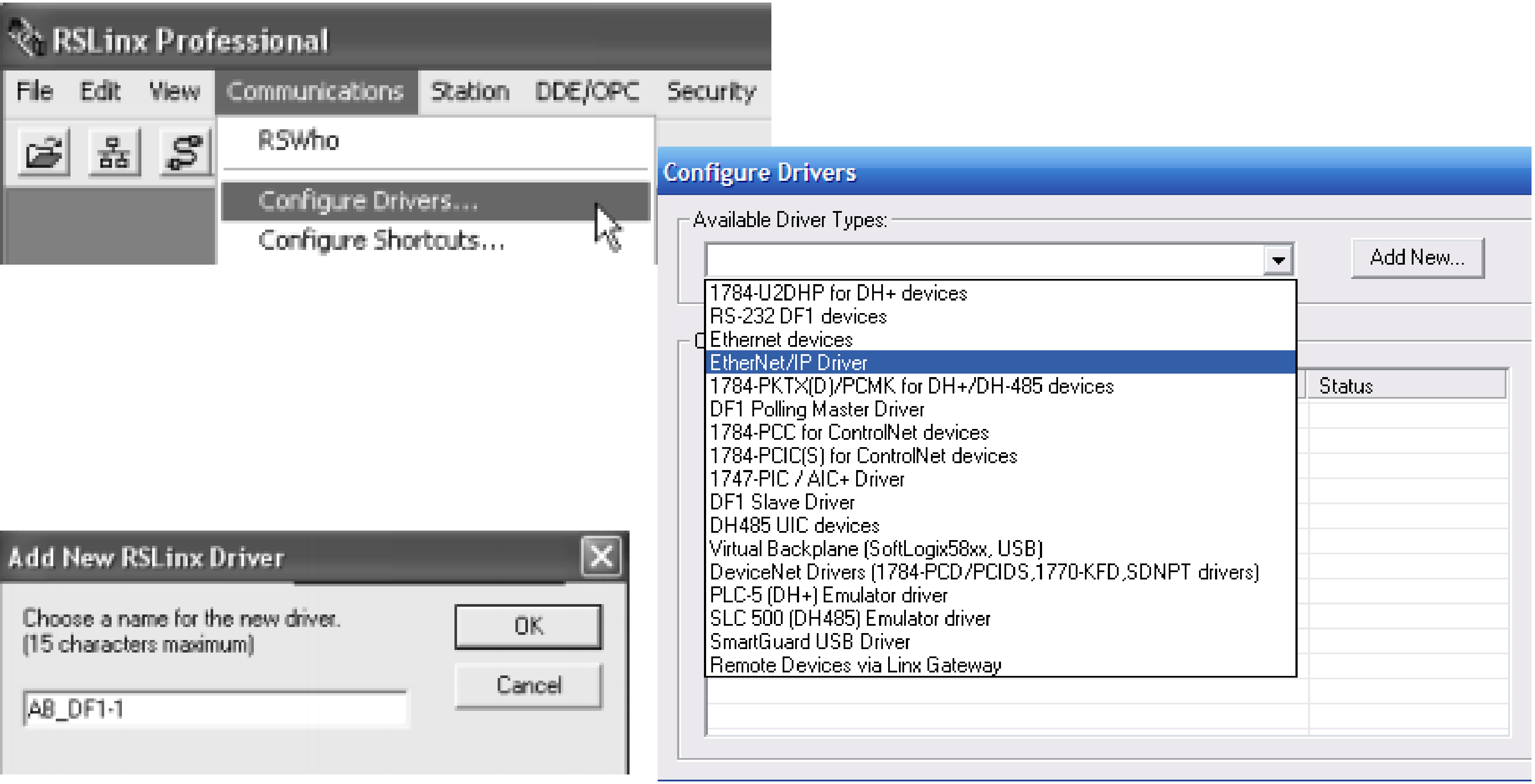
CONNECT **PC** TO **CPU**

Connect to the CPU via the Ethernet port



CONNECT **PC** TO **CPU**

Configure the Ethernet driver via RSlinx


RSLinx Professional
File
Edit
View
Communications
Station
DDE/OPC
Security
RSWho
Configure Drivers...
Configure Shortcuts...
Configure Drivers
Available Driver Types:
Add New...
1784-U2DHP for DH+ devices
RS-232 DF1 devices
Ethernet devices
EtherNet/IP Driver
1784-PKTX(D)/PCMK for DH+/DH-485 devices
DF1 Polling Master Driver
1784-PCC for ControlNet devices
1784-PCIC(S) for ControlNet devices
1747-PIC / AIC+ Driver
DF1 Slave Driver
DH485 UIC devices
Virtual Backplane (SoftLogix58xx, USB)
DeviceNet Drivers (1784-PCD/PCIDS,1770-KFD,SDNPT drivers)
PLC-5 (DH+) Emulator driver
SLC 500 (DH485) Emulator driver
SmartGuard USB Driver
Remote Devices via Linx Gateway
Status
Add New RSLinx Driver
OK
Cancel

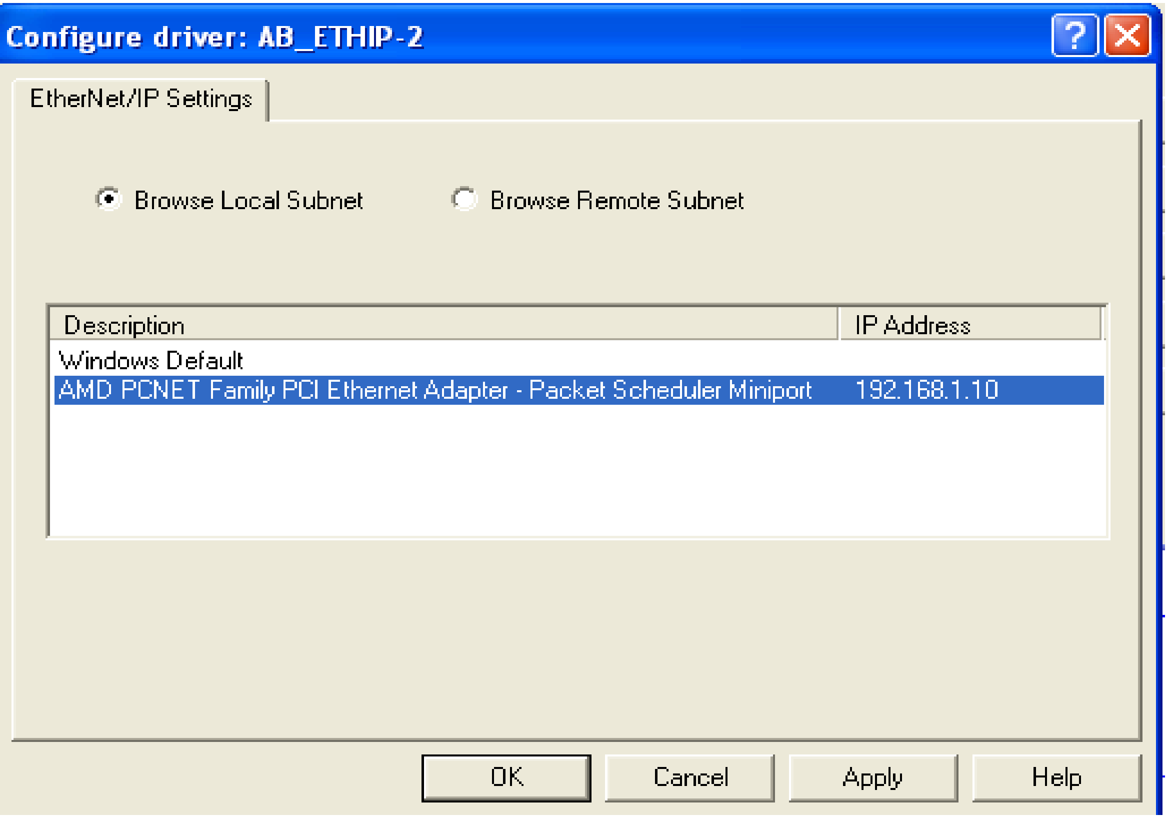
Choose a name for the new driver.
(15 characters maximum)

AB_DF1-1

From communication tab in Rslink, choose configure Driver, Ethernet/IP Driver, enter an appropriate name

CONNECT **PC** TO **CPU**

Configure the Ethernet driver via RSlinX



Configure driver: AB_ETHIP-2

EtherNet/IP Settings

● Browse Local Subnet ○ Browse Remote Subnet

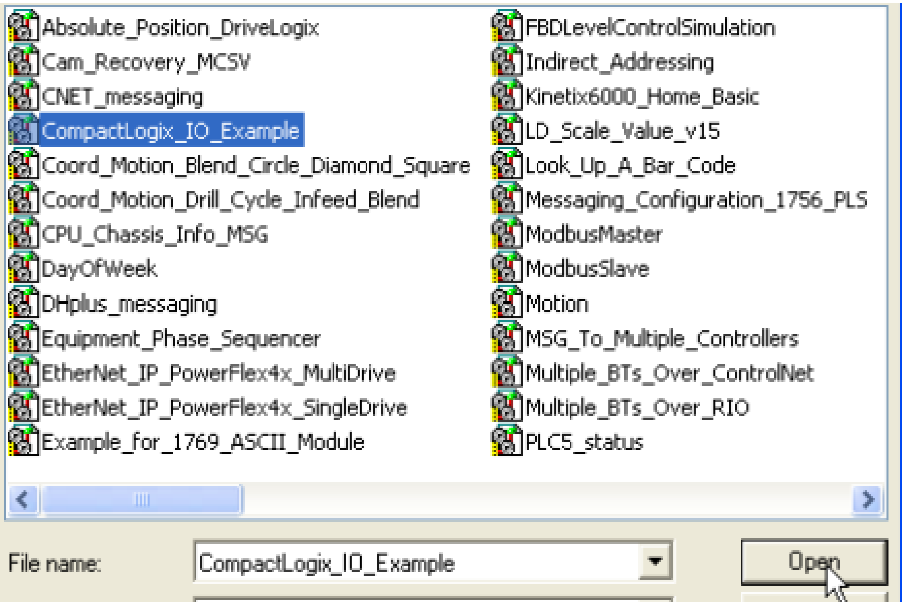
Description	IP Address
Windows Default	
AMD PCNET Family PCI Ethernet Adapter - Packet Scheduler Miniport	192.168.1.10

Choose Network connection

and IP address

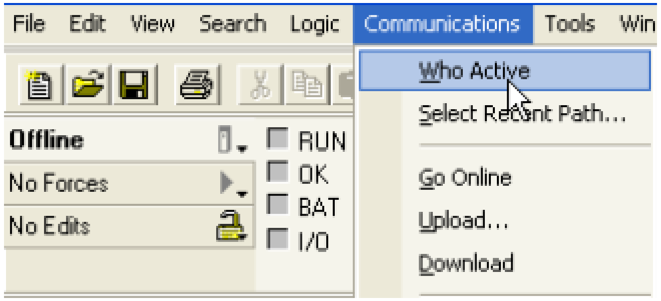
CONNECT **PC** TO **CPU**

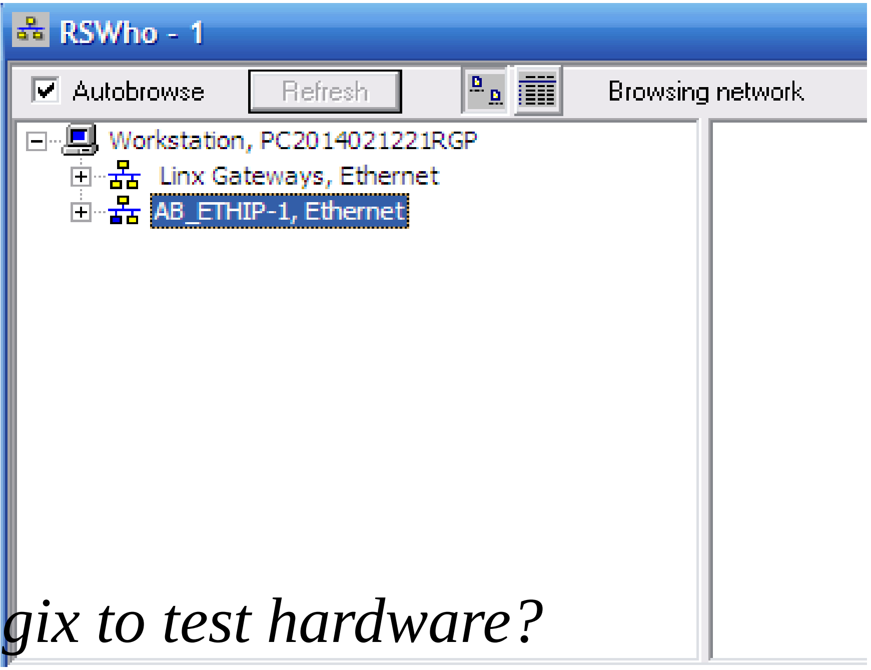
Select the Controller Path to download to the CPU: Open a project, choose Who Active then choose CPU to download via ethernet



Absolute_Position_DriveLogix
Cam_Recovery_MCSV
CNET_messaging
CompactLogix_IO_Example
Coord_Motion_Blend_Circle_Diamond_Square
Coord_Motion_Drill_Cycle_Infeed_Blend
CPU_Chassis_Info_MSG
DayOfWeek
DHplus_messaging
Equipment_Phase_Sequencer
EtherNet_IP_PowerFlex4x_MultiDrive
EtherNet_IP_PowerFlex4x_SingleDrive
Example_for_1769_ASCII_Module
FBDLevelControlSimulation
Indirect_Addressing
Kinetix6000_Home_Basic
LD_Scale_Value_v15
Look_Up_A_Bar_Code
Messaging_Configuration_1756_PLS
ModbusMaster
ModbusSlave
Motion
MSG_To_Multiple_Controllers
Multiple_BTs_Over_ControlNet
Multiple_BTs_Over_RIO
PLC5_status

File name: CompactLogix_IO_Example



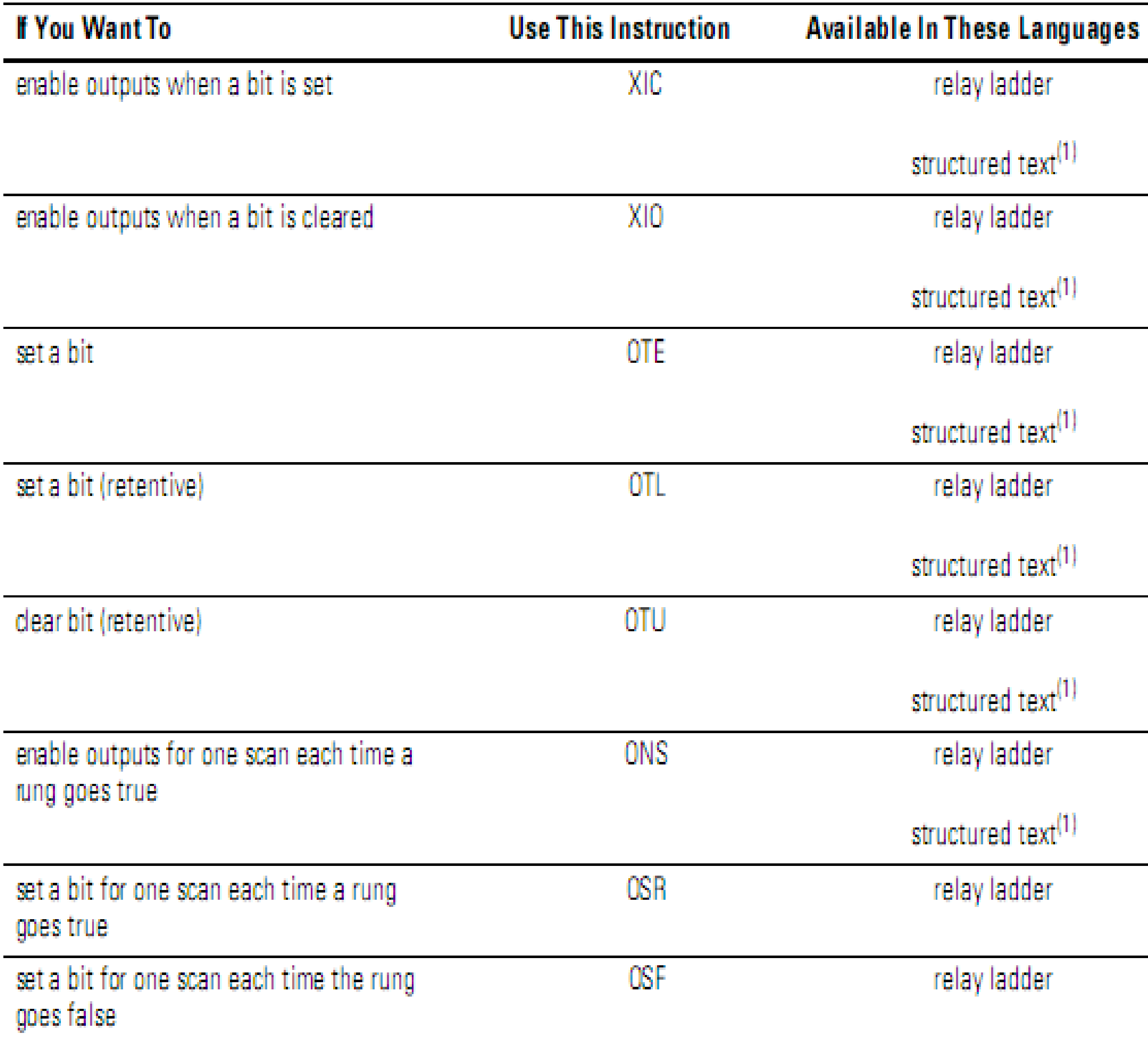


Workstation, PC2014021221RGP
Linx Gateways, Ethernet
AB_ETHIP-1, Ethernet

Participants download to ComactLogix to test hardware?

BASIC INSTRUCTION

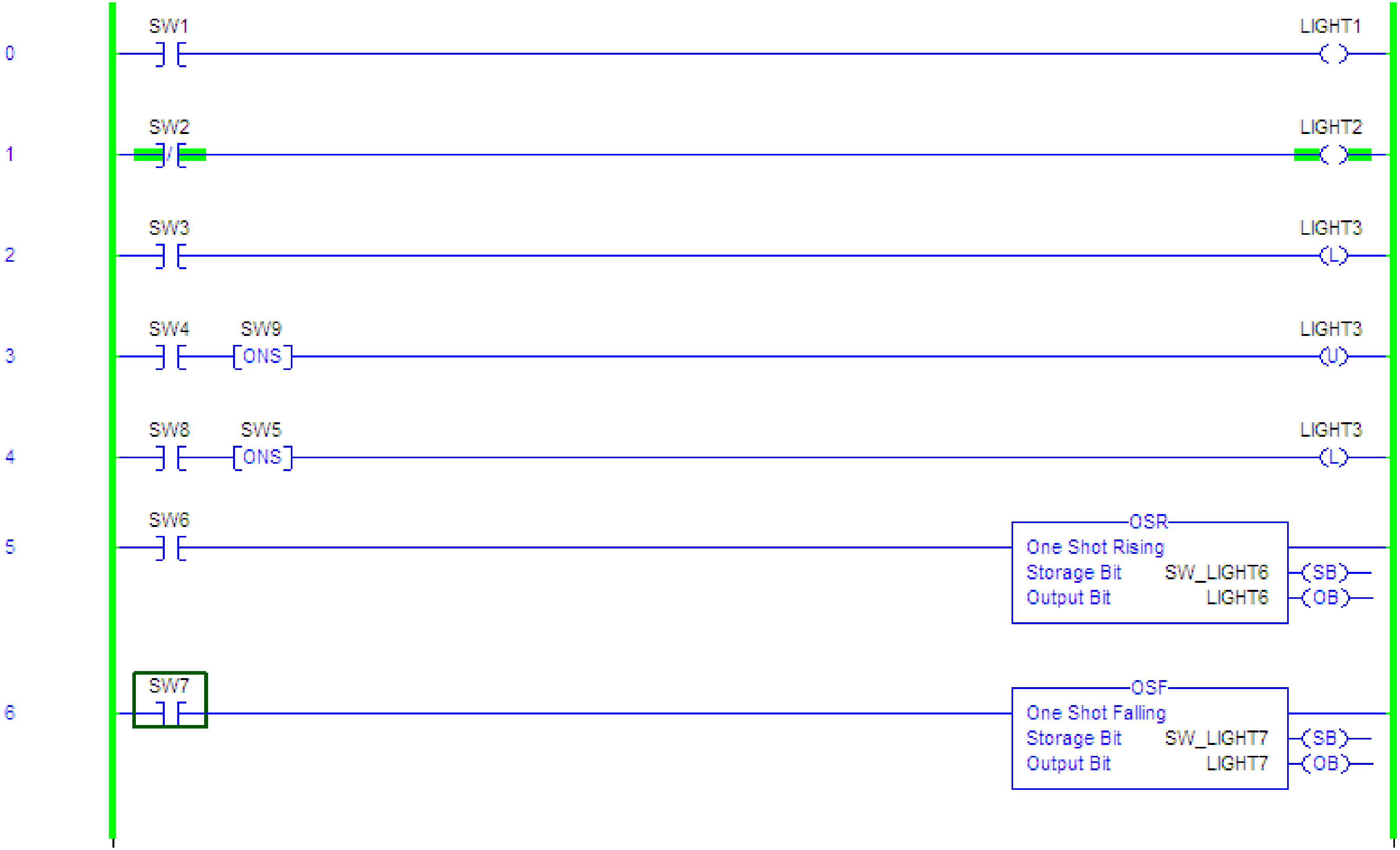
Bit instructions



If You Want To	Use This Instruction	Available In These Languages
enable outputs when a bit is set	XIC	relay ladder structured text[1]
enable outputs when a bit is cleared	XIO	relay ladder structured text[1]
set a bit	OTE	relay ladder structured text[1]
set a bit (retentive)	OTL	relay ladder structured text[1]
clear bit (retentive)	OTU	relay ladder structured text[1]
enable outputs for one scan each time a rung goes true	ONS	relay ladder structured text[1]
set a bit for one scan each time a rung goes true	OSR	relay ladder
set a bit for one scan each time the rung goes false	OSF	relay ladder

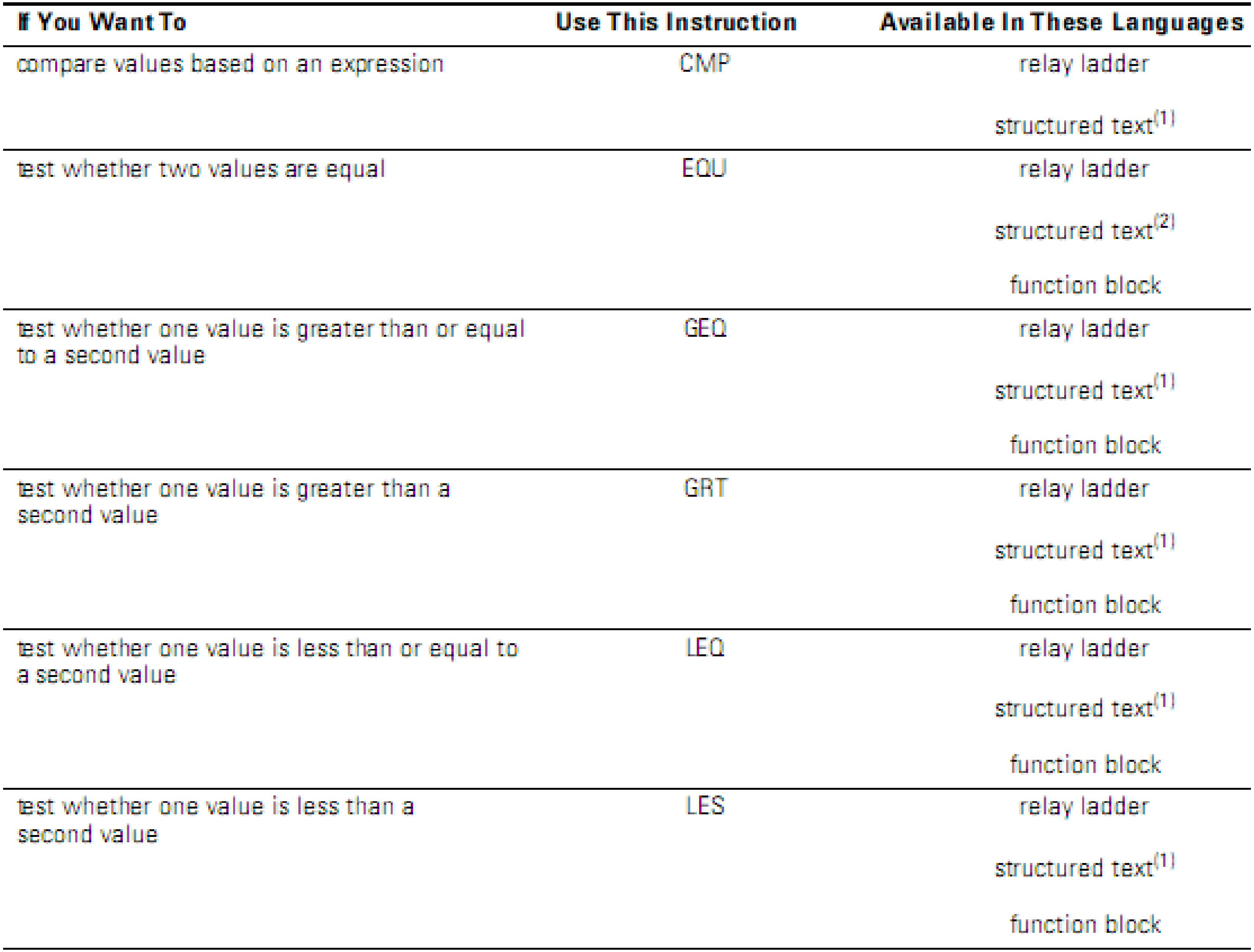
BASIC INSTRUCTION

Bit instructions



BASIC INSTRUCTION

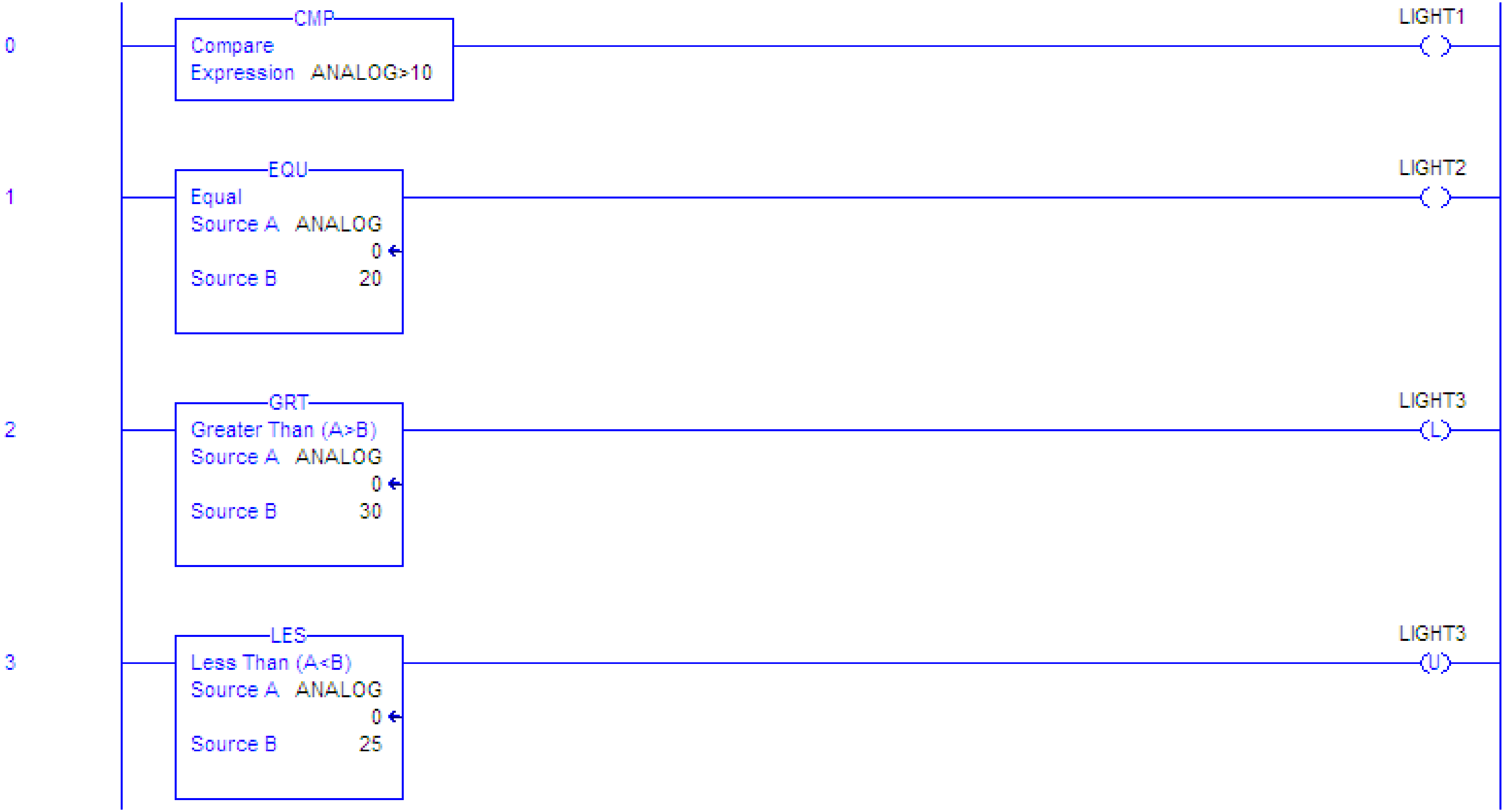
Compare instructions



If You Want To	Use This Instruction	Available In These Languages
compare values based on an expression	CMP	relay ladder structured text[1]
test whether two values are equal	EQU	relay ladder structured text[2] function block
test whether one value is greater than or equal to a second value	GEQ	relay ladder structured text[1] function block
test whether one value is greater than a second value	GRT	relay ladder structured text[1] function block
test whether one value is less than or equal to a second value	LEQ	relay ladder structured text[1] function block
test whether one value is less than a second value	LES	relay ladder structured text[1] function block

BASIC INSTRUCTION

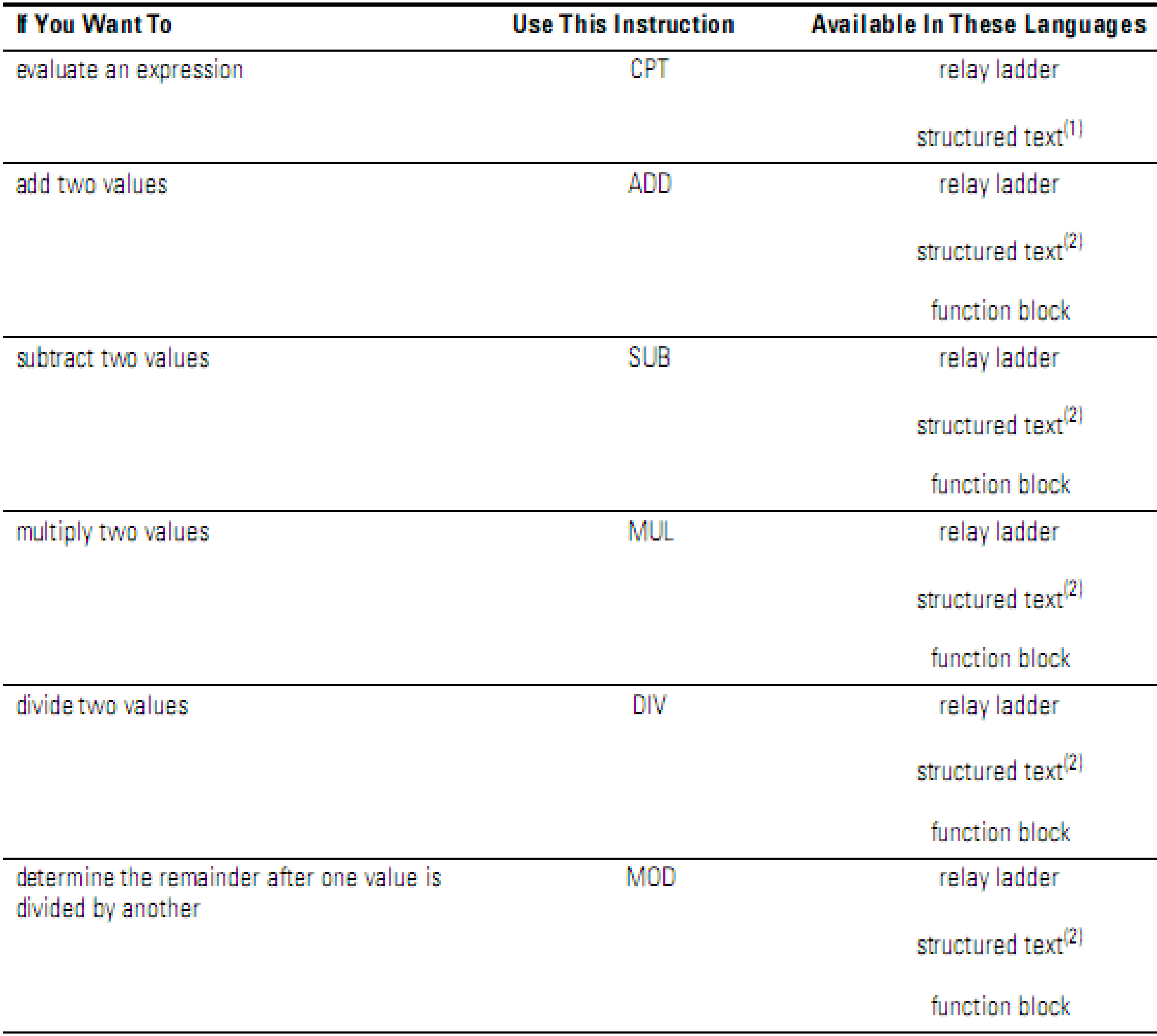
Compare instruction



```
0  CMP
   Compare
   Expression  ANALOG>10                                   LIGHT1
                                                           ( )

1  EQU
   Equal
   Source A  ANALOG
                  0 ←
   Source B      20                                        LIGHT2
                                                           ( )

2  GRT
   Greater Than (A>B)
   Source A  ANALOG
                  0 ←
   Source B      30                                        LIGHT3
                                                           (L)

3  LES
   Less Than (A<B)
   Source A  ANALOG
                  0 ←
   Source B      25                                        LIGHT3
                                                           (U)
```

BASIC INSTRUCTION

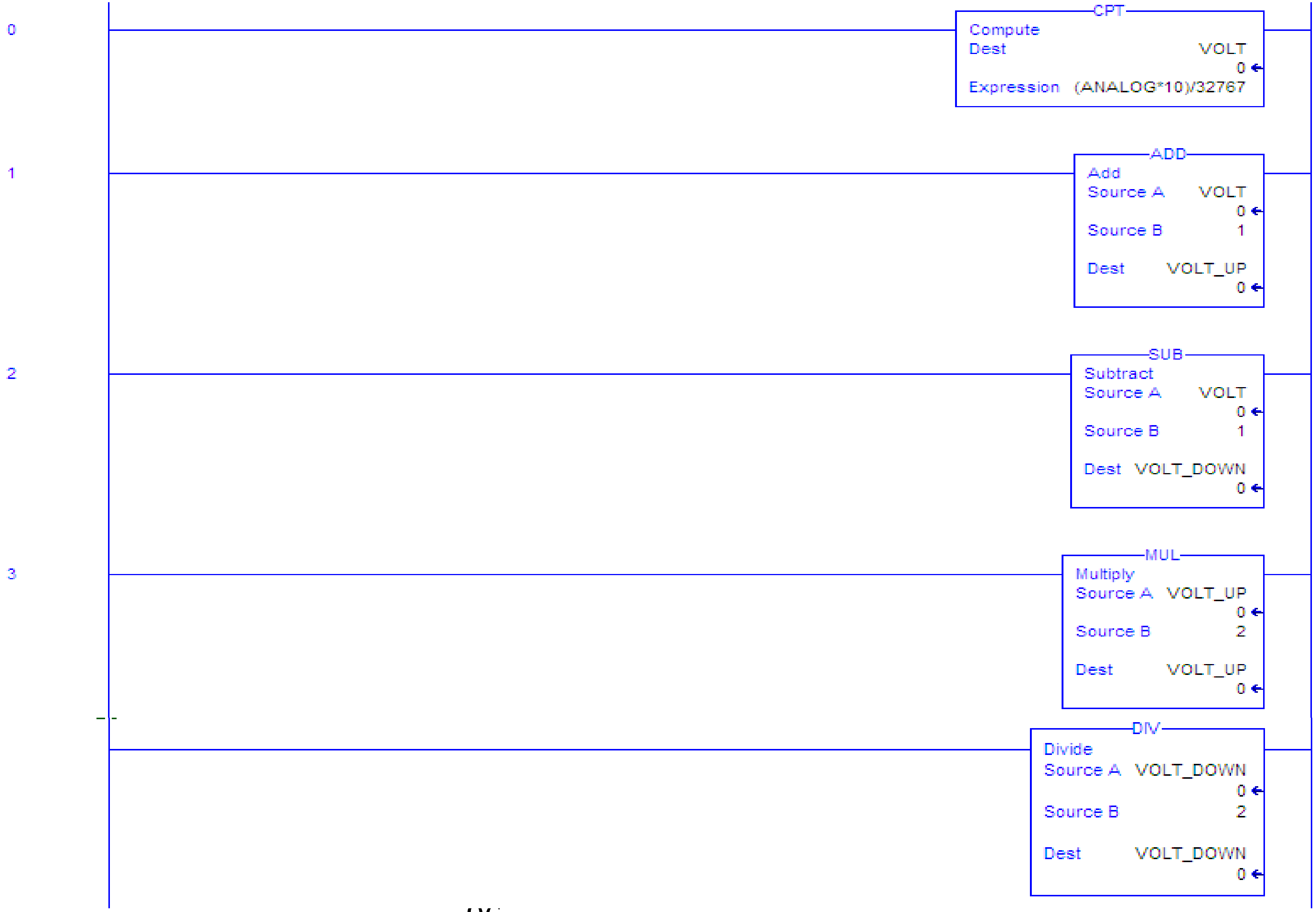
Math instructions



If You Want To	Use This Instruction	Available In These Languages
evaluate an expression	CPT	relay ladder structured text[1]
add two values	ADD	relay ladder structured text[2] function block
subtract two values	SUB	relay ladder structured text[2] function block
multiply two values	MUL	relay ladder structured text[2] function block
divide two values	DIV	relay ladder structured text[2] function block
determine the remainder after one value is divided by another	MOD	relay ladder structured text[2] function block

BASIC INSTRUCTION

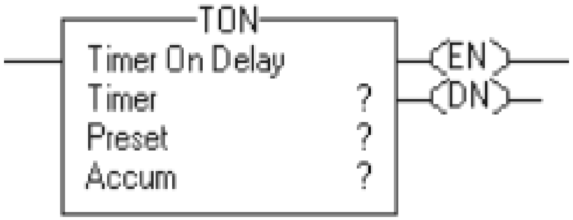
Math instruction

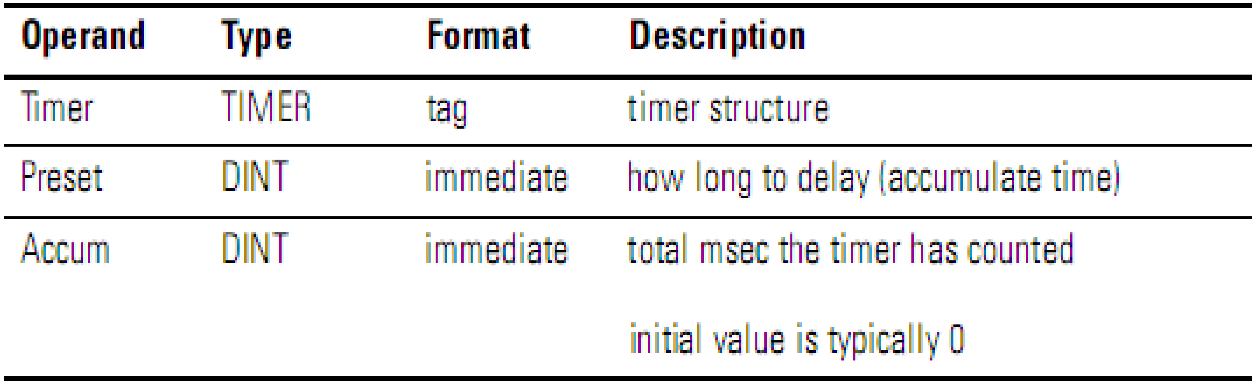


```
0   CPT
    Compute
    Dest                      VOLT
                                 0
    Expression  (ANALOG*10)/32767

1   ADD
    Add
    Source A      VOLT
                     0
    Source B         1
    Dest       VOLT_UP
                     0

2   SUB
    Subtract
    Source A        VOLT
                       0
    Source B           1
    Dest   VOLT_DOWN
                       0

3   MUL
    Multiply
    Source A   VOLT_UP
                     0
    Source B         2
    Dest       VOLT_UP
                     0

    DIV
    Divide
    Source A   VOLT_DOWN
                       0
    Source B           2
    Dest       VOLT_DOWN
                       0
```

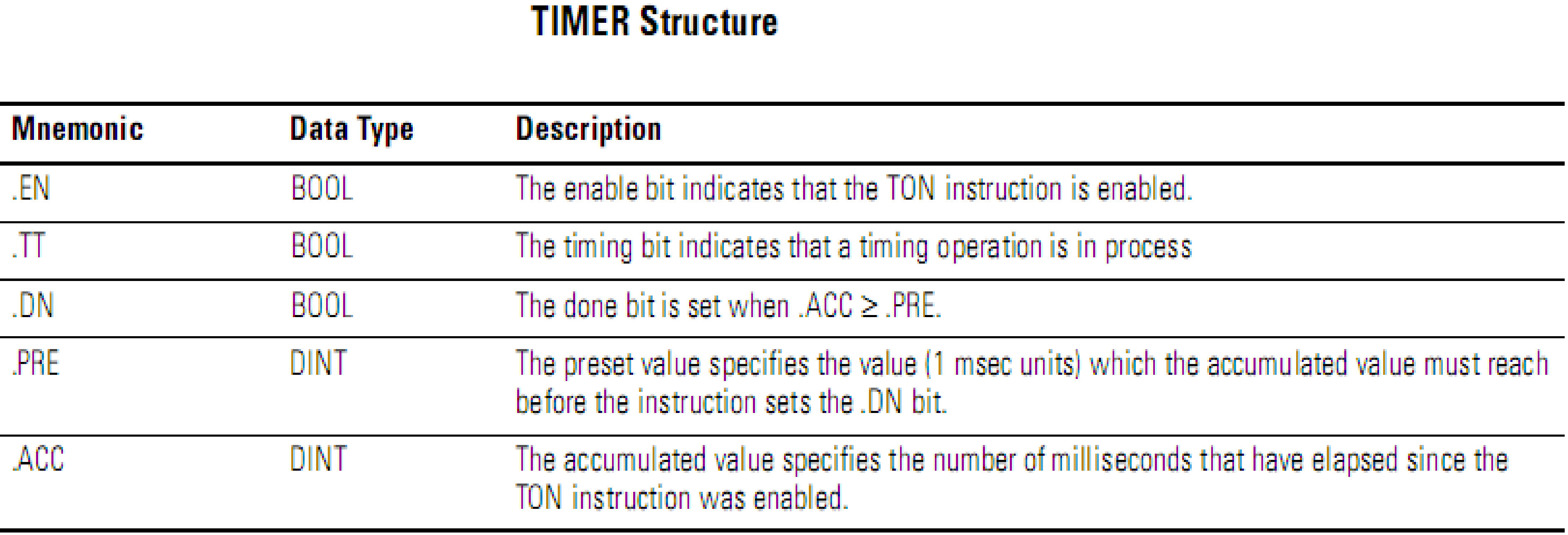
BASIC INSTRUCTION

Timer





Operand	Type	Format	Description
Timer	TIMER	tag	timer structure
Preset	DINT	immediate	how long to delay (accumulate time)
Accum	DINT	immediate	total msec the timer has counted initial value is typically 0



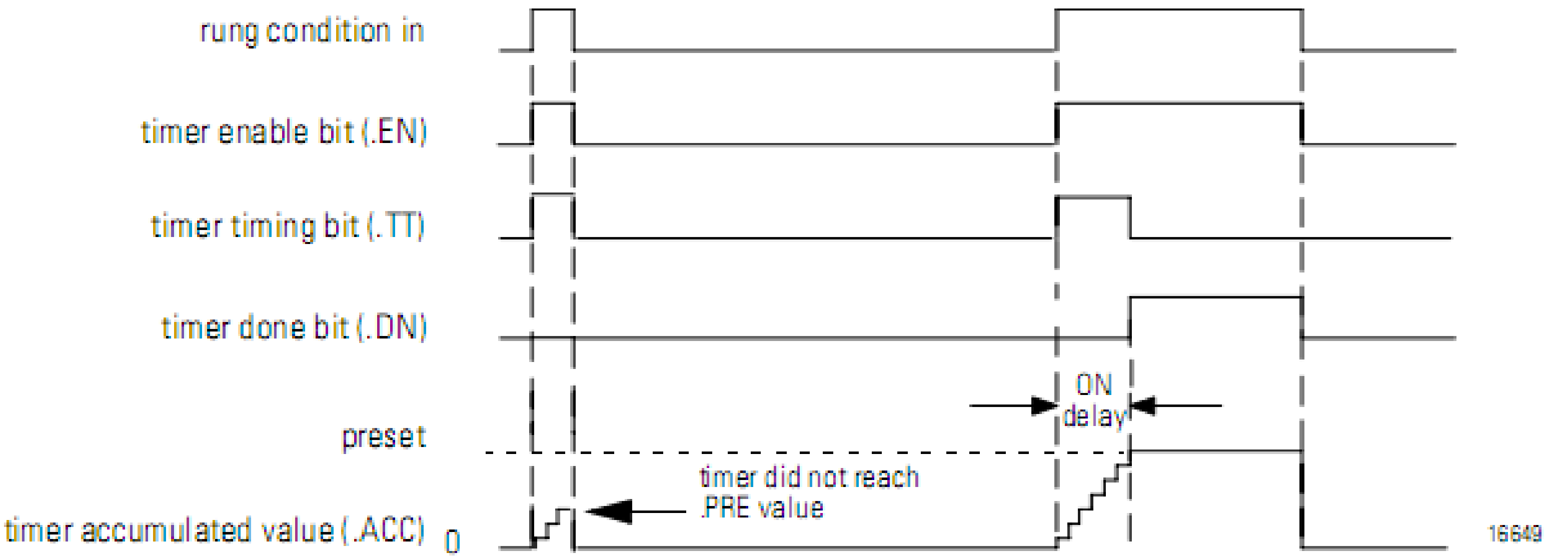
TIMER Structure

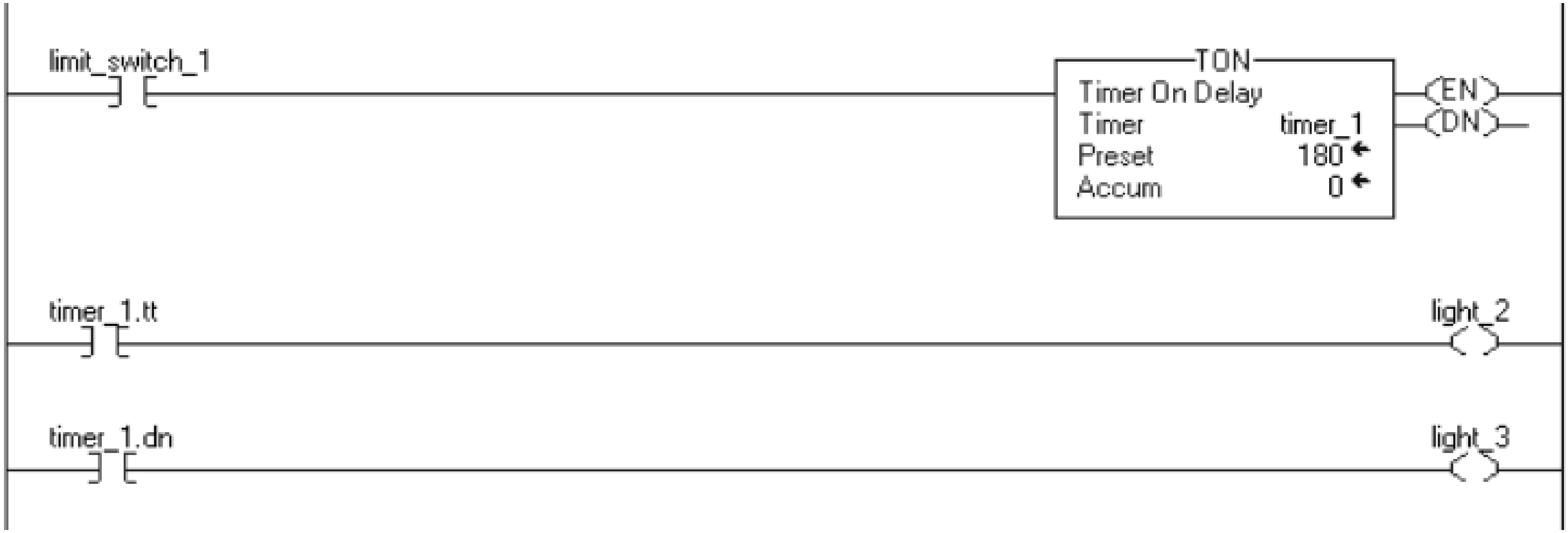
Mnemonic	Data Type	Description
.EN	BOOL	The enable bit indicates that the TON instruction is enabled.
.TT	BOOL	The timing bit indicates that a timing operation is in process
.DN	BOOL	The done bit is set when .ACC ≥ .PRE.
.PRE	DINT	The preset value specifies the value (1 msec units) which the accumulated value must reach before the instruction sets the .DN bit.
.ACC	DINT	The accumulated value specifies the number of milliseconds that have elapsed since the TON instruction was enabled.

BASIC INSTRUCTION

Timer

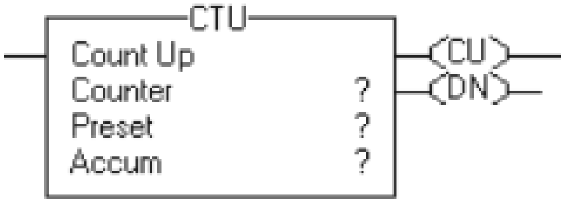
When the TON instruction is disabled, the .ACC value is cleared.

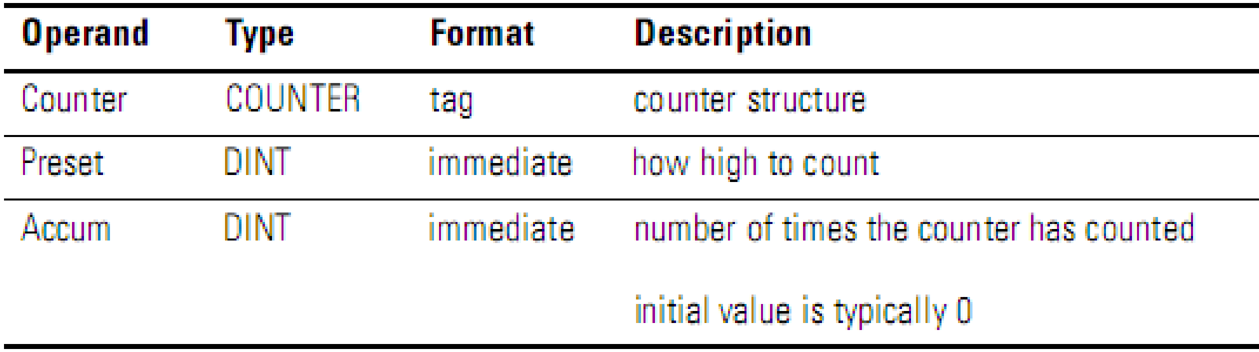




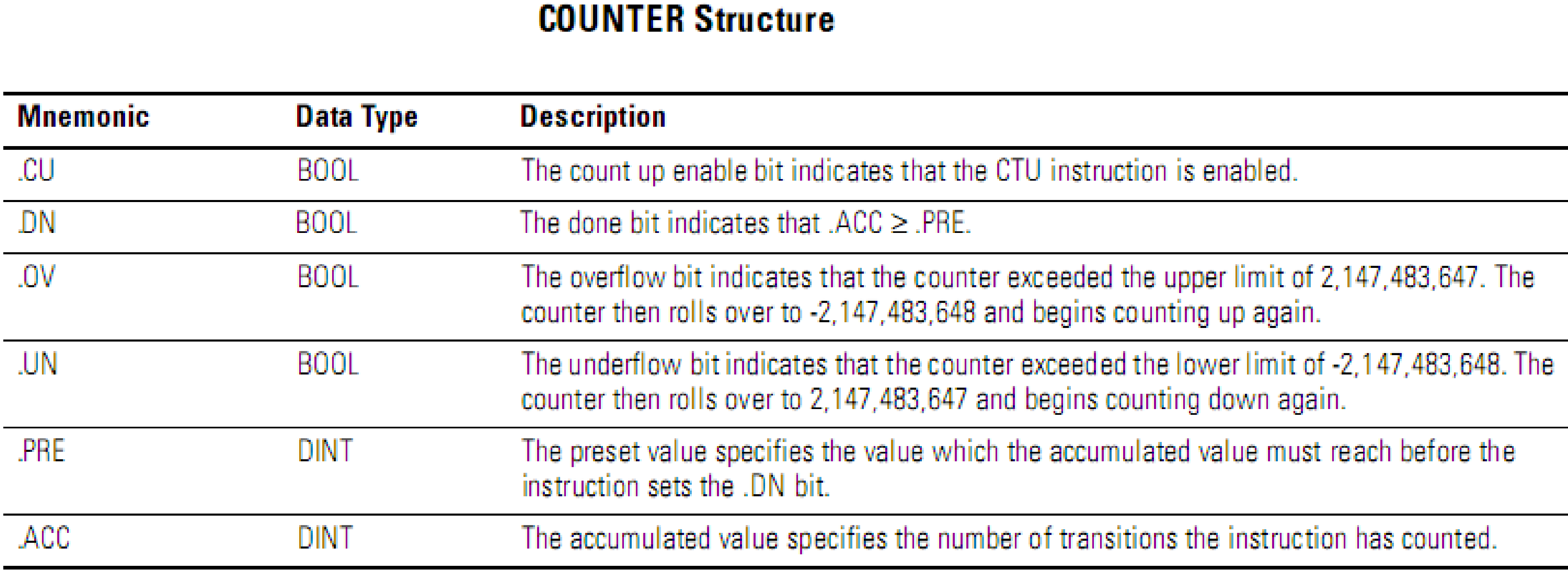
BASIC INSTRUCTION

Counter





Operand	Type	Format	Description
Counter	COUNTER	tag	counter structure
Preset	DINT	immediate	how high to count
Accum	DINT	immediate	number of times the counter has counted initial value is typically 0

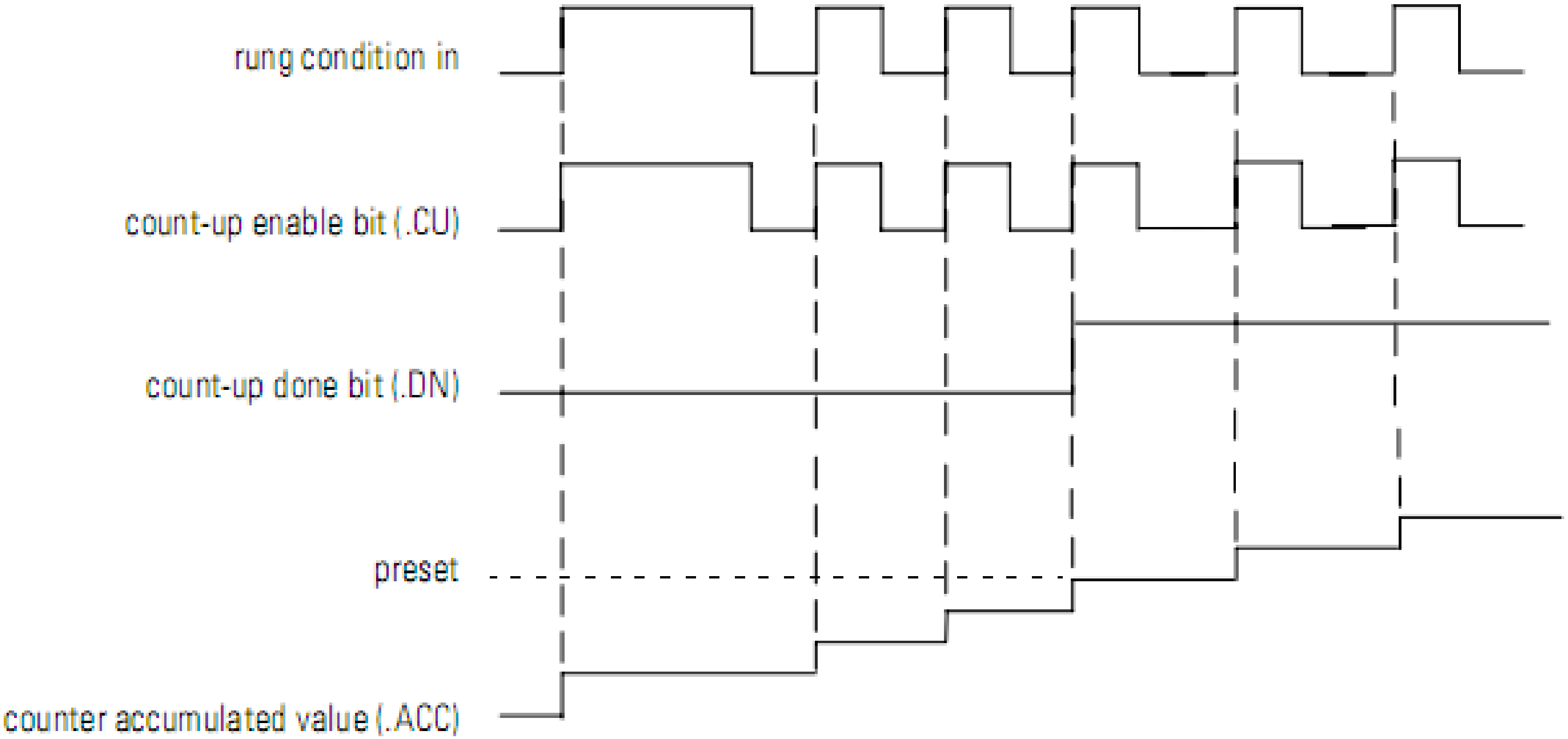


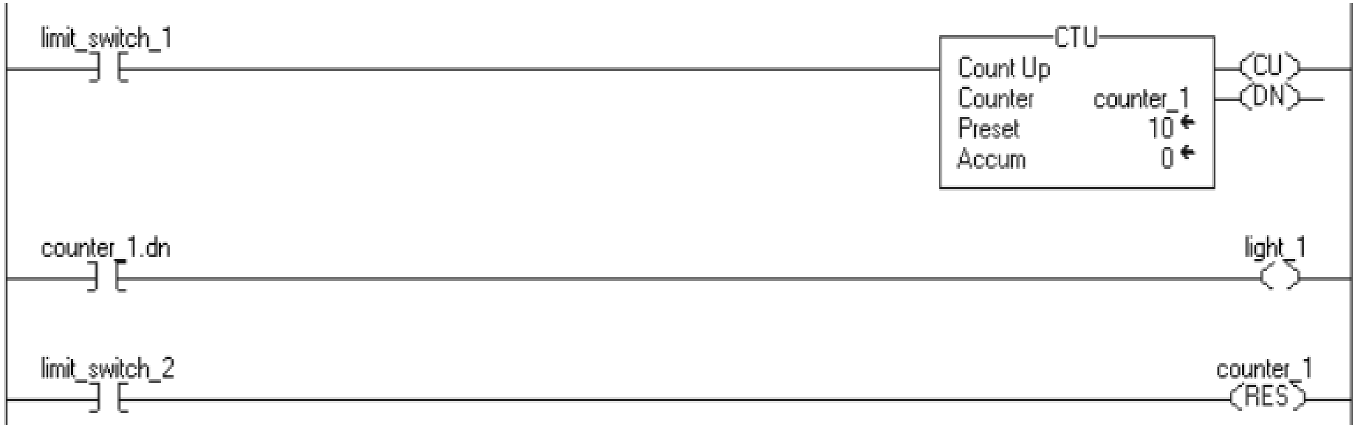
COUNTER Structure

Mnemonic	Data Type	Description
.CU	BOOL	The count up enable bit indicates that the CTU instruction is enabled.
.DN	BOOL	The done bit indicates that .ACC ≥ .PRE.
.OV	BOOL	The overflow bit indicates that the counter exceeded the upper limit of 2,147,483,647. The counter then rolls over to -2,147,483,648 and begins counting up again.
.UN	BOOL	The underflow bit indicates that the counter exceeded the lower limit of -2,147,483,648. The counter then rolls over to 2,147,483,647 and begins counting down again.
.PRE	DINT	The preset value specifies the value which the accumulated value must reach before the instruction sets the .DN bit.
.ACC	DINT	The accumulated value specifies the number of transitions the instruction has counted.

BASIC INSTRUCTION

Counter

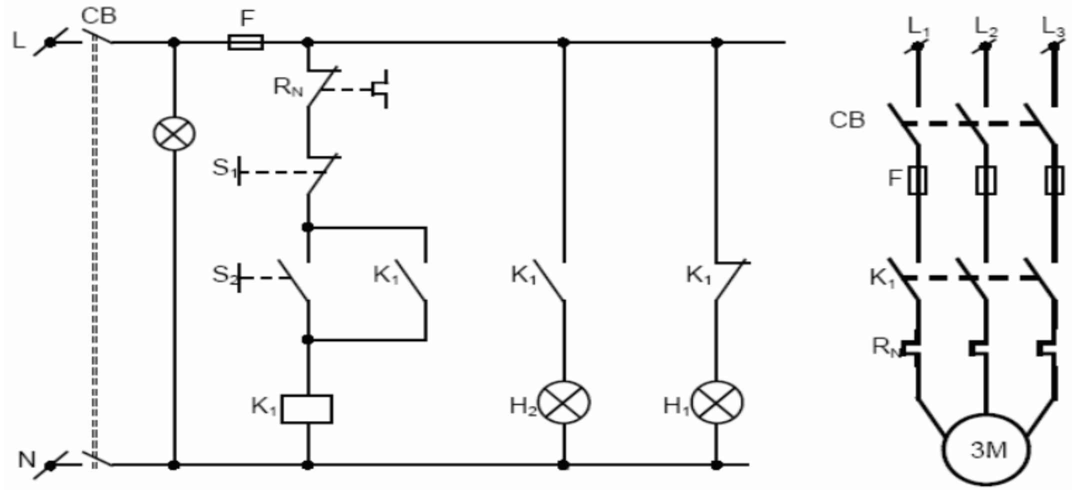




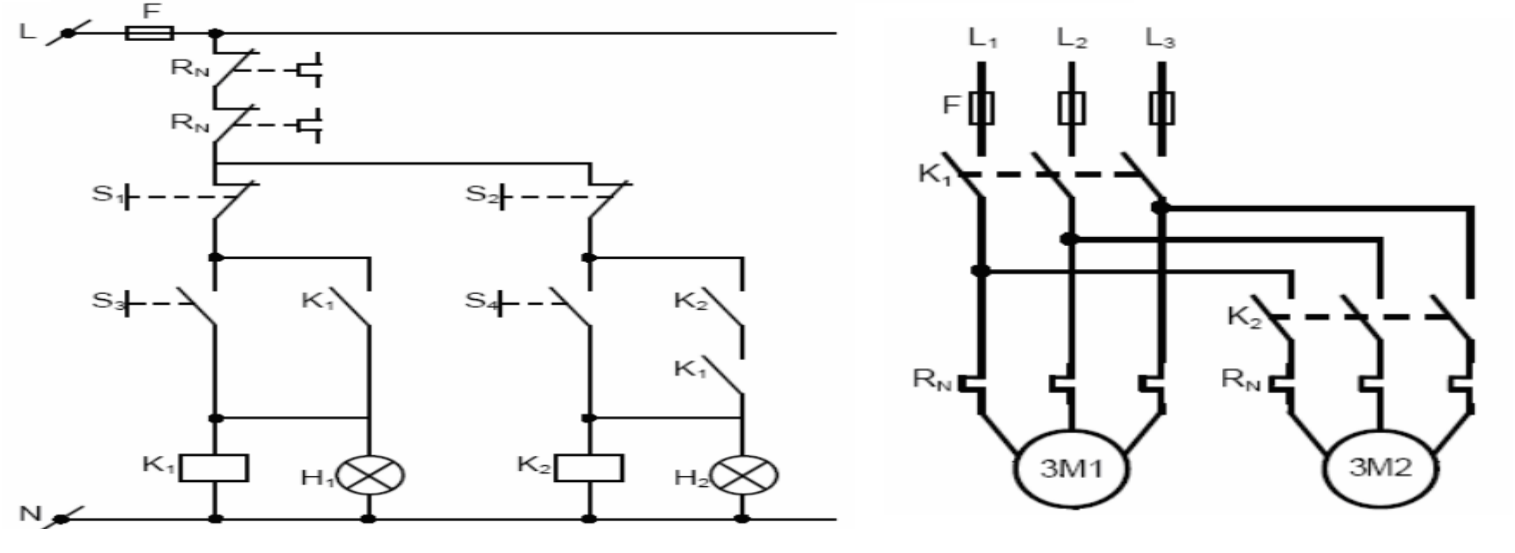
EXAMPLE OF INSTRUCTIONS

Using LAD, FBD, ST, SFC to program for relay control circuits from Ex11 to Ex13

Ex11

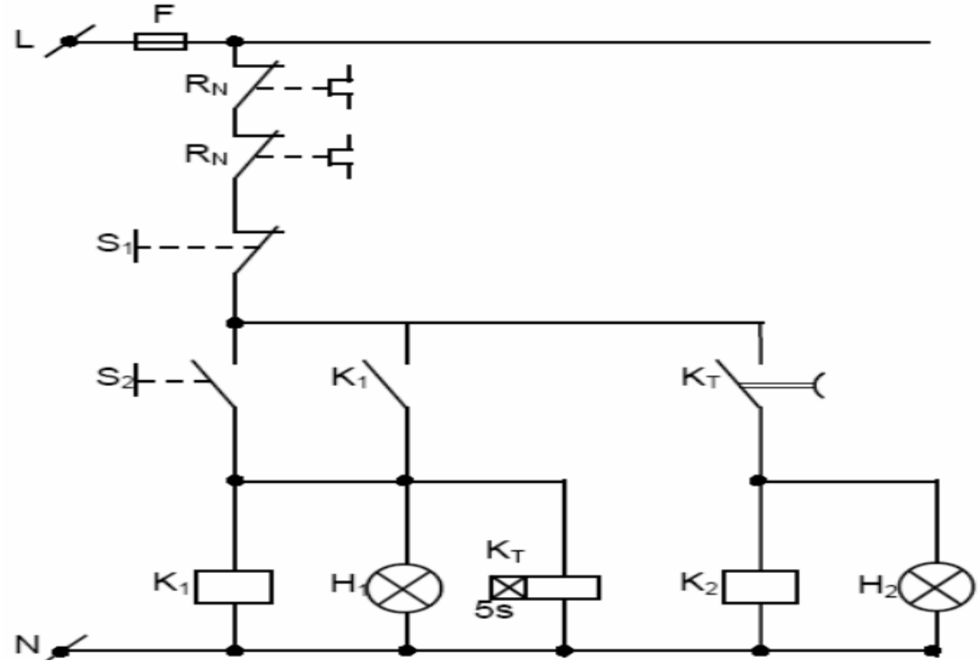


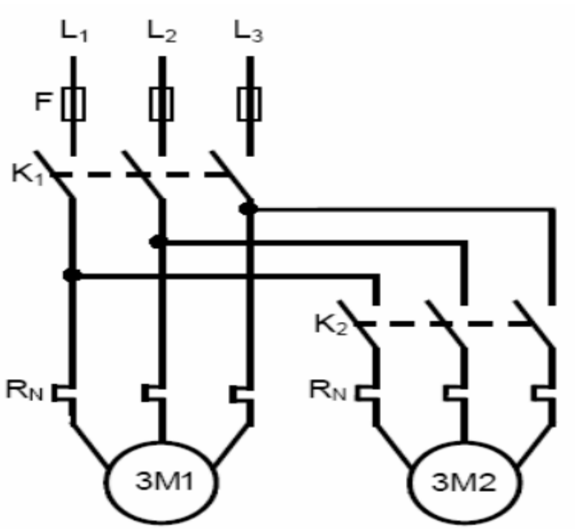
Ex12



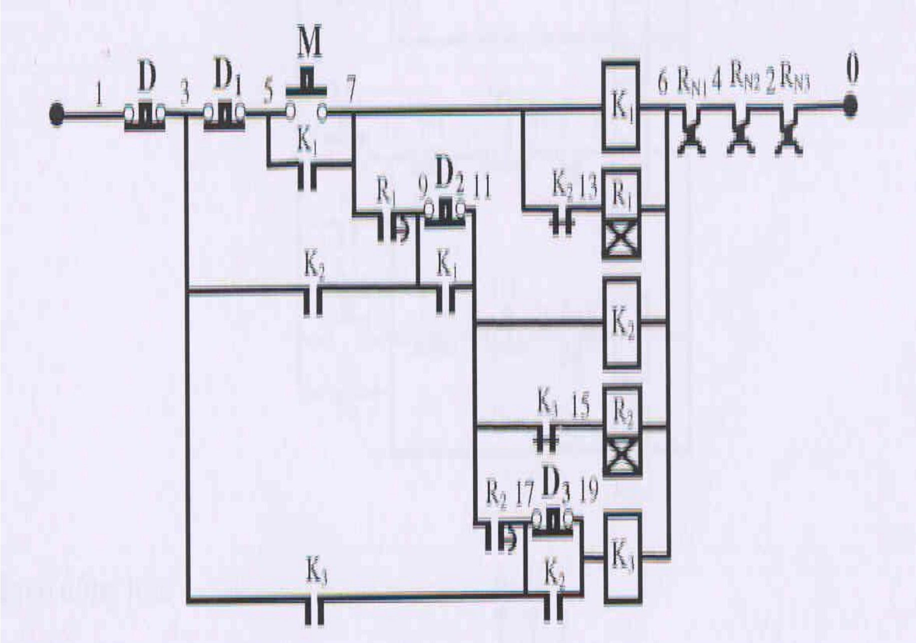
EXAMPLE OF INSTRUCTIONS

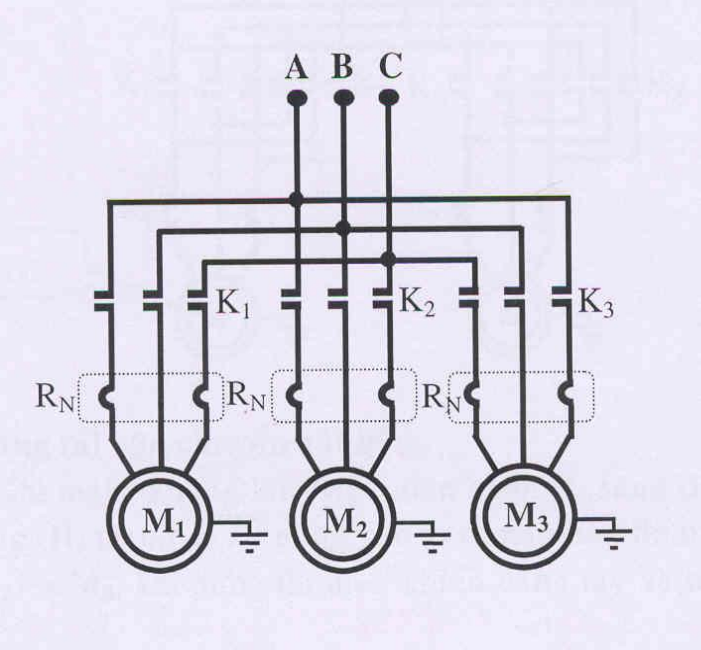
Ex13:





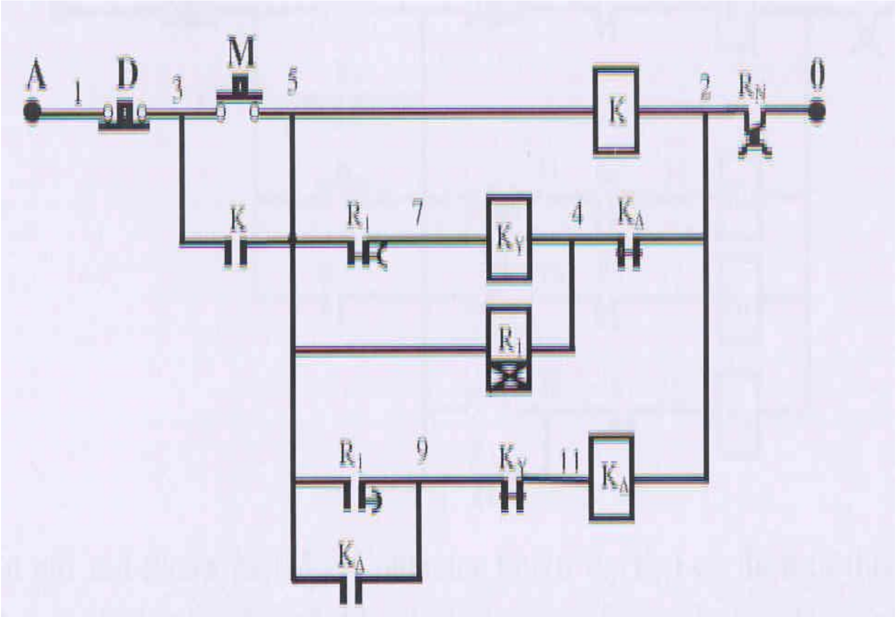
Ex14:

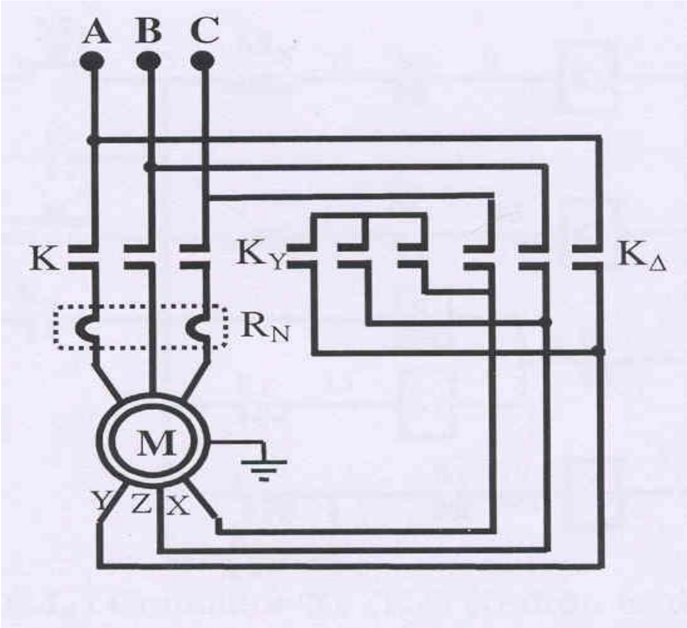




EXAMPLES OF INSTRUCTION

Ex15:

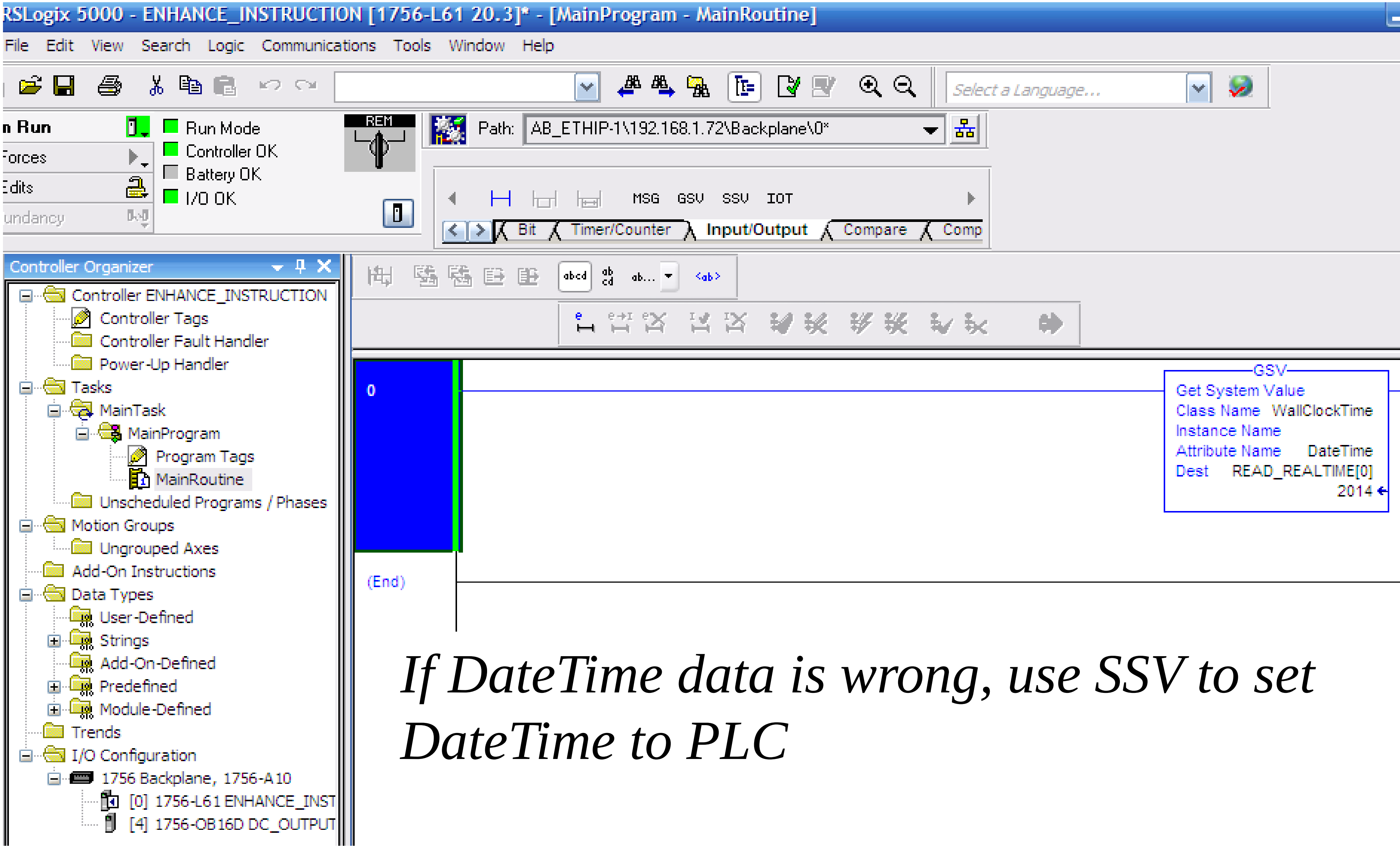




MSG, GSV, SSV INSTRUCTIONS

Use GSV instruction to read and store Realtime in plc

Depend on your applications, which data in array is used



RSLogix 5000 - ENHANCE_INSTRUCTION [1756-L61 20.3]* - [MainProgram - MainRoutine]

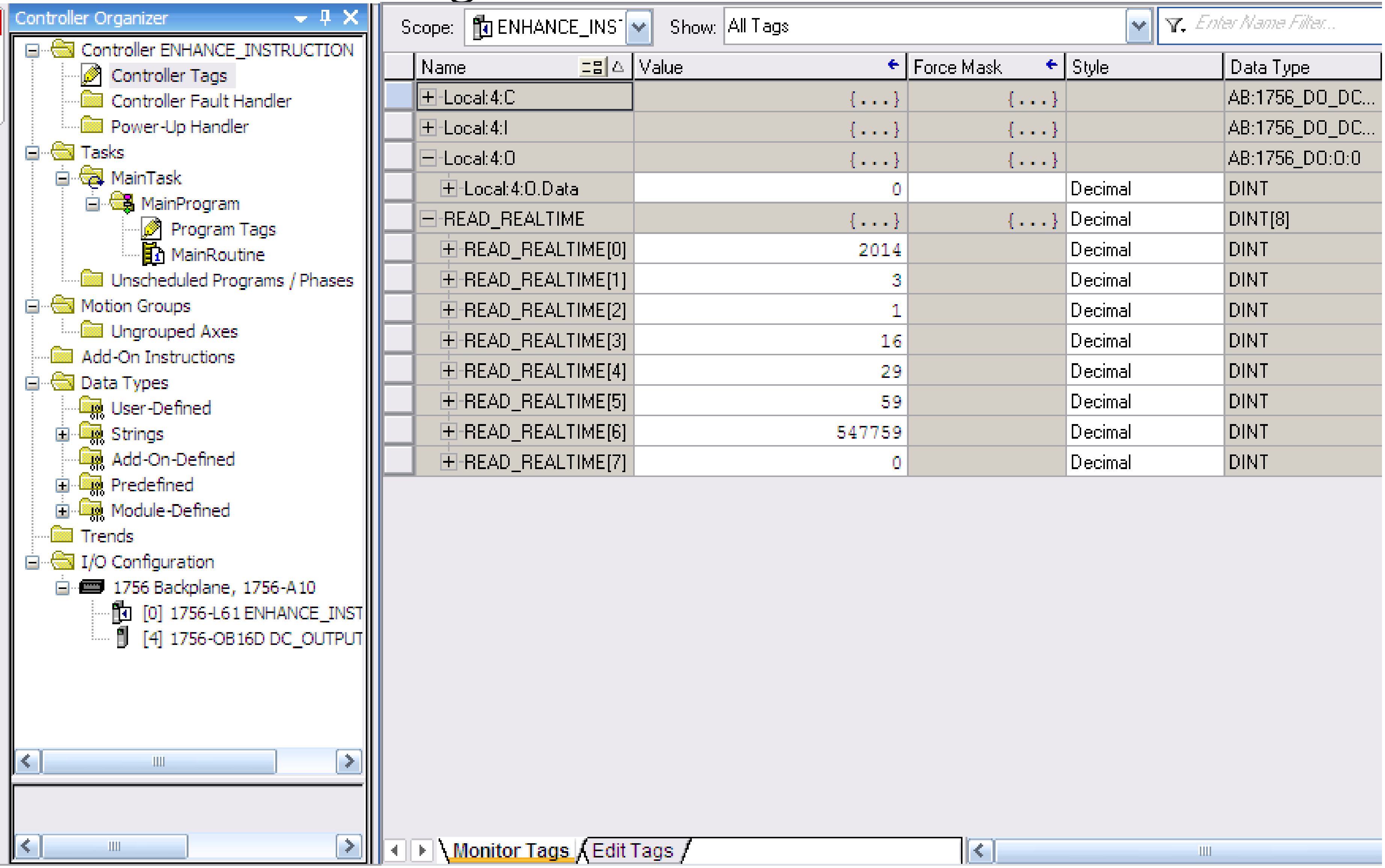
Path: AB_ETHIP-1\192.168.1.72\Backplane\0*

GSV
Get System Value
Class Name WallClockTime
Instance Name
Attribute Name DateTime
Dest READ_REALTIME[0]
2014

If DateTime data is wrong, use SSV to set DateTime to PLC

MSG, GSV, SSV INSTRUCTIONS

Choose Monitor Tags to view DateTime data of the controller



Controller Organizer

- Controller ENHANCE_INSTRUCTION
  - Controller Tags
  - Controller Fault Handler
  - Power-Up Handler
- Tasks
  - MainTask
    - MainProgram
      - Program Tags
      - MainRoutine
  - Unscheduled Programs / Phases
- Motion Groups
  - Ungrouped Axes
- Add-On Instructions
- Data Types
  - User-Defined
  - Strings
  - Add-On-Defined
  - Predefined
  - Module-Defined
- Trends
- I/O Configuration
  - 1756 Backplane, 1756-A10
    - [0] 1756-L61 ENHANCE_INST
    - [4] 1756-OB16D DC_OUTPUT

Scope: ENHANCE_INS' Show: All Tags Enter Name Filter...

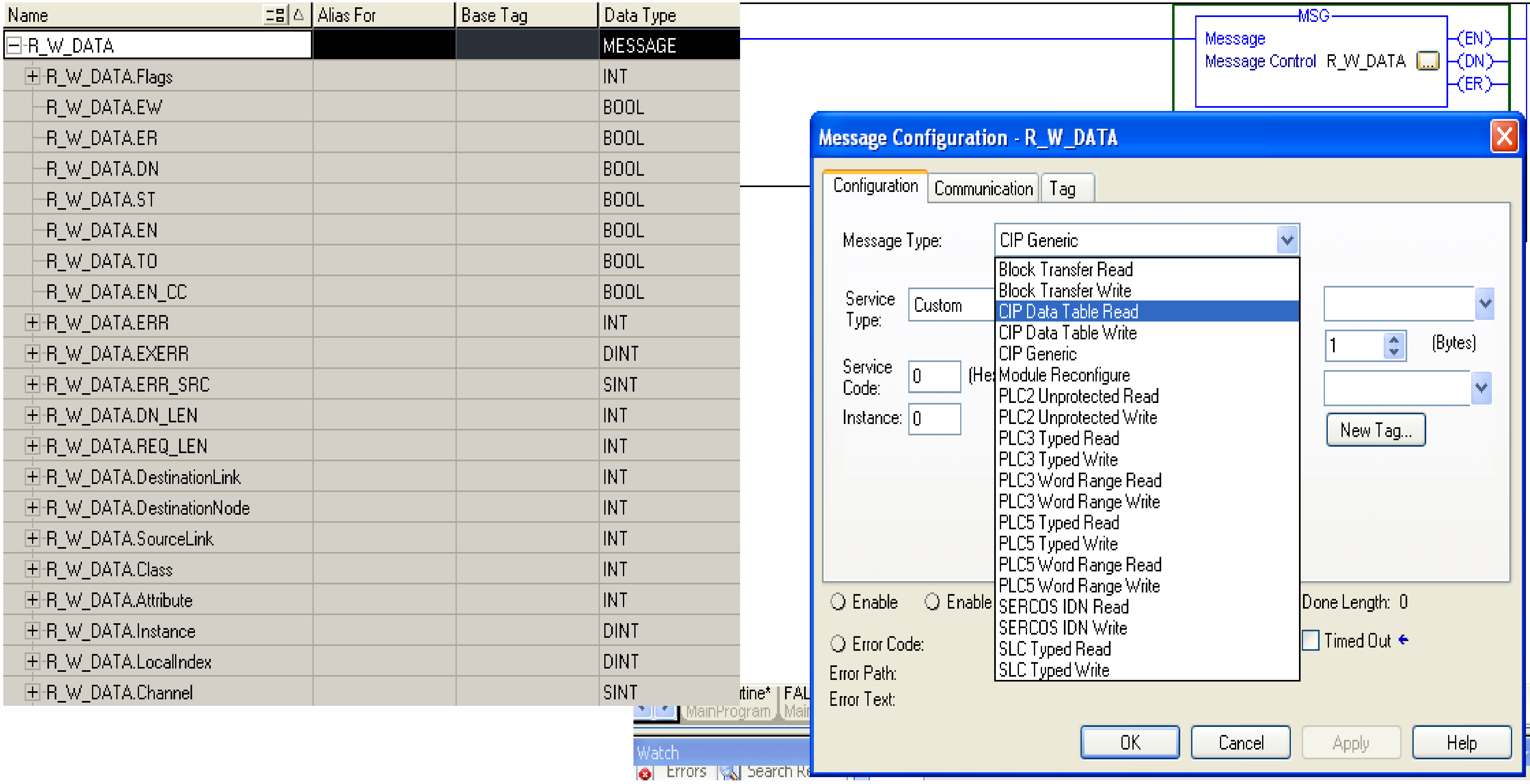
Name	Value	Force Mask	Style	Data Type
+ Local:4:C	{...}	{...}		AB:1756_DO_DC...
+ Local:4:I	{...}	{...}		AB:1756_DO_DC...
− Local:4:O	{...}	{...}		AB:1756_DO:O:0
+ Local:4:O.Data	0		Decimal	DINT
− READ_REALTIME	{...}	{...}	Decimal	DINT[8]
+ READ_REALTIME[0]	2014		Decimal	DINT
+ READ_REALTIME[1]	3		Decimal	DINT
+ READ_REALTIME[2]	1		Decimal	DINT
+ READ_REALTIME[3]	16		Decimal	DINT
+ READ_REALTIME[4]	29		Decimal	DINT
+ READ_REALTIME[5]	59		Decimal	DINT
+ READ_REALTIME[6]	547759		Decimal	DINT
+ READ_REALTIME[7]	0		Decimal	DINT

Monitor Tags / Edit Tags

MSG, GSV, SSV INSTRUCTIONS

Message Control (MSG)

Read or write data to or from the controller or a block of data to or from another module on another network.



Name	Alias For	Base Tag	Data Type
⊟ R_W_DATA			MESSAGE
⊞ R_W_DATA.Flags			INT
R_W_DATA.EW			BOOL
R_W_DATA.ER			BOOL
R_W_DATA.DN			BOOL
R_W_DATA.ST			BOOL
R_W_DATA.EN			BOOL
R_W_DATA.TO			BOOL
R_W_DATA.EN_CC			BOOL
⊞ R_W_DATA.ERR			INT
⊞ R_W_DATA.EXERR			DINT
⊞ R_W_DATA.ERR_SRC			SINT
⊞ R_W_DATA.DN_LEN			INT
⊞ R_W_DATA.REQ_LEN			INT
⊞ R_W_DATA.DestinationLink			INT
⊞ R_W_DATA.DestinationNode			INT
⊞ R_W_DATA.SourceLink			INT
⊞ R_W_DATA.Class			INT
⊞ R_W_DATA.Attribute			INT
⊞ R_W_DATA.Instance			DINT
⊞ R_W_DATA.LocalIndex			DINT
⊞ R_W_DATA.Channel			SINT

MSG
Message
Message Control R_W_DATA
(EN)
(DN)
(ER)

Message Configuration - R_W_DATA

Configuration | Communication | Tag

Message Type: CIP Generic

- Block Transfer Read
- Block Transfer Write
- CIP Data Table Read
- CIP Data Table Write
- CIP Generic
- Module Reconfigure
- PLC2 Unprotected Read
- PLC2 Unprotected Write
- PLC3 Typed Read
- PLC3 Typed Write
- PLC3 Word Range Read
- PLC3 Word Range Write
- PLC5 Typed Read
- PLC5 Typed Write
- PLC5 Word Range Read
- PLC5 Word Range Write
- SERCOS IDN Read
- SERCOS IDN Write
- SLC Typed Read
- SLC Typed Write

Service Type: Custom

Service Code: 0 (He

Instance: 0

1 (Bytes)

New Tag...

○ Enable ○ Enable

Done Length: 0

○ Error Code:

□ Timed Out

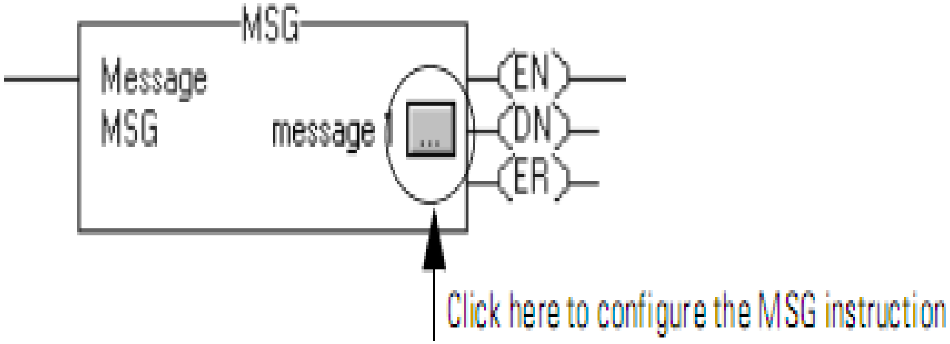
Error Path:

Error Text:

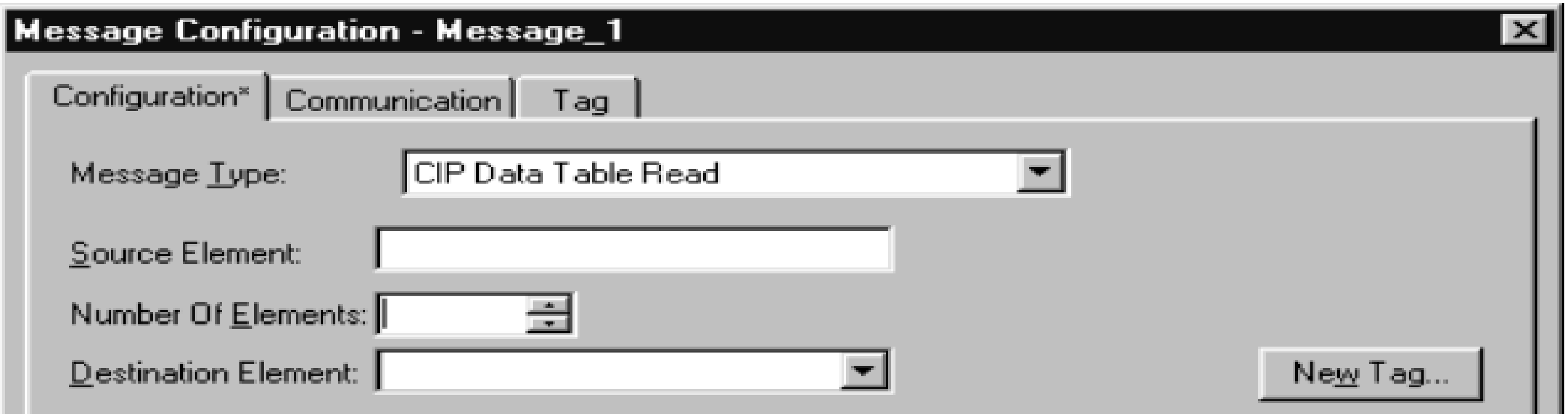
MSG, GSV, SSV INSTRUCTIONS

Message Control (MSG)

- Message configuration



Click here to configure the MSG instruction



Message Type: CIP Data Table Read

Source Element:

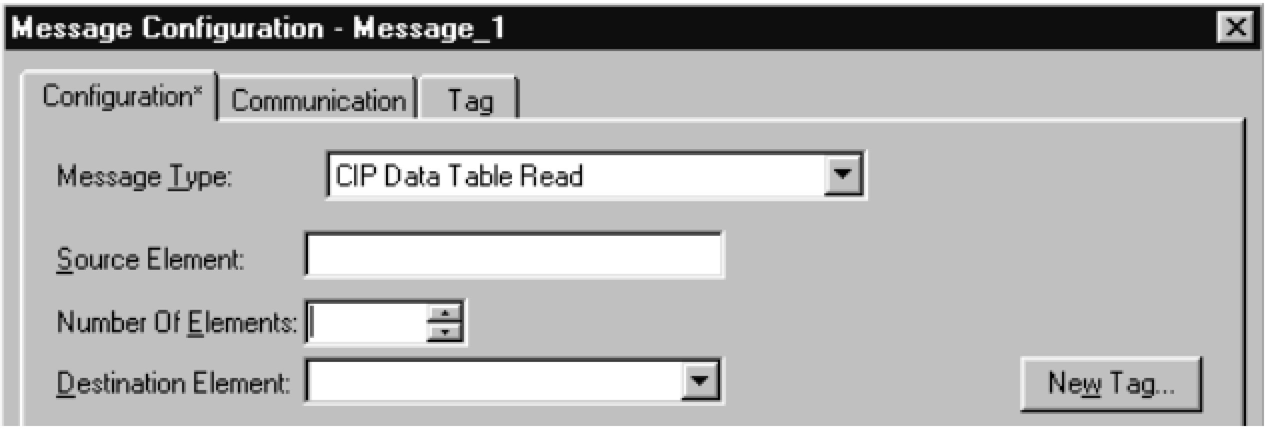
Number Of Elements:

Destination Element:

If The Target Device Is a	Select One Of These Message Types
Logix5000 controller	CIP Data Table Read
	CIP Data Table Write
I/O module that you configure using RSLogix 5000 software	Module Reconfigure
	CIP Generic

ENHANCE INSTRUCTIONS

Message Control (MSG): Message configuration



Message Configuration - Message_1

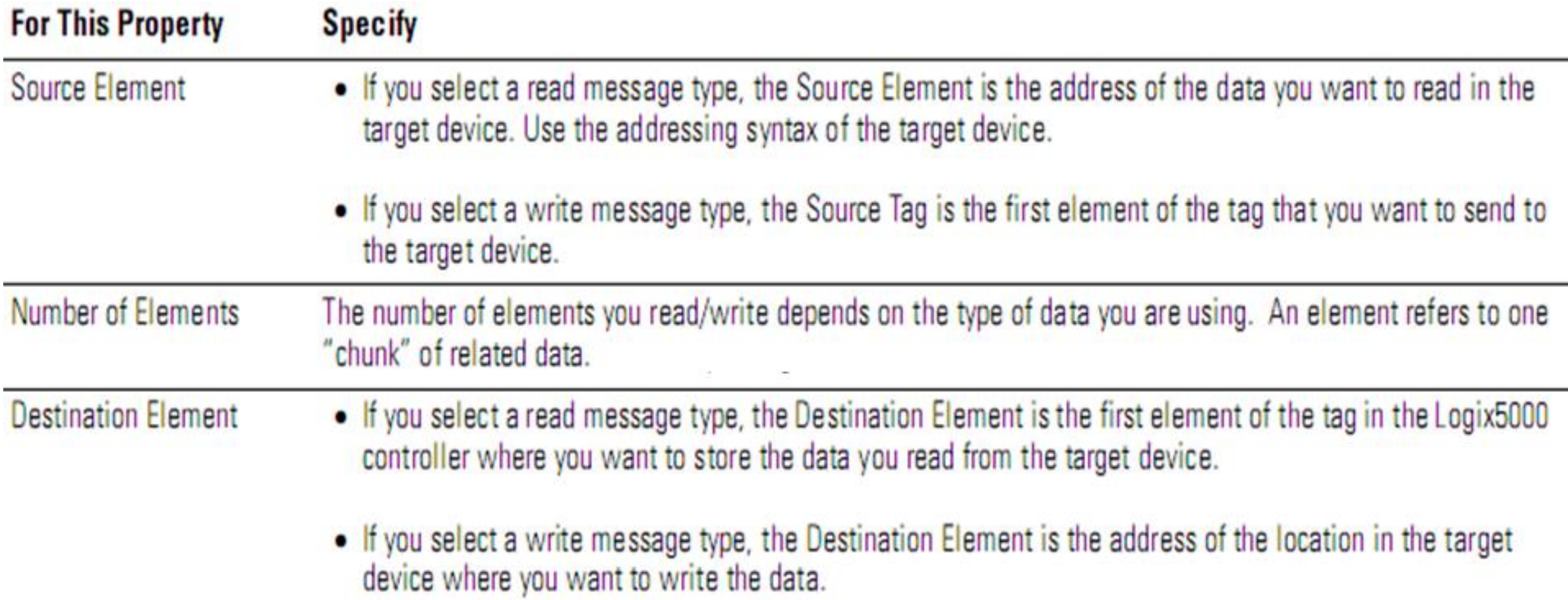
Configuration* | Communication | Tag

Message Type: CIP Data Table Read

Source Element:

Number Of Elements:

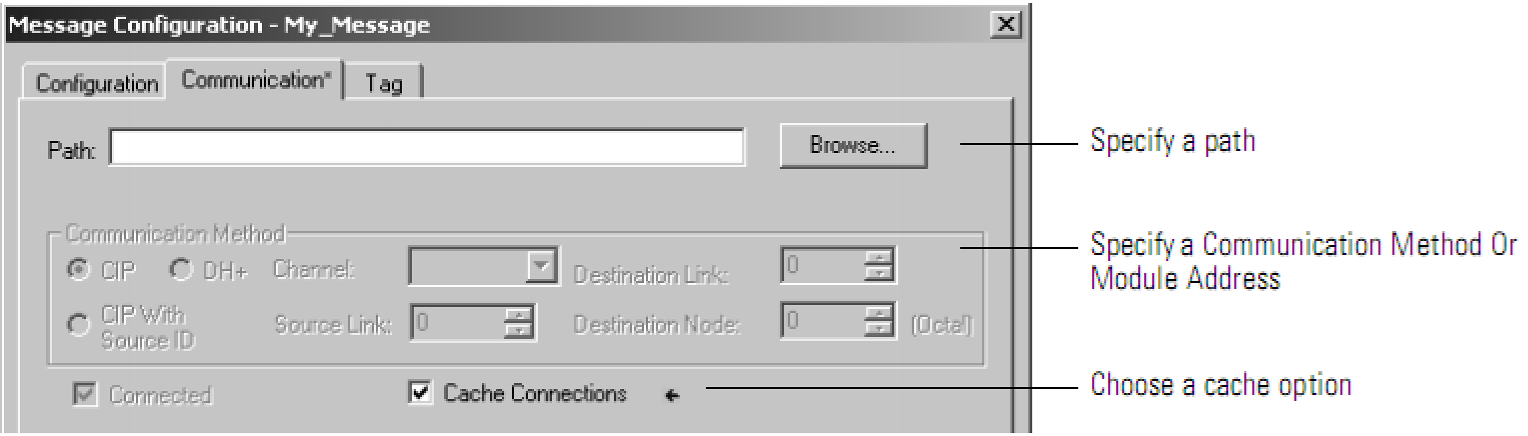
Destination Element:



For This Property	Specify
Source Element	• If you select a read message type, the Source Element is the address of the data you want to read in the target device. Use the addressing syntax of the target device. • If you select a write message type, the Source Tag is the first element of the tag that you want to send to the target device.
Number of Elements	The number of elements you read/write depends on the type of data you are using. An element refers to one "chunk" of related data.
Destination Element	• If you select a read message type, the Destination Element is the first element of the tag in the Logix5000 controller where you want to store the data you read from the target device. • If you select a write message type, the Destination Element is the address of the location in the target device where you want to write the data.

MSG, GSV, SSV INSTRUCTIONS

Message Control (MSG): Message configuration



Path:

● CIP ○ DH+ Channel:

Destination Link: 0

○ CIP With Source ID Source Link: 0

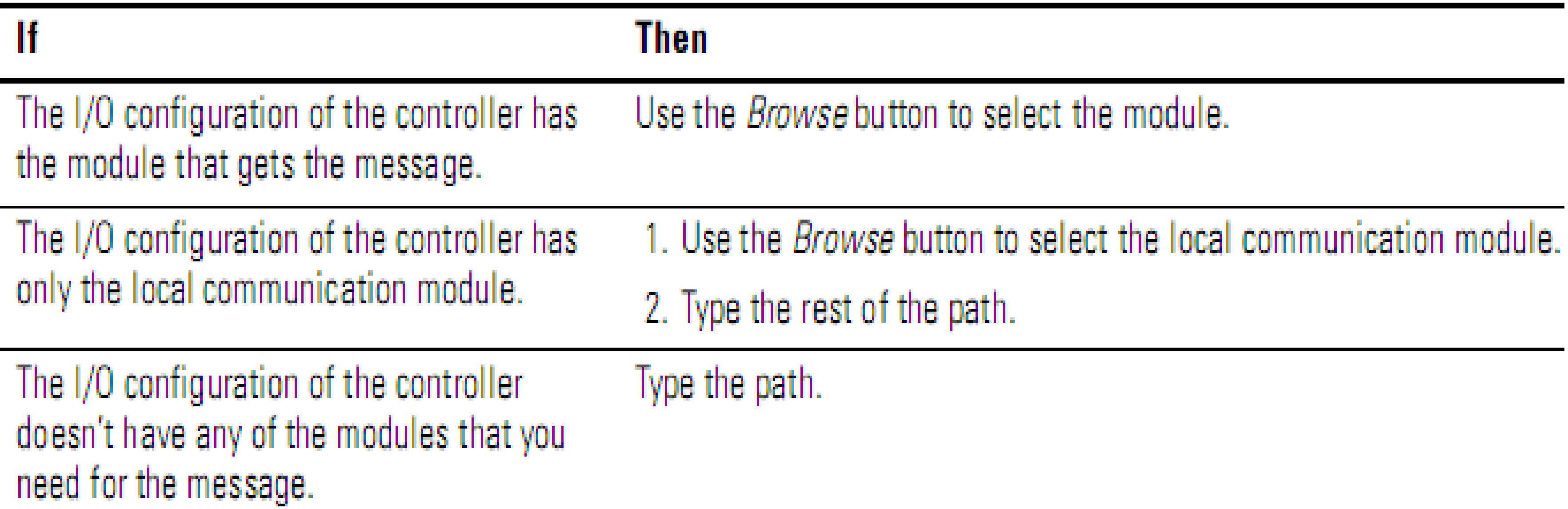
Destination Node: 0 (Octal)

☑ Connected ☑ Cache Connections

Specify a path

Specify a Communication Method Or Module Address

Choose a cache option



If	Then
The I/O configuration of the controller has the module that gets the message.	Use the *Browse* button to select the module.
The I/O configuration of the controller has only the local communication module.	1. Use the *Browse* button to select the local communication module. 2. Type the rest of the path.
The I/O configuration of the controller doesn't have any of the modules that you need for the message.	Type the path.

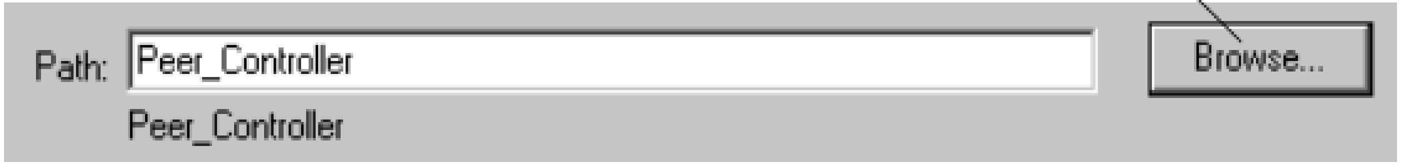
MSG, GSV, SSV INSTRUCTIONS

Message Control (MSG): Message configuration

Example

The I/O configuration of the controller has the module that gets the message.

Click the Browse button and select the module.



Path: Peer_Controller

Peer_Controller

The I/O configuration of the controller has only the local communication module.

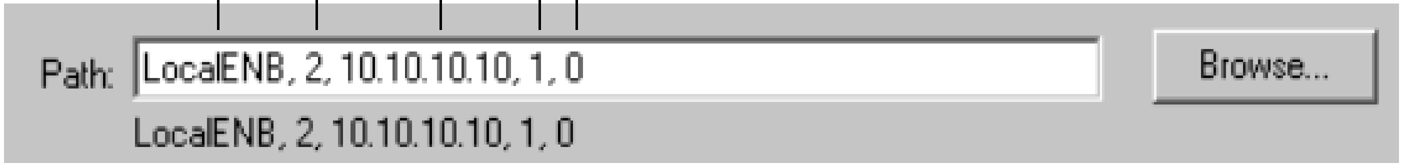
Go to the local communication module.

Go out the EtherNet/IP port....

to the address of 10.10.10.10.

Go across the backplane...

to the module in slot 0.



Path: LocalENB, 2, 10.10.10.10, 1, 0

LocalENB, 2, 10.10.10.10, 1, 0

MSG, GSV, SSV INSTRUCTIONS

Message Control (MSG) Example

Send data from Master_CPU (Slot 0) to Peer CPU(Slot 5) or vice versa

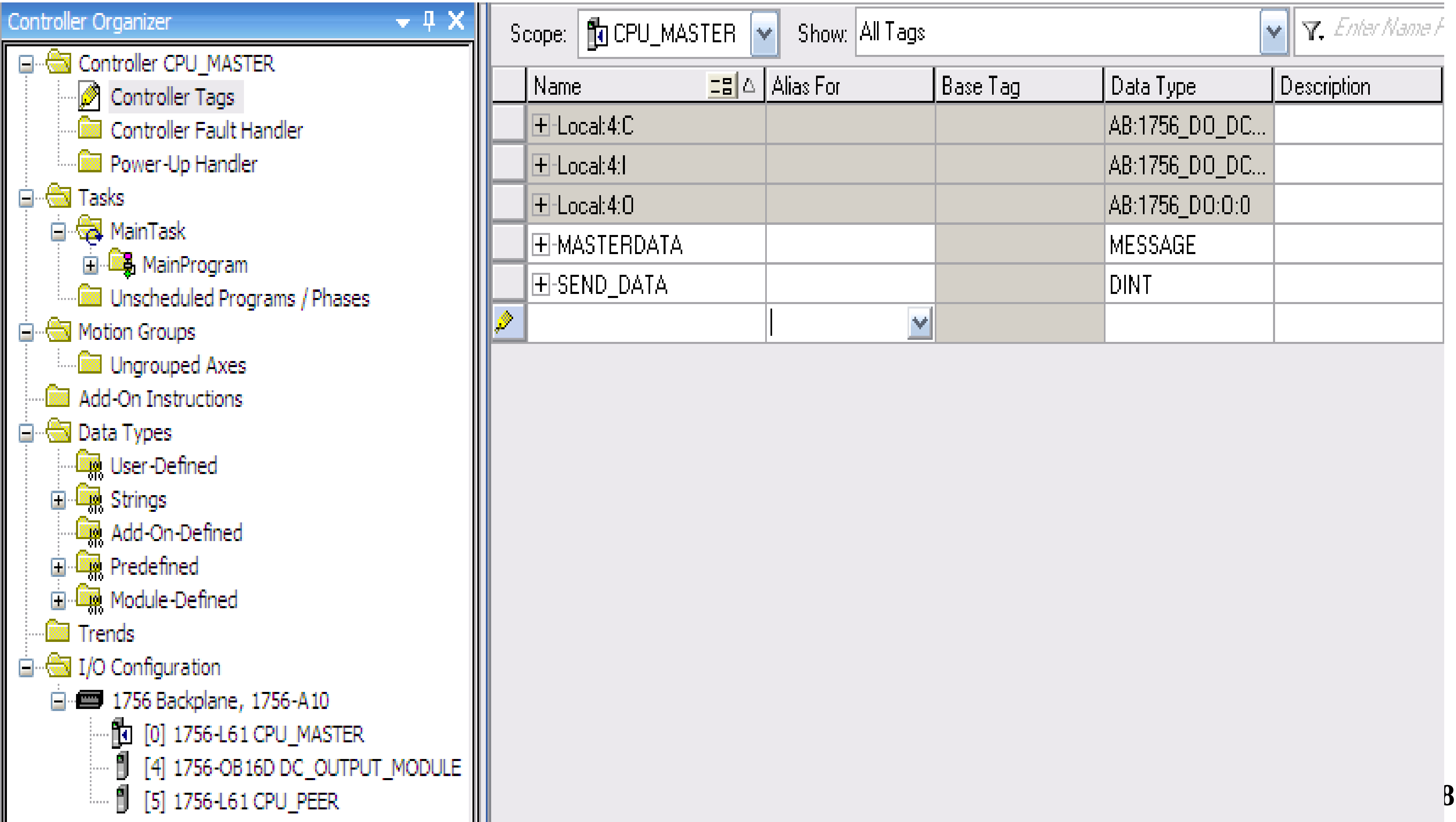
- ➢Create a project with two CPUs and a Send_Data tag in controller tag
- ➢Create another project with two CPUs and Read_Data tag in controller tag
- ➢Use MSG instruction to send or read data from Master_CPU to PEER_CPU or vice versa

All tags are created in controller tag

MSG, GSV, SSV INSTRUCTIONS

Message Control (MSG) Example

➢Create a project with two CPUs and download to CPU_Master



Controller Organizer

- Controller CPU_MASTER
  - Controller Tags
  - Controller Fault Handler
  - Power-Up Handler
- Tasks
  - MainTask
    - MainProgram
  - Unscheduled Programs / Phases
- Motion Groups
  - Ungrouped Axes
- Add-On Instructions
- Data Types
  - User-Defined
  - Strings
  - Add-On-Defined
  - Predefined
  - Module-Defined
- Trends
- I/O Configuration
  - 1756 Backplane, 1756-A10
    - [0] 1756-L61 CPU_MASTER
    - [4] 1756-OB16D DC_OUTPUT_MODULE
    - [5] 1756-L61 CPU_PEER

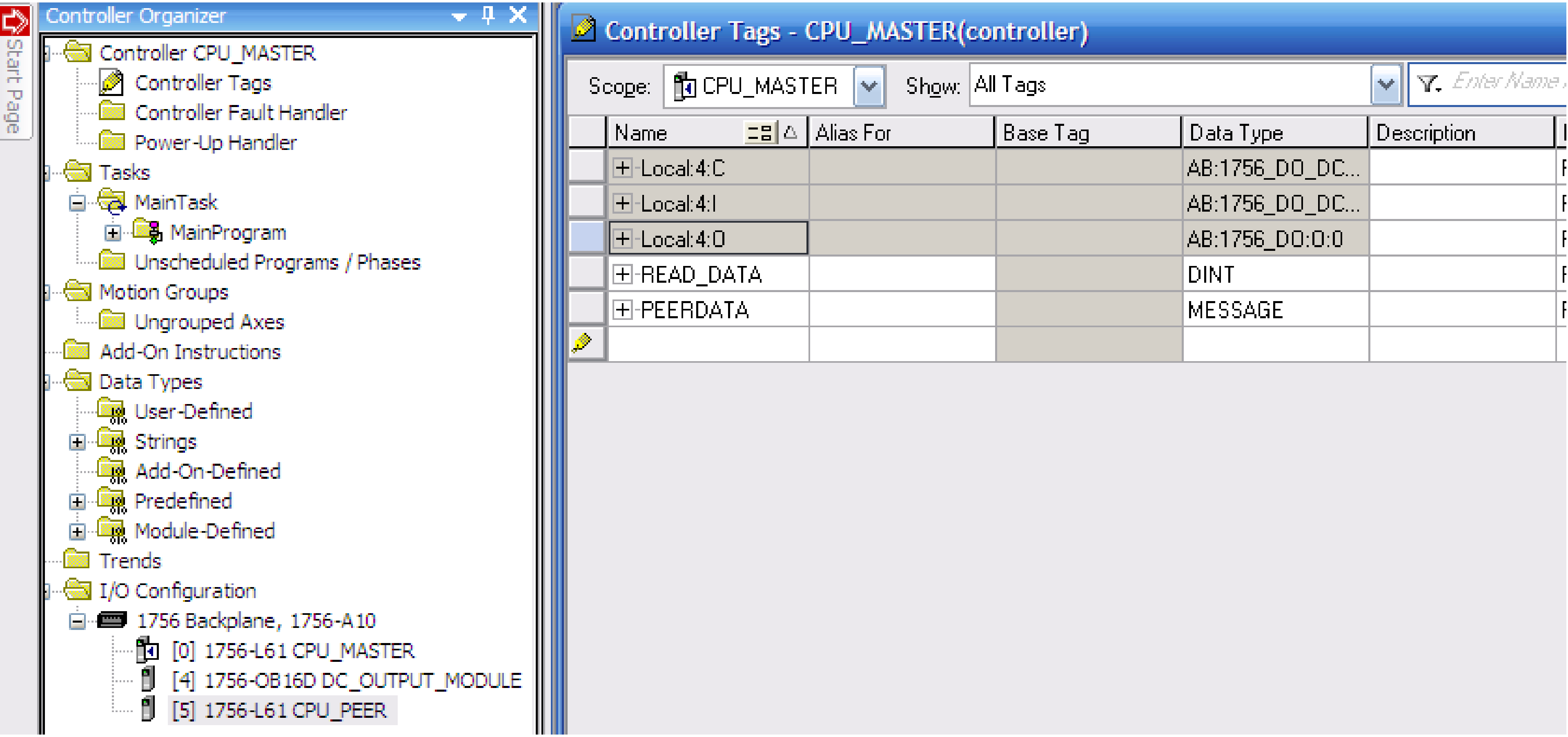
Scope: CPU_MASTER Show: All Tags Enter Name F

Name	Alias For	Base Tag	Data Type	Description
+ Local:4:C			AB:1756_DO_DC...	
+ Local:4:I			AB:1756_DO_DC...	
+ Local:4:O			AB:1756_DO:O:0	
+ MASTERDATA			MESSAGE	
+ SEND_DATA			DINT	

MSG, GSV, SSV INSTRUCTIONS

Message Control (MSG) Example

➢Create an another project with two CPUs and download to CPU_Peer



Controller Tags - CPU_MASTER(controller)

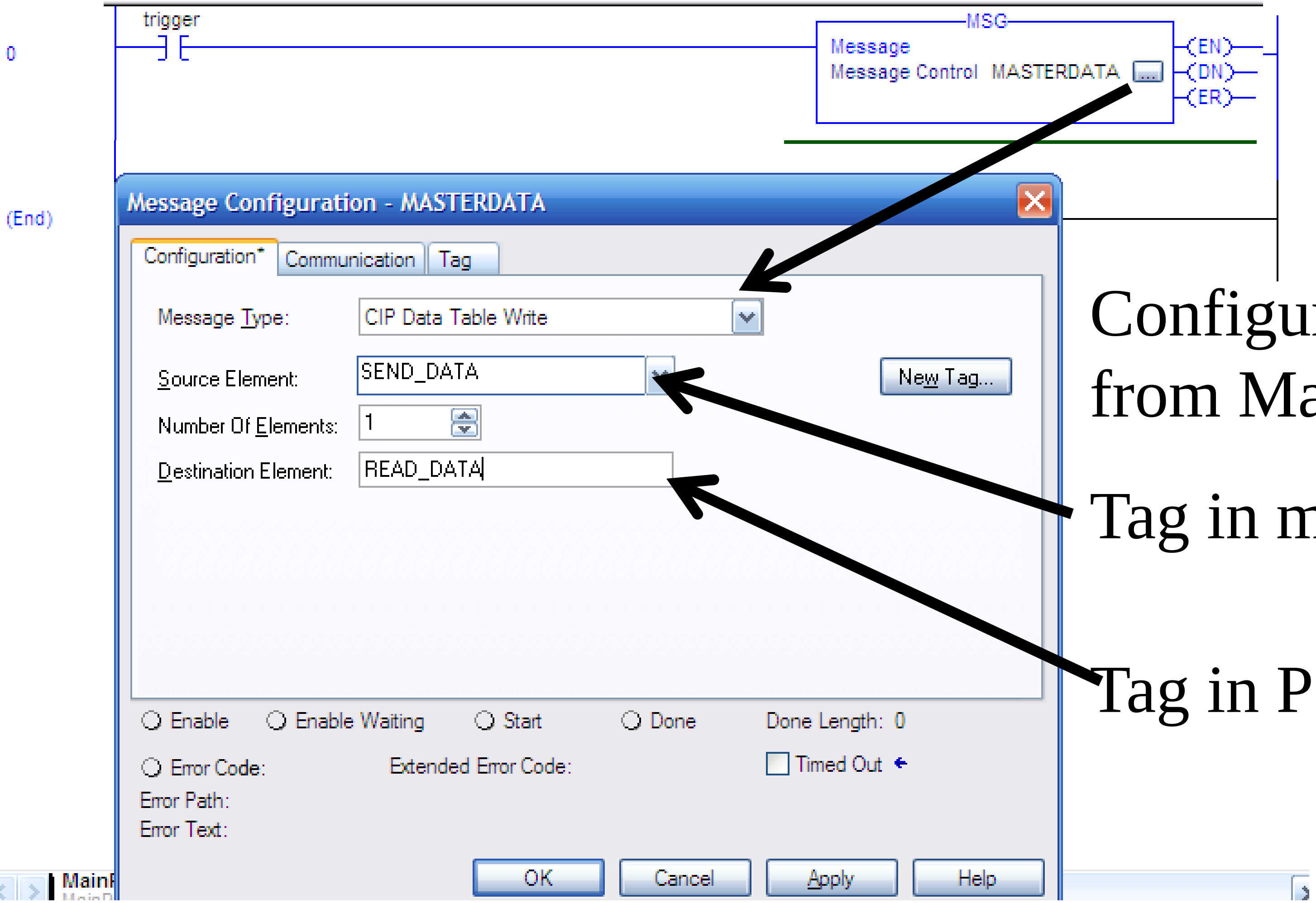
Scope: CPU_MASTER Show: All Tags

Name	Alias For	Base Tag	Data Type	Description
+ Local:4:C			AB:1756_DO_DC...	
+ Local:4:I			AB:1756_DO_DC...	
+ Local:4:O			AB:1756_DO:O:0	
+ READ_DATA			DINT	
+ PEERDATA			MESSAGE	

MSG, GSV, SSV INSTRUCTIONS

Message Control (MSG) Example

➢Use MSG instruction to write or read Data from Master to Peer or vice versa



Message Configuration - MASTERDATA

Message Type: CIP Data Table Write

Source Element: SEND_DATA

Number Of Elements: 1

Destination Element: READ_DATA

○ Enable ○ Enable Waiting ○ Start ○ Done Done Length: 0

○ Error Code: Extended Error Code: □ Timed Out

Error Path:

Error Text:

Configure to write data from Master to Peer

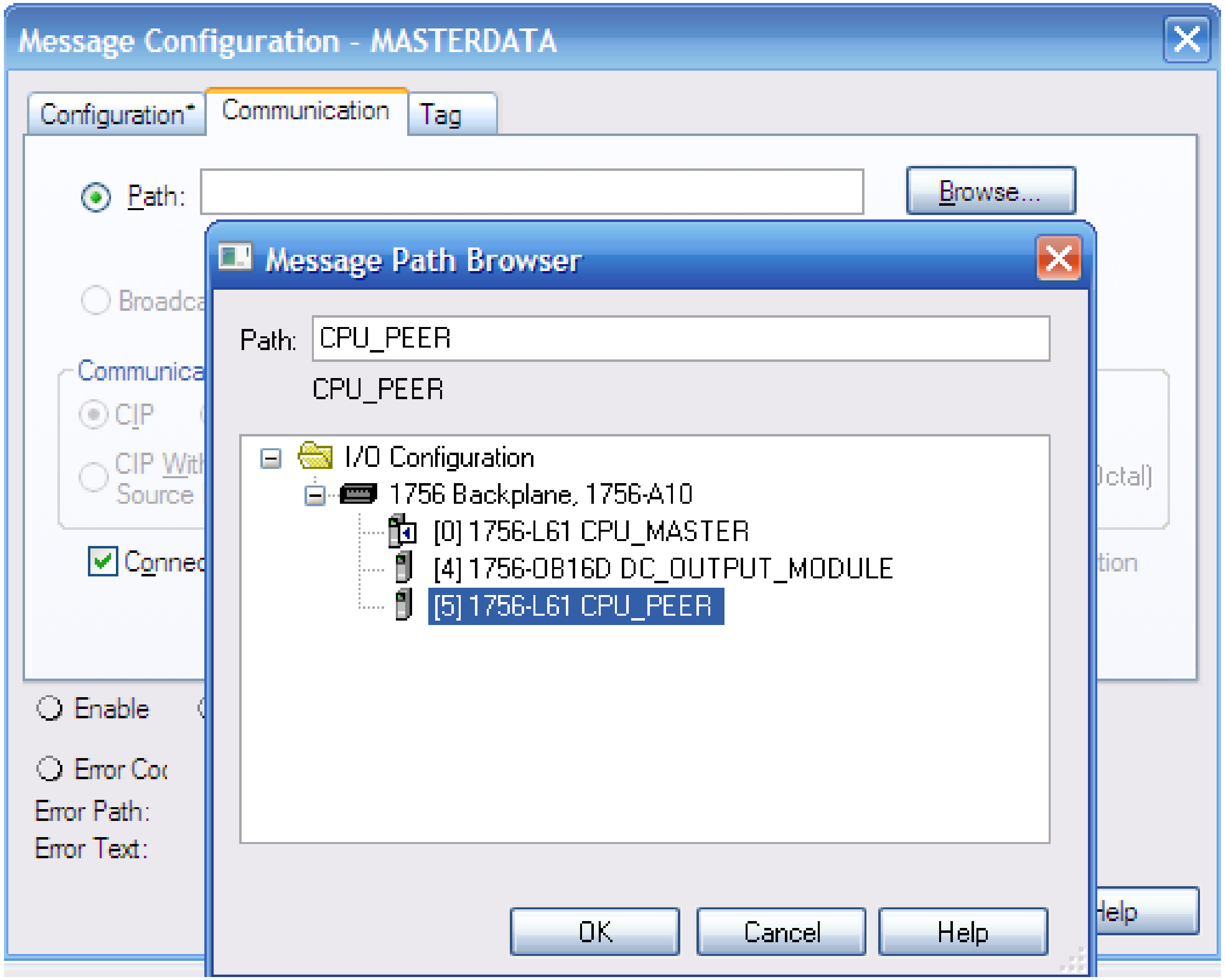
Tag in master

Tag in Peer

MSG, GSV, SSV INSTRUCTIONS

Message Control (MSG) Example

- Slect path to transfer data



Message Configuration - MASTERDATA

Configuration* | Communication | Tag

● Path:

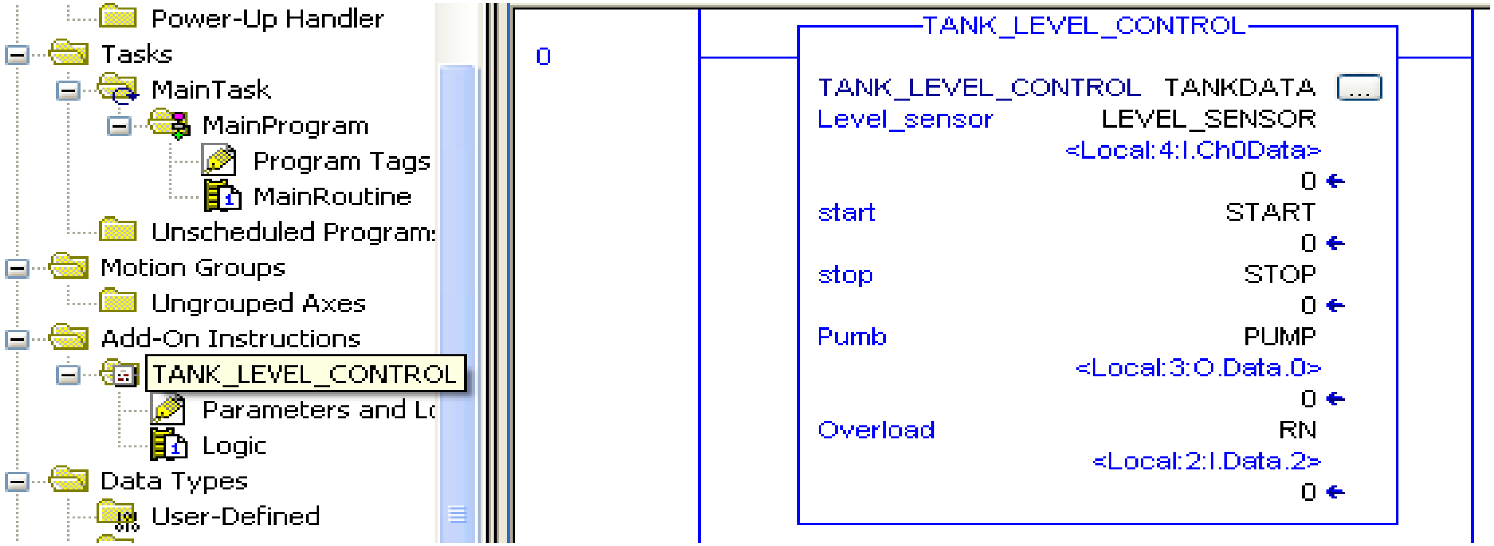
Message Path Browser

Path: CPU_PEER

CPU_PEER

```
⊟ I/O Configuration
   ⊟ 1756 Backplane, 1756-A10
        [0] 1756-L61 CPU_MASTER
        [4] 1756-OB16D DC_OUTPUT_MODULE
        [5] 1756-L61 CPU_PEER
```

ADDON INSTRUCTION



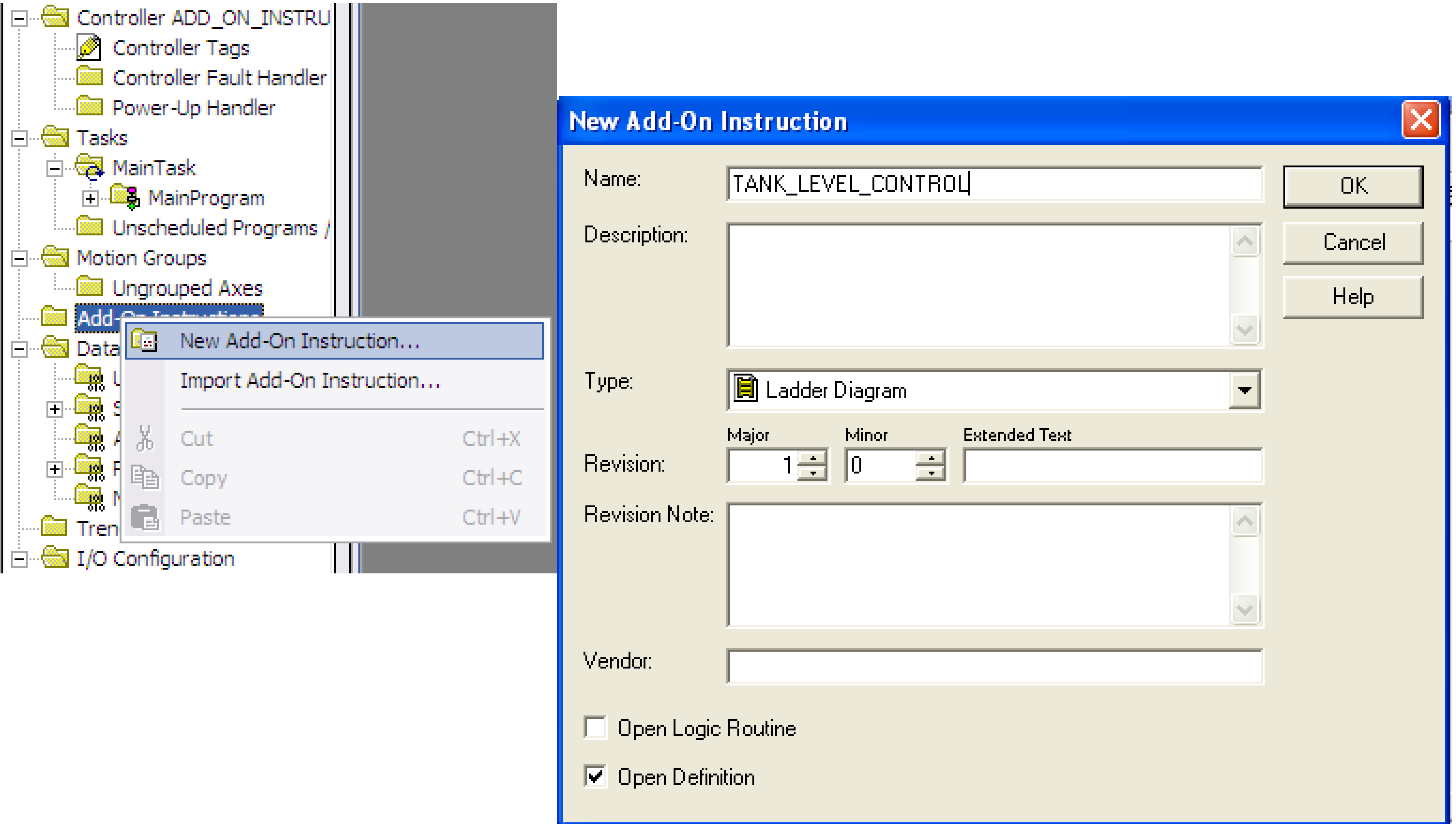
```
TANK_LEVEL_CONTROL
TANK_LEVEL_CONTROL  TANKDATA  ...
Level_sensor        LEVEL_SENSOR
              <Local:4:I.Ch0Data>
                                0 ←
start                      START
                                0 ←
stop                        STOP
                                0 ←
Pumb                        PUMP
              <Local:3:O.Data.0>
                                0 ←
Overload                      RN
              <Local:2:I.Data.2>
                                0 ←
```

Add-on Instruction introduction

- Custom Instruction
- Reuse code
- Provide an easier to understand interface
- Export and Import an Add-on Instruction

ADDON INSTRUCTION

Creating Add-on Instruction



New Add-On Instruction

Name: TANK_LEVEL_CONTROL

Description:

Type: Ladder Diagram

Revision: Major 1 Minor 0 Extended Text

Revision Note:

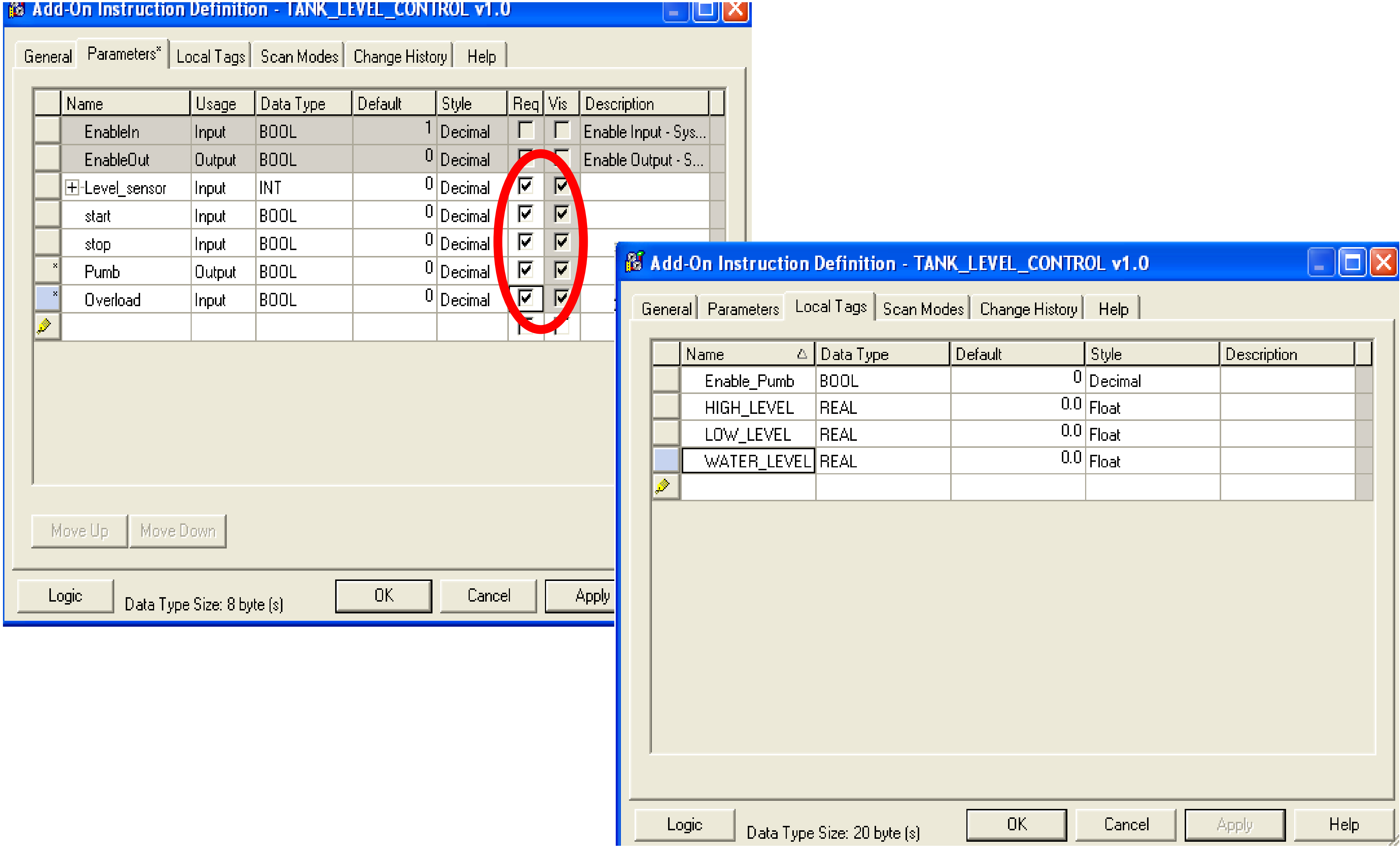
Vendor:

□ Open Logic Routine

☑ Open Definition

ADDON INSTRUCTION

Creating parameters and Local Tags



Add-On Instruction Definition - TANK_LEVEL_CONTROL v1.0

Name	Usage	Data Type	Default	Style	Req	Vis	Description
EnableIn	Input	BOOL	1	Decimal	□	□	Enable Input - Sys...
EnableOut	Output	BOOL	0	Decimal	□	□	Enable Output - S...
⊞ Level_sensor	Input	INT	0	Decimal	☑	☑	
start	Input	BOOL	0	Decimal	☑	☑	
stop	Input	BOOL	0	Decimal	☑	☑	
Pumb	Output	BOOL	0	Decimal	☑	☑	
Overload	Input	BOOL	0	Decimal	☑	☑	

Data Type Size: 8 byte (s)

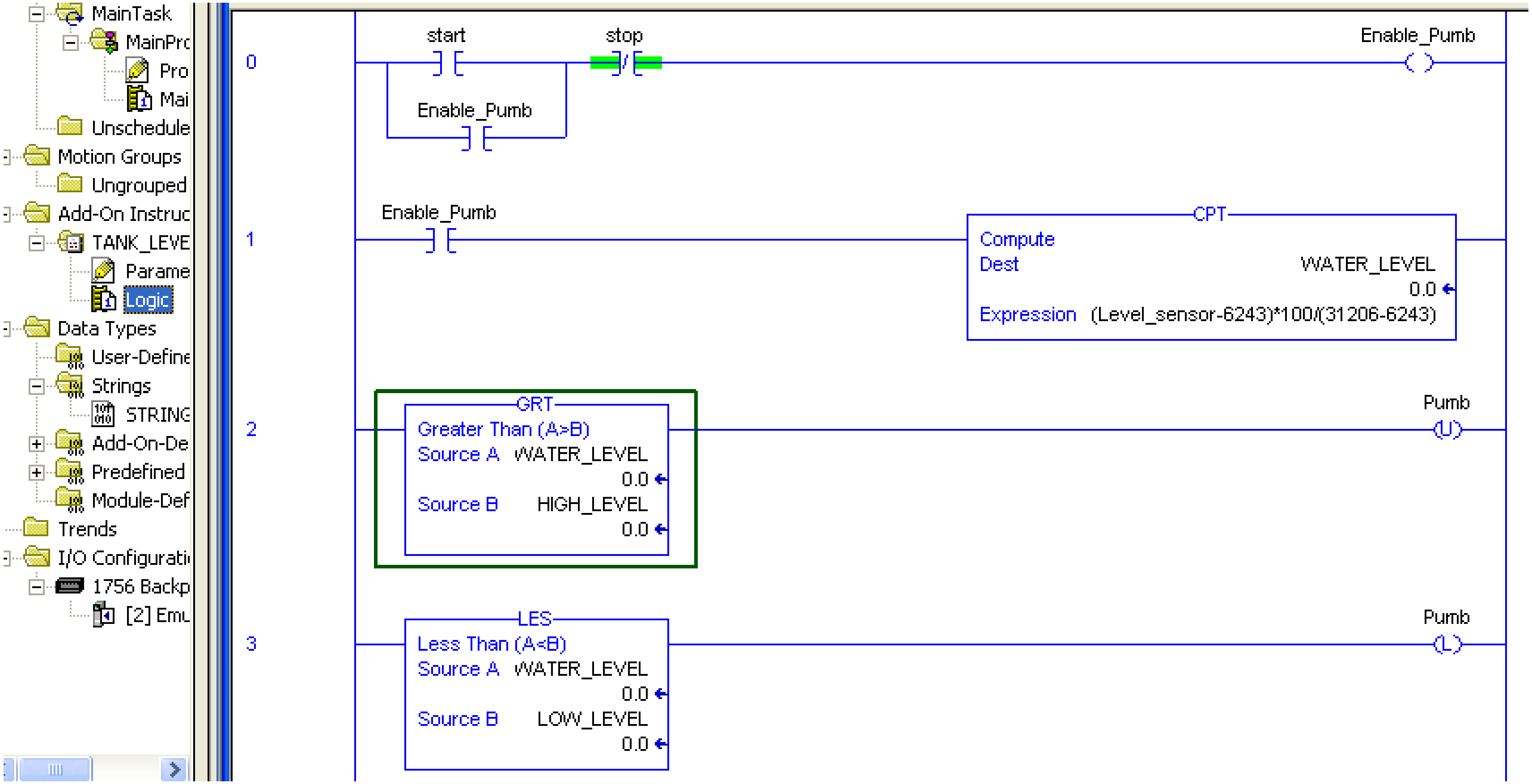
Add-On Instruction Definition - TANK_LEVEL_CONTROL v1.0

Name	Data Type	Default	Style	Description
Enable_Pumb	BOOL	0	Decimal	
HIGH_LEVEL	REAL	0.0	Float	
LOW_LEVEL	REAL	0.0	Float	
WATER_LEVEL	REAL	0.0	Float	

Data Type Size: 20 byte (s)

ADDON INSTRUCTION

Creating logic for the Add-on Instruction



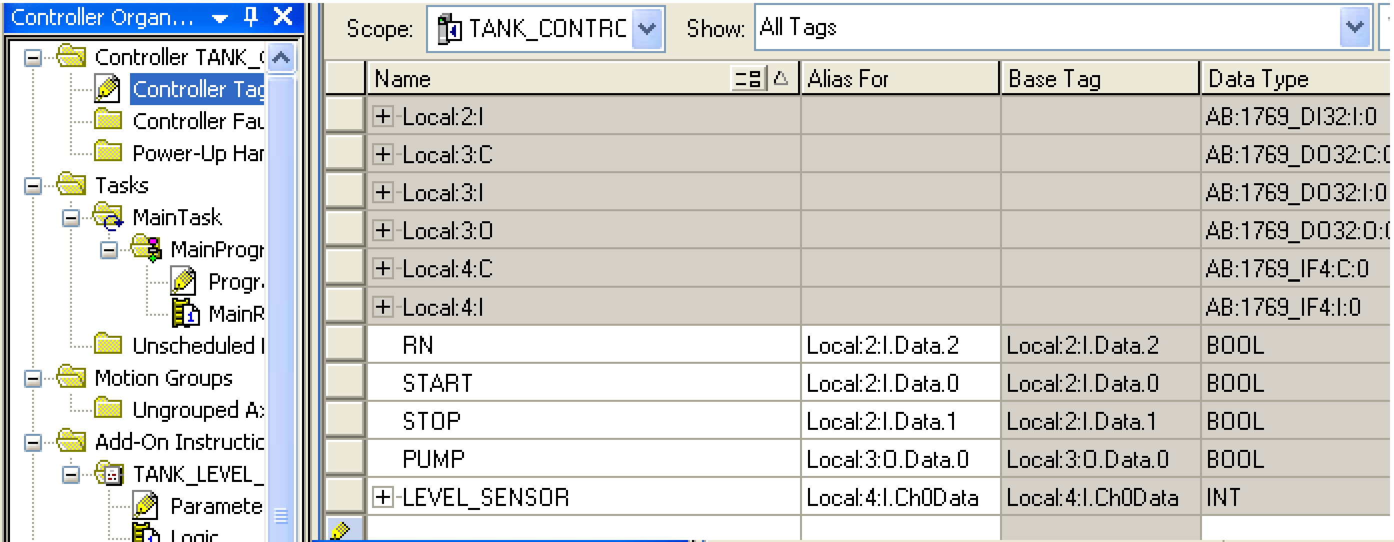
CPT
Compute
Dest WATER_LEVEL
0.0
Expression (Level_sensor-6243)*100/(31206-6243)

GRT
Greater Than (A>B)
Source A WATER_LEVEL
0.0
Source B HIGH_LEVEL
0.0

LES
Less Than (A<B)
Source A WATER_LEVEL
0.0
Source B LOW_LEVEL
0.0

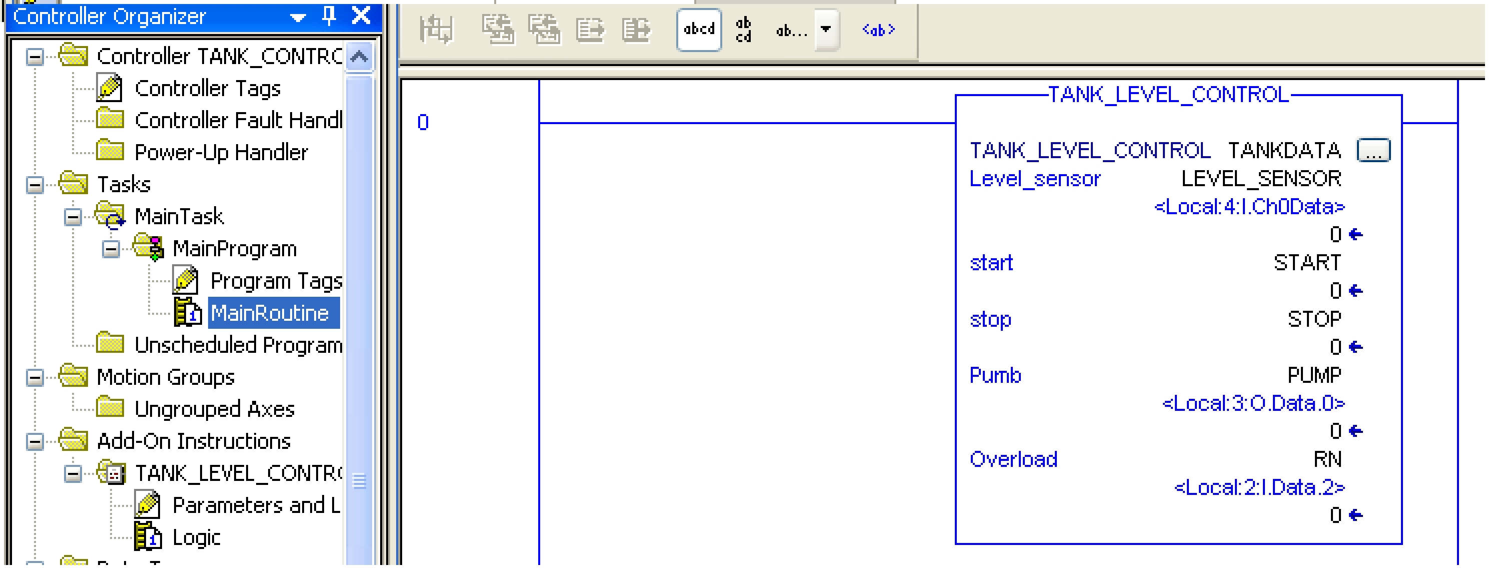
ADDON INSTRUCTION

Creating I/O Tags and Adding the Add_on instruction to project.



Scope: TANK_CONTRC Show: All Tags

Name	Alias For	Base Tag	Data Type
+ Local:2:I			AB:1769_DI32:I:0
+ Local:3:C			AB:1769_DO32:C:(
+ Local:3:I			AB:1769_DO32:I:0
+ Local:3:O			AB:1769_DO32:O:(
+ Local:4:C			AB:1769_IF4:C:0
+ Local:4:I			AB:1769_IF4:I:0
RN	Local:2:I.Data.2	Local:2:I.Data.2	BOOL
START	Local:2:I.Data.0	Local:2:I.Data.0	BOOL
STOP	Local:2:I.Data.1	Local:2:I.Data.1	BOOL
PUMP	Local:3:O.Data.0	Local:3:O.Data.0	BOOL
+ LEVEL_SENSOR	Local:4:I.Ch0Data	Local:4:I.Ch0Data	INT



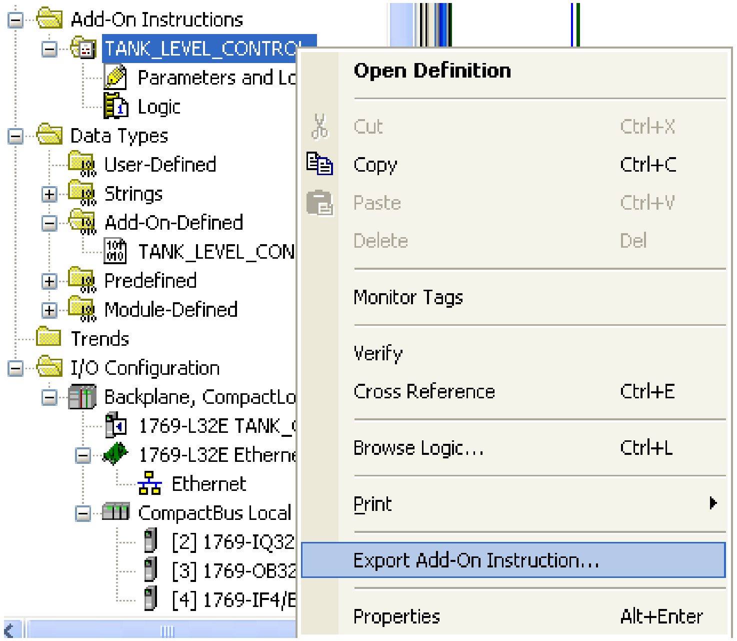
TANK_LEVEL_CONTROL

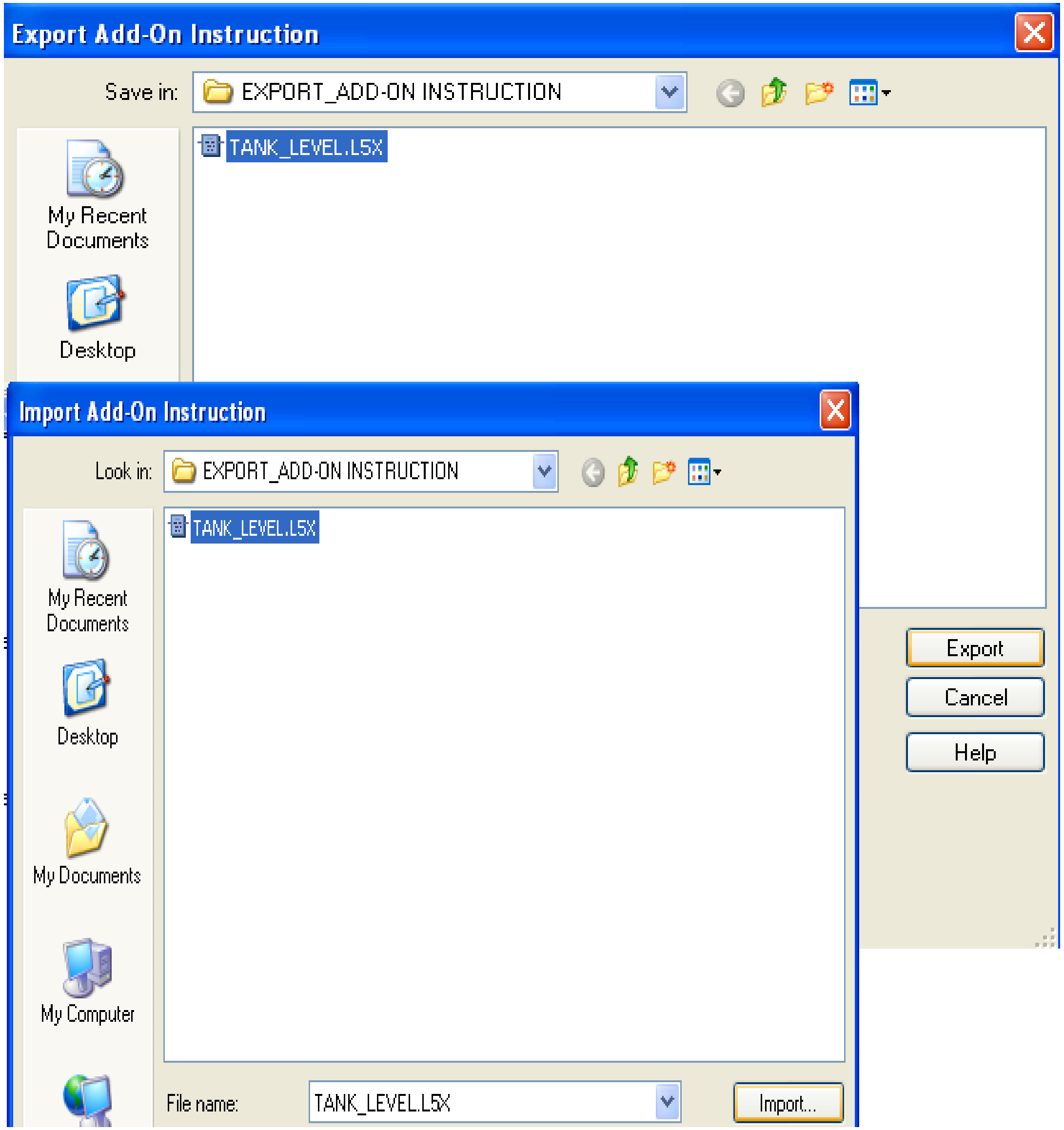
TANK_LEVEL_CONTROL	TANKDATA ...
Level_sensor	LEVEL_SENSOR <Local:4:I.Ch0Data> 0 ←
start	START 0 ←
stop	STOP 0 ←
Pumb	PUMP <Local:3:O.Data.0> 0 ←
Overload	RN <Local:2:I.Data.2> 0 ←

Participants program to control Tank Level using Add-on Instruction?

ADDON INSTRUCTION

Export and Import the Add-on Instruction





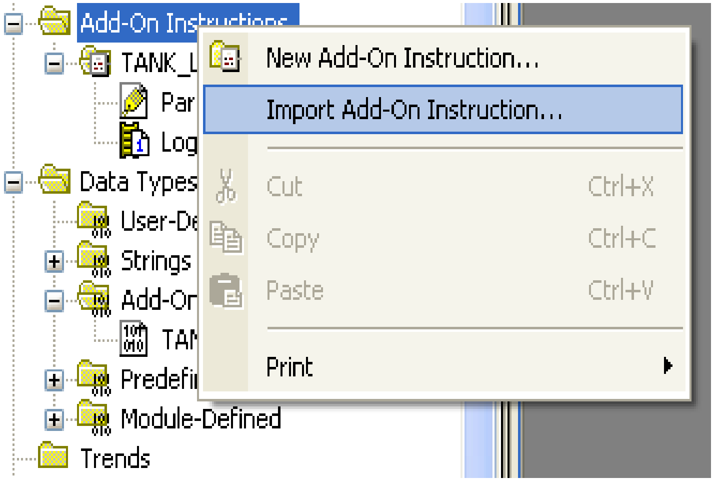
Save in: EXPORT_ADD-ON INSTRUCTION

TANK_LEVEL.L5X

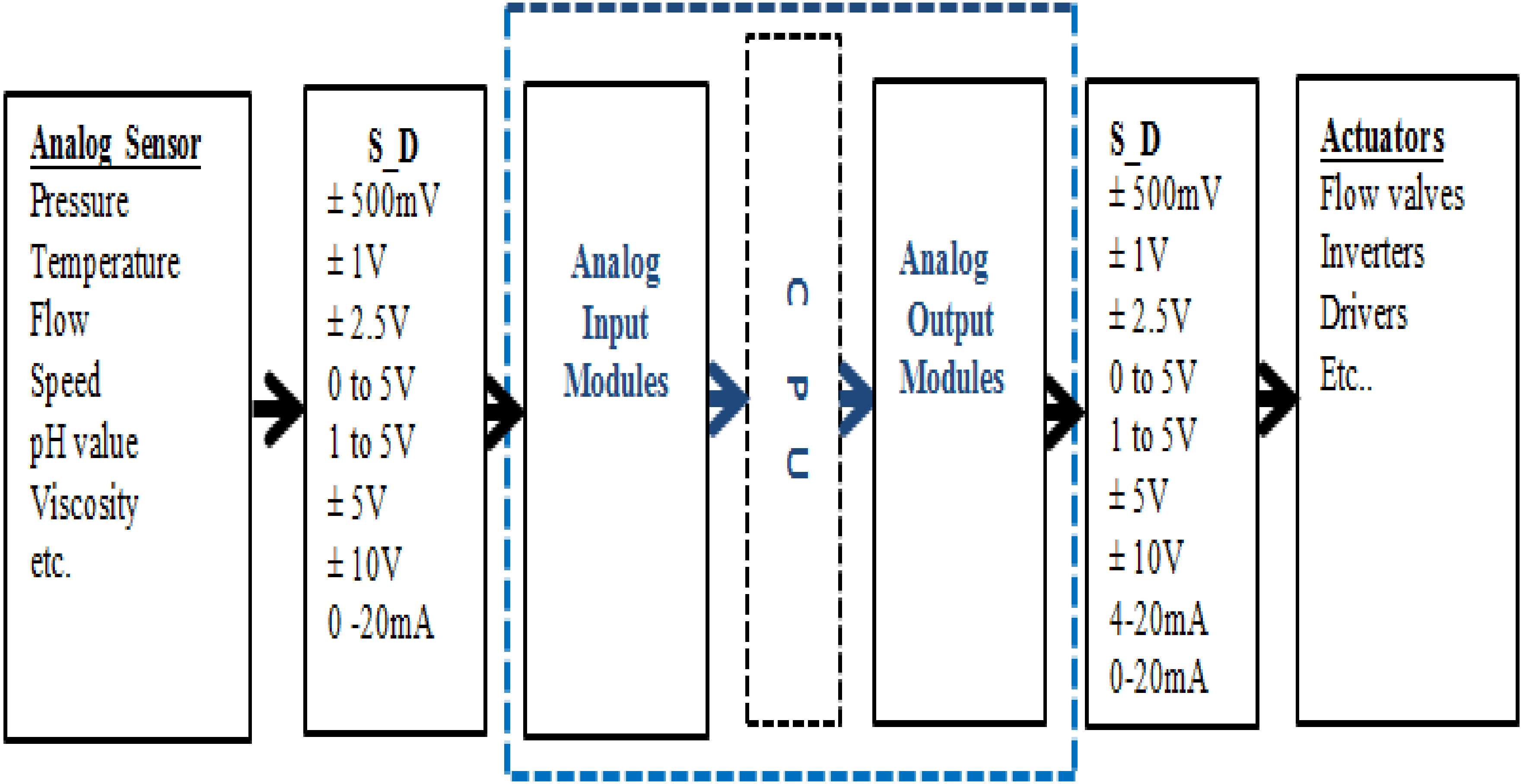
Look in: EXPORT_ADD-ON INSTRUCTION

TANK_LEVEL.L5X

File name: TANK_LEVEL.L5X



ANALOG MODULES



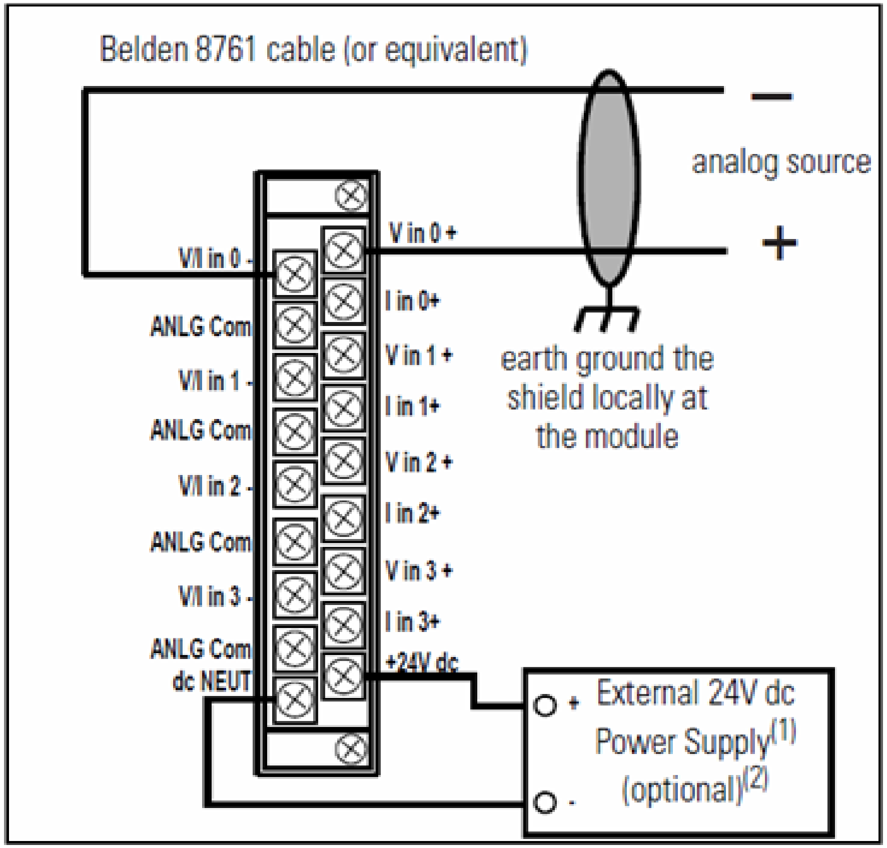
ANALOG MODULES

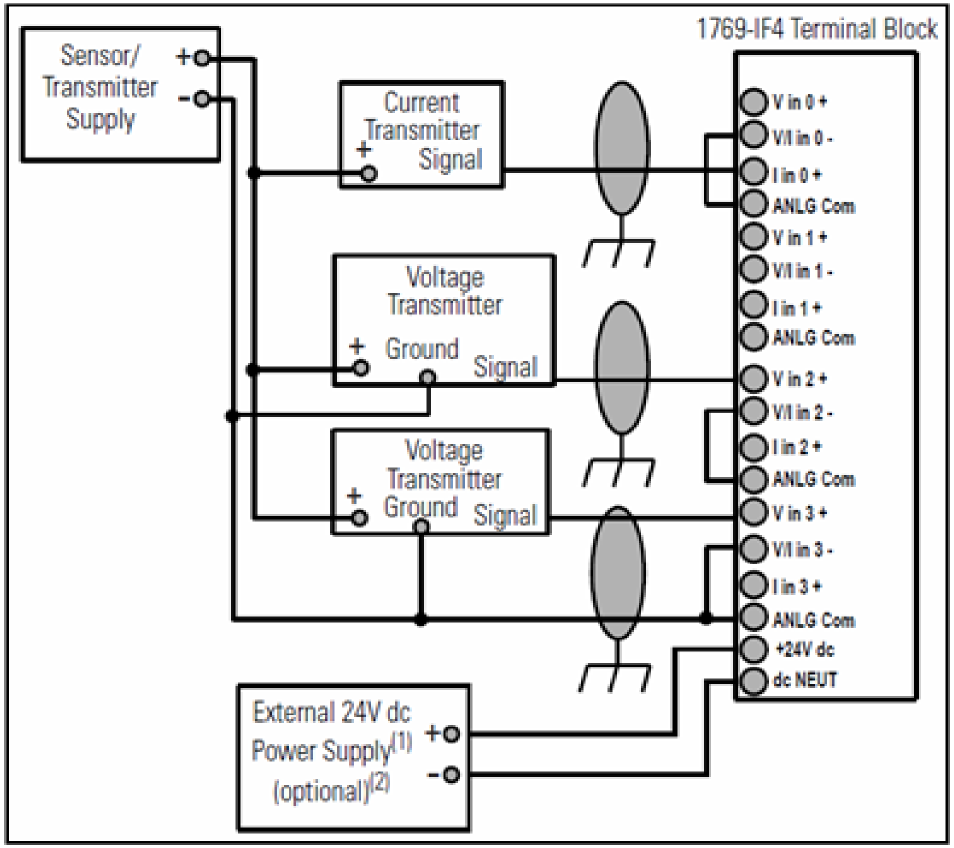
1769-IF4 Analog Input

- Configure input voltage range
  - -10V…10V DC
  - 0…10V DC
  - 0 …5V DC
  - 1…5V DC
- Configure input current range
  - 0…20mA
  - 4…20mA

ANALOG MODULES

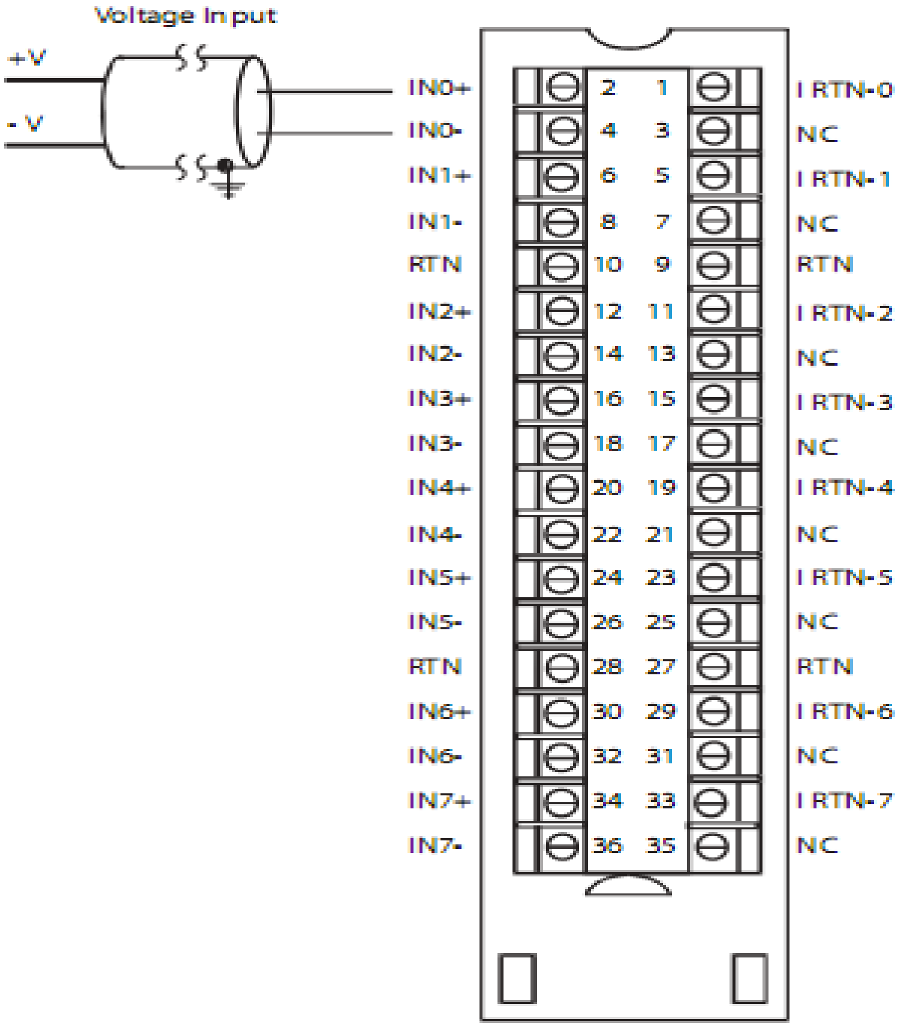
Connecting voltage and current Sensors





ANALOG MODULES

Analog Input Module, connecting voltage sensors



IN0+	2	1	I RTN-0
IN0-	4	3	NC
IN1+	6	5	I RTN-1
IN1-	8	7	NC
RTN	10	9	RTN
IN2+	12	11	I RTN-2
IN2-	14	13	NC
IN3+	16	15	I RTN-3
IN3-	18	17	NC
IN4+	20	19	I RTN-4
IN4-	22	21	NC
IN5+	24	23	I RTN-5
IN5-	26	25	NC
RTN	28	27	RTN
IN6+	30	29	I RTN-6
IN6-	32	31	NC
IN7+	34	33	I RTN-7
IN7-	36	35	NC

- Configure input voltage range

  -10V...10V DC

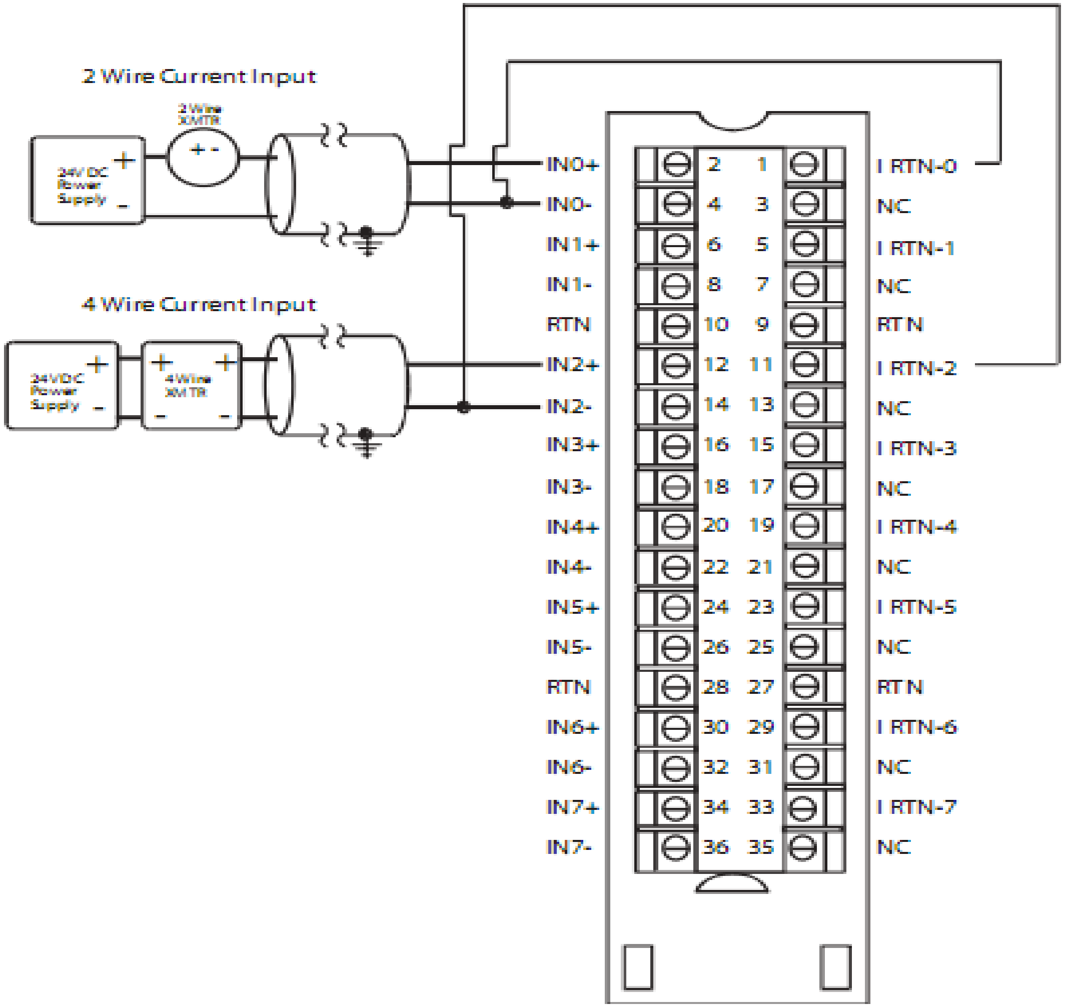
  0...10V DC

  0 ...5V DC

  1...5V DC

ANALOG MODULES

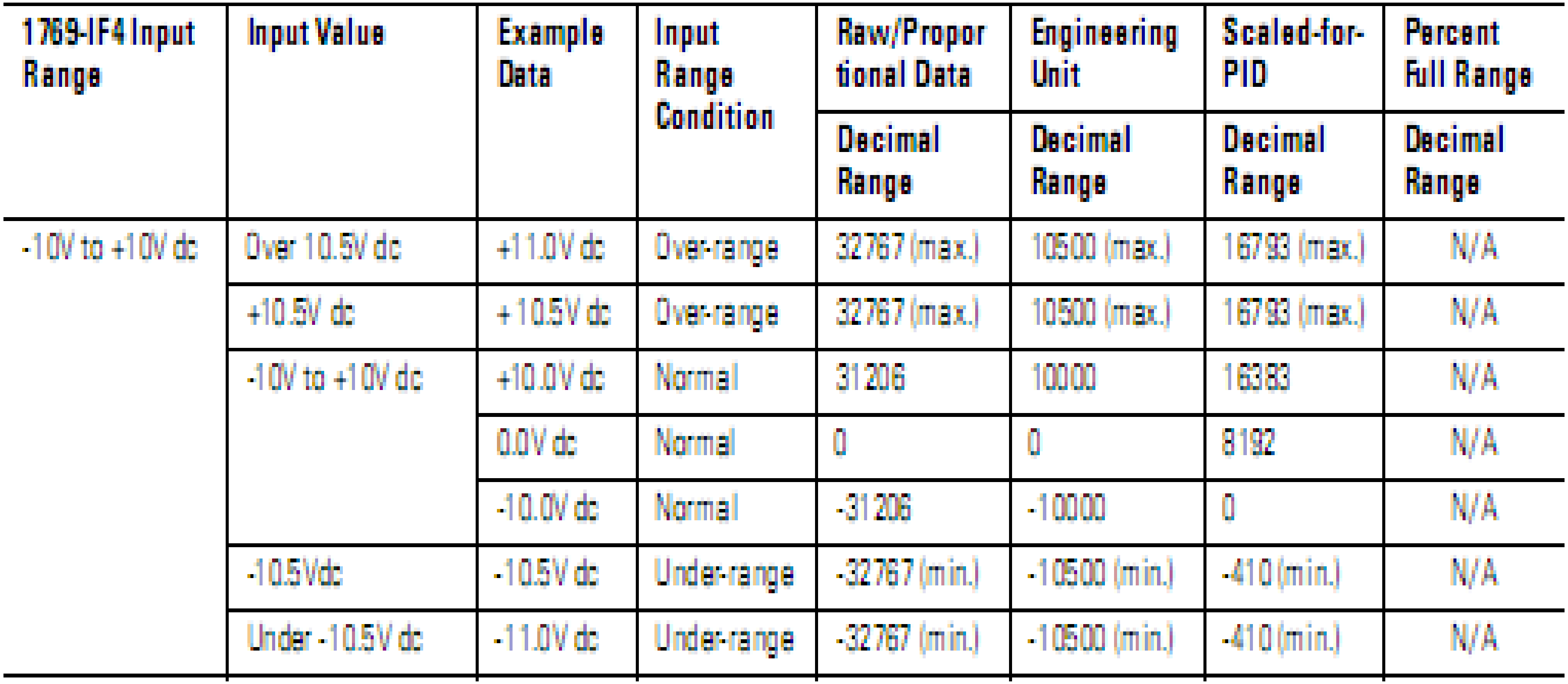
Analog Input Module, connecting current sensors



- Configure input current range 0…20mA or 4…20mA.

ANALOG VALUE REPRESENTATION

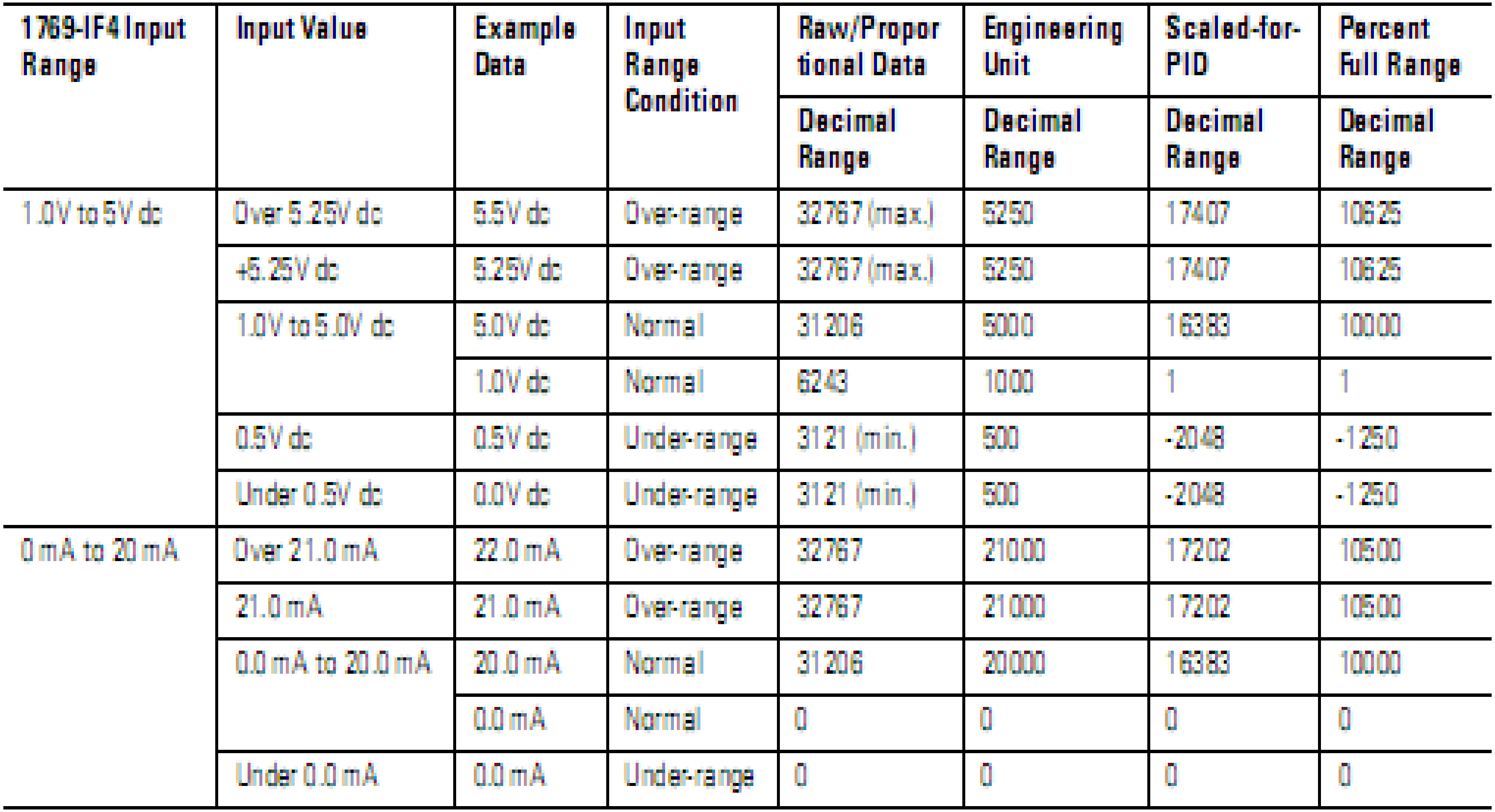
Valid Input Data



1769-IF4 Input Range	Input Value	Example Data	Input Range Condition	Raw/Proportional Data	Engineering Unit	Scaled-for-PID	Percent Full Range
				Decimal Range	Decimal Range	Decimal Range	Decimal Range
-10V to +10V dc	Over 10.5V dc	+11.0V dc	Over-range	32767 (max.)	10500 (max.)	16793 (max.)	N/A
	+10.5V dc	+10.5V dc	Over-range	32767 (max.)	10500 (max.)	16793 (max.)	N/A
	-10V to +10V dc	+10.0V dc	Normal	31206	10000	16383	N/A
		0.0V dc	Normal	0	0	8192	N/A
		-10.0V dc	Normal	-31206	-10000	0	N/A
	-10.5Vdc	-10.5V dc	Under-range	-32767 (min.)	-10500 (min.)	-410 (min.)	N/A
	Under -10.5V dc	-11.0V dc	Under-range	-32767 (min.)	-10500 (min.)	-410 (min.)	N/A

ANALOG VALUE REPRESENTATION

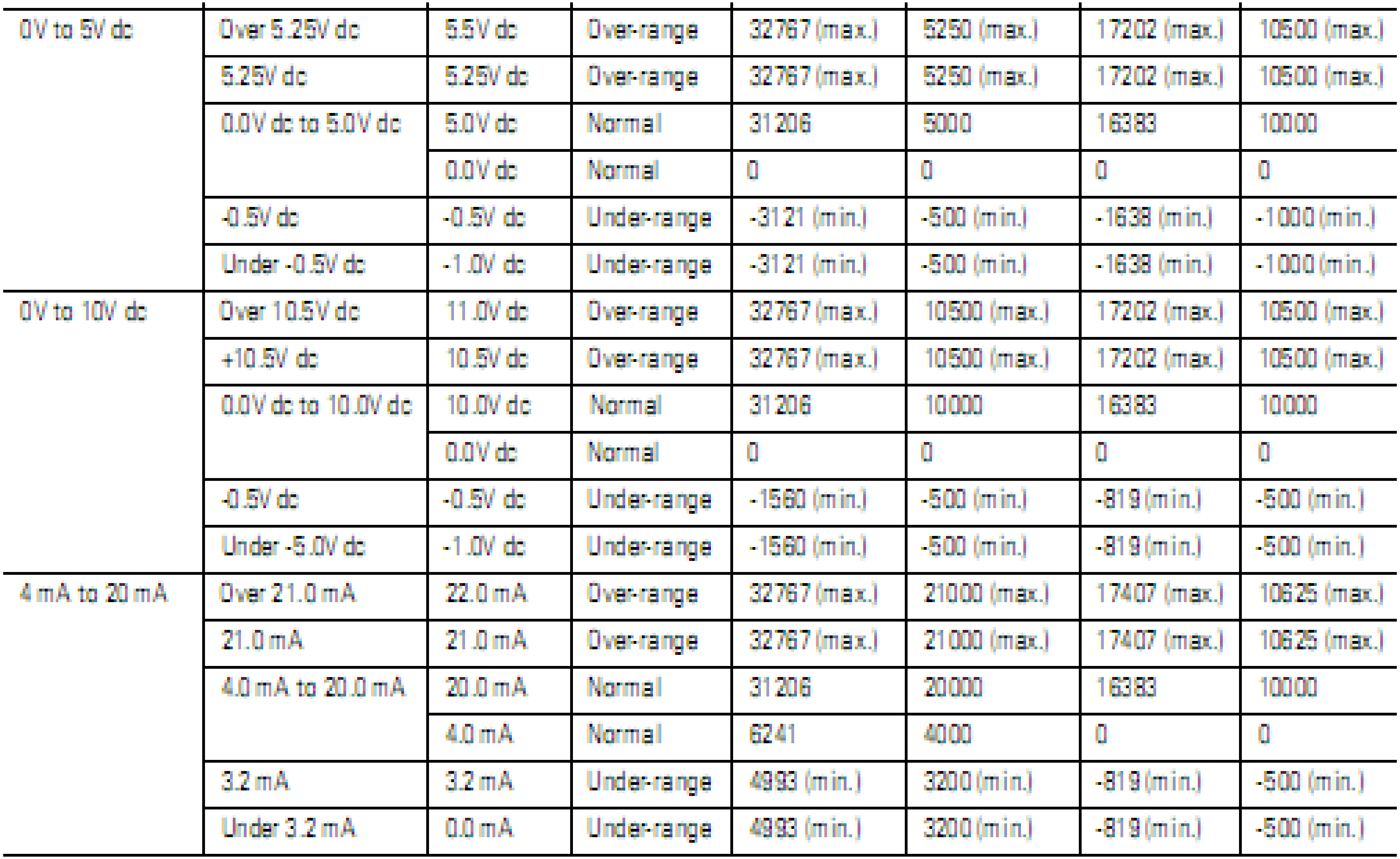
Valid Input Data



1769-IF4 Input Range	Input Value	Example Data	Input Range Condition	Raw/Proportional Data	Engineering Unit	Scaled-for-PID	Percent Full Range
				Decimal Range	Decimal Range	Decimal Range	Decimal Range
1.0V to 5V dc	Over 5.25V dc	5.5V dc	Over-range	32767 (max.)	5250	17407	10625
	+5.25V dc	5.25V dc	Over-range	32767 (max.)	5250	17407	10625
	1.0V to 5.0V dc	5.0V dc	Normal	31206	5000	16383	10000
		1.0V dc	Normal	6243	1000	1	1
	0.5V dc	0.5V dc	Under-range	3121 (min.)	500	-2048	-1250
	Under 0.5V dc	0.0V dc	Under-range	3121 (min.)	500	-2048	-1250
0 mA to 20 mA	Over 21.0 mA	22.0 mA	Over-range	32767	21000	17202	10500
	21.0 mA	21.0 mA	Over-range	32767	21000	17202	10500
	0.0 mA to 20.0 mA	20.0 mA	Normal	31206	20000	16383	10000
		0.0 mA	Normal	0	0	0	0
	Under 0.0 mA	0.0 mA	Under-range	0	0	0	0

ANALOG VALUE REPRESENTATION

Valid Input Data



0V to 5V dc	Over 5.25V dc	5.5V dc	Over-range	32767 (max.)	5250 (max.)	17202 (max.)	10500 (max.)
	5.25V dc	5.25V dc	Over-range	32767 (max.)	5250 (max.)	17202 (max.)	10500 (max.)
	0.0V dc to 5.0V dc	5.0V dc	Normal	31206	5000	16383	10000
		0.0V dc	Normal	0	0	0	0
	-0.5V dc	-0.5V dc	Under-range	-3121 (min.)	-500 (min.)	-1638 (min.)	-1000 (min.)
	Under -0.5V dc	-1.0V dc	Under-range	-3121 (min.)	-500 (min.)	-1638 (min.)	-1000 (min.)
0V to 10V dc	Over 10.5V dc	11.0V dc	Over-range	32767 (max.)	10500 (max.)	17202 (max.)	10500 (max.)
	+10.5V dc	10.5V dc	Over-range	32767 (max.)	10500 (max.)	17202 (max.)	10500 (max.)
	0.0V dc to 10.0V dc	10.0V dc	Normal	31206	10000	16383	10000
		0.0V dc	Normal	0	0	0	0
	-0.5V dc	-0.5V dc	Under-range	-1560 (min.)	-500 (min.)	-819 (min.)	-500 (min.)
	Under -5.0V dc	-1.0V dc	Under-range	-1560 (min.)	-500 (min.)	-819 (min.)	-500 (min.)
4 mA to 20 mA	Over 21.0 mA	22.0 mA	Over-range	32767 (max.)	21000 (max.)	17407 (max.)	10625 (max.)
	21.0 mA	21.0 mA	Over-range	32767 (max.)	21000 (max.)	17407 (max.)	10625 (max.)
	4.0 mA to 20.0 mA	20.0 mA	Normal	31206	20000	16383	10000
		4.0 mA	Normal	6241	4000	0	0
	3.2 mA	3.2 mA	Under-range	4993 (min.)	3200 (min.)	-819 (min.)	-500 (min.)
	Under 3.2 mA	0.0 mA	Under-range	4993 (min.)	3200 (min.)	-819 (min.)	-500 (min.)

ANALOG MODULES

1769-OF2 Analog Output

- Configure input voltage range

  -10V…10V DC

  0…10V DC

  0…5V DC

  1…5V DC.

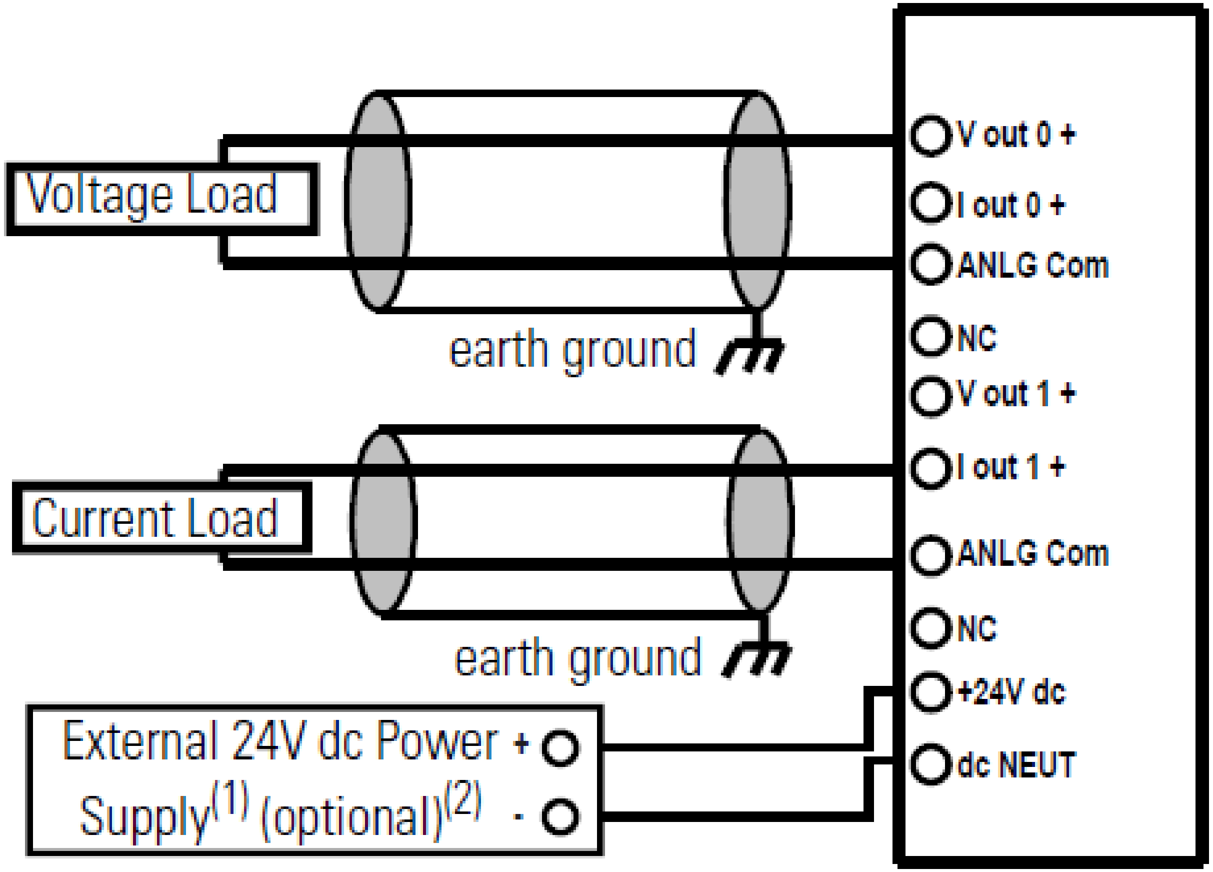
- Configure input current range

  0…20mA

  4…20mA

ANALOG MODULES

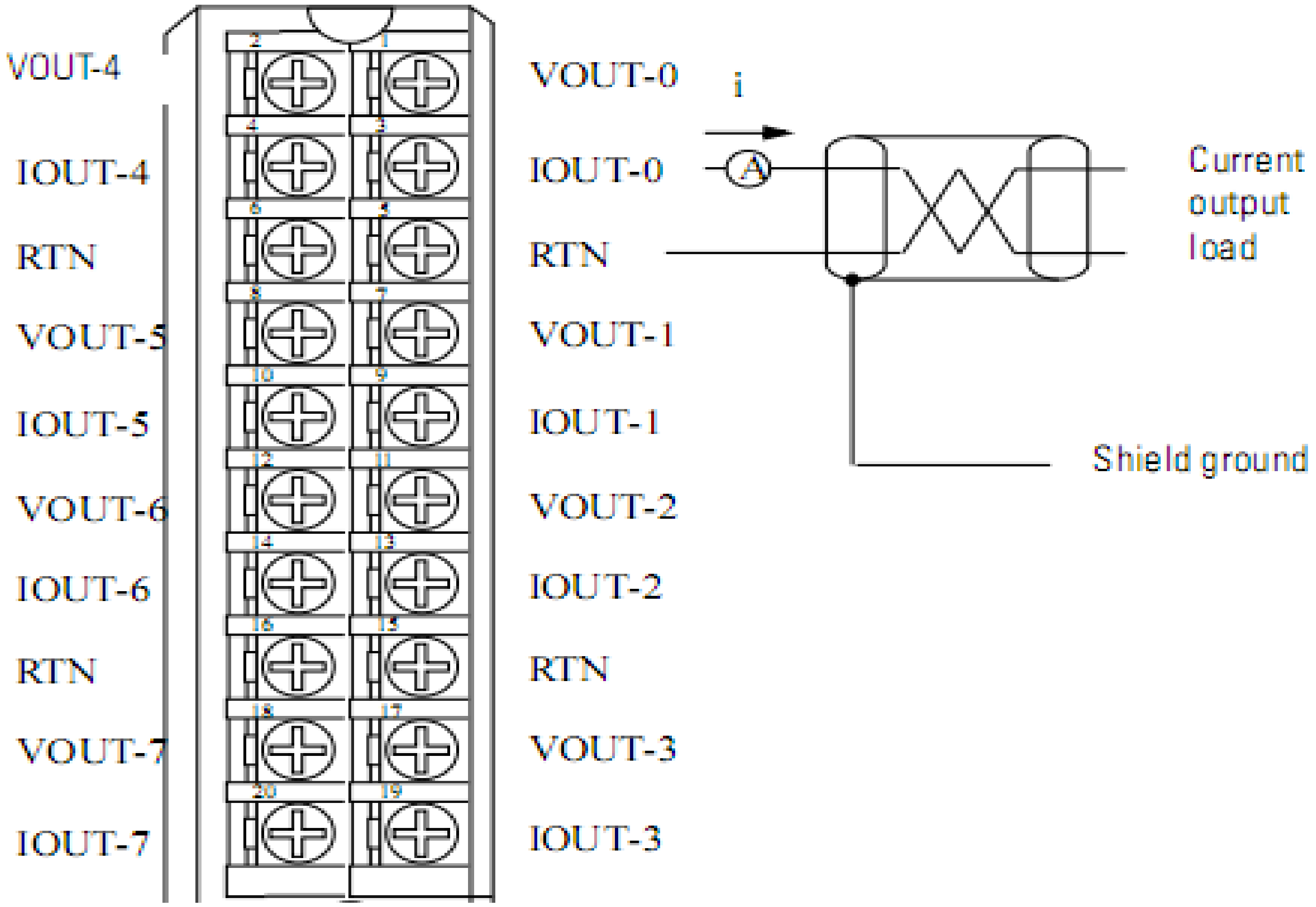
Connecting Actuators to current and voltage Output



ANALOG MODULES

Connecting Actuator to current Output

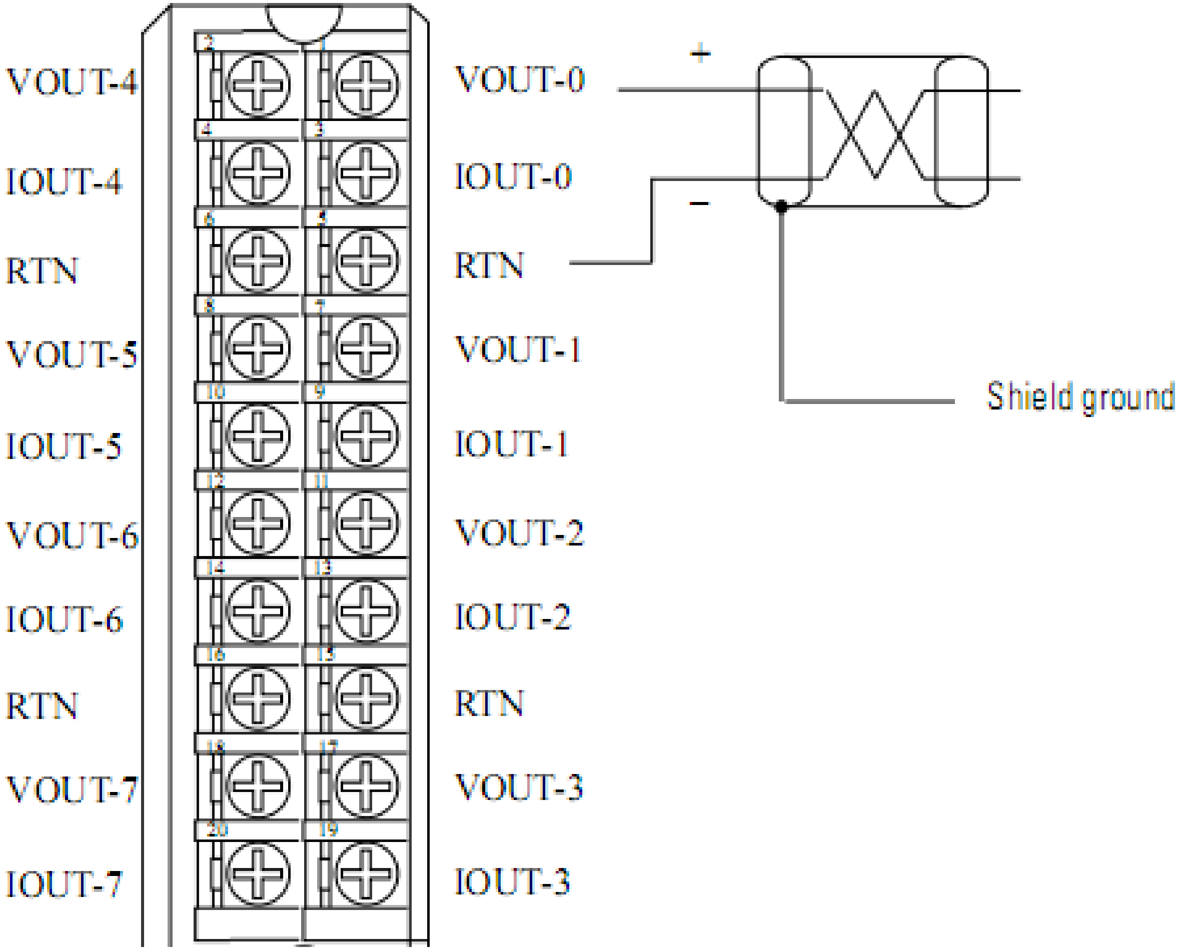
1756-OF8 Current wiring example



ANALOG MODULES

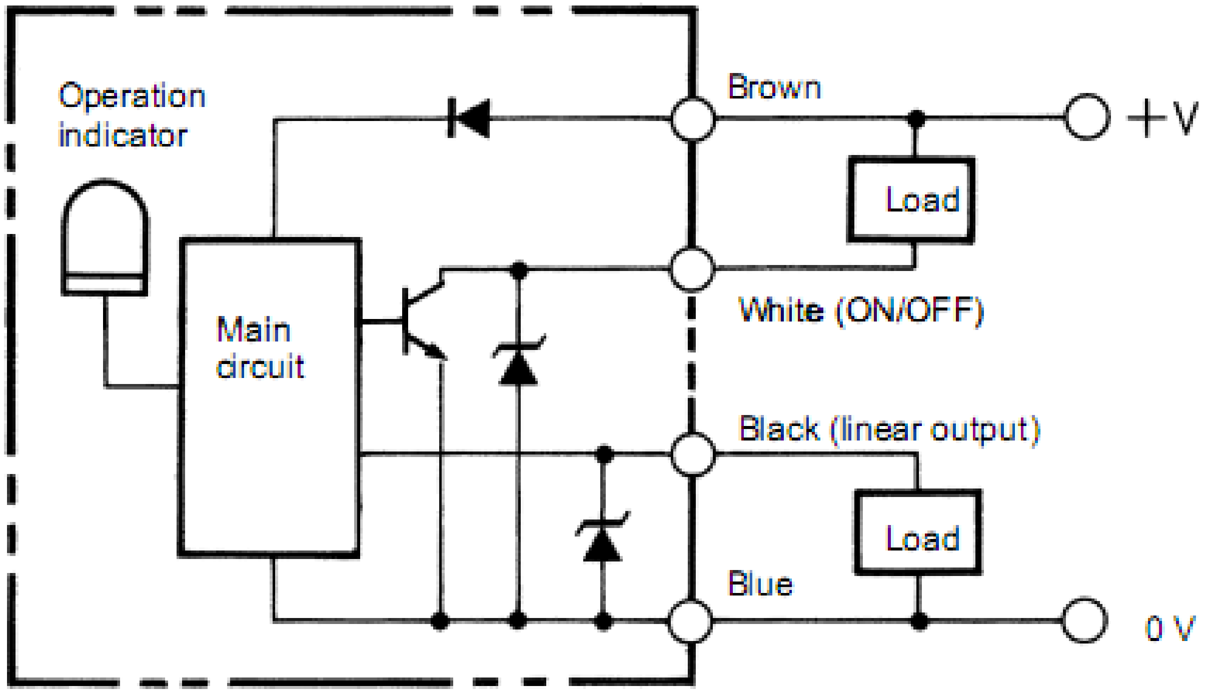
Connecting Actuator to voltage output

1756-OF8 Voltage wiring example



CONNECTING ANALOG INPUT MODULE

EX16: Connecting pressure sensor with voltage output to input analog module

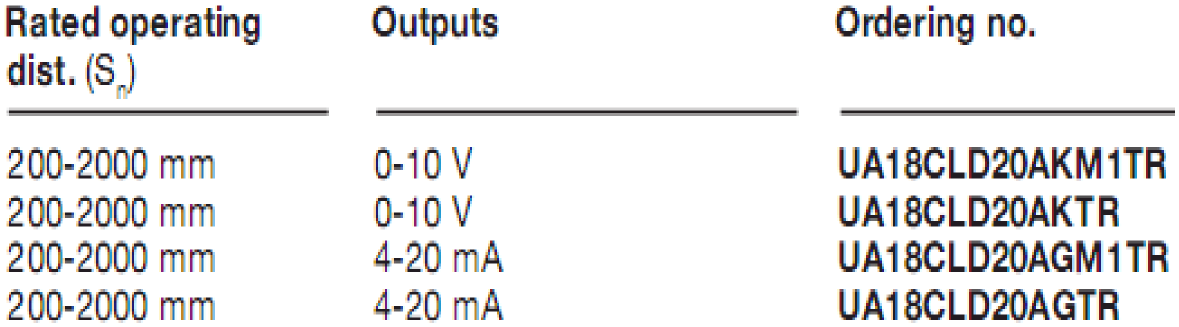


CONNECTING ANALOG INPUT MODULE

EX16: Answer

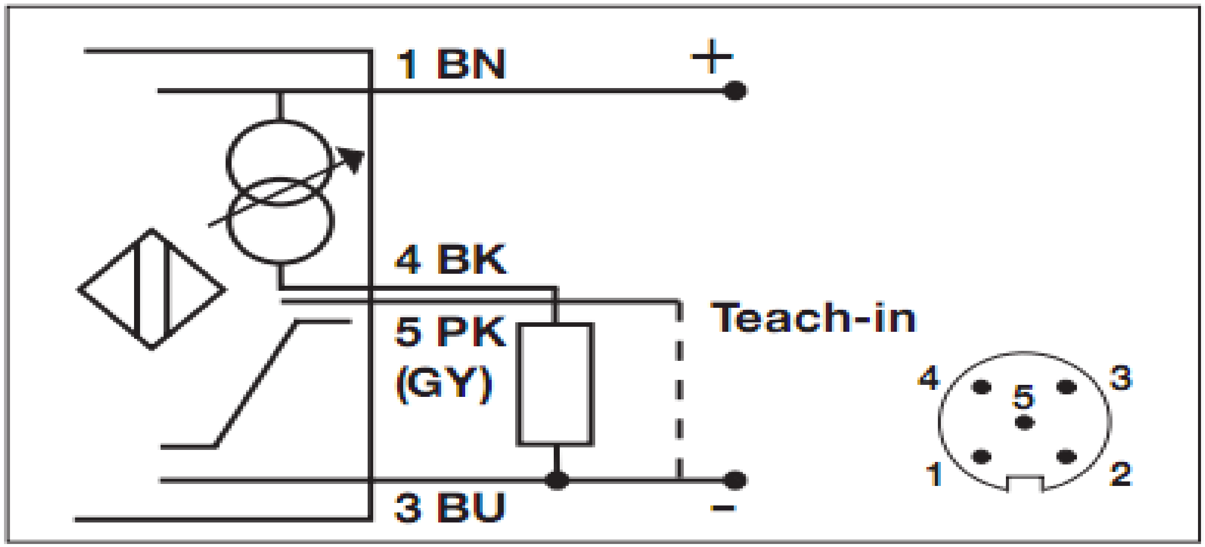
CONNECTING ANALOG INPUT MODULE

EX17: Connecting an Ultrasonic sensor with current output to input analog module



Rated operating dist. (S_n)	Outputs	Ordering no.
200-2000 mm	0-10 V	UA18CLD20AKM1TR
200-2000 mm	0-10 V	UA18CLD20AKTR
200-2000 mm	4-20 mA	UA18CLD20AGM1TR
200-2000 mm	4-20 mA	UA18CLD20AGTR

Wiring Diagram

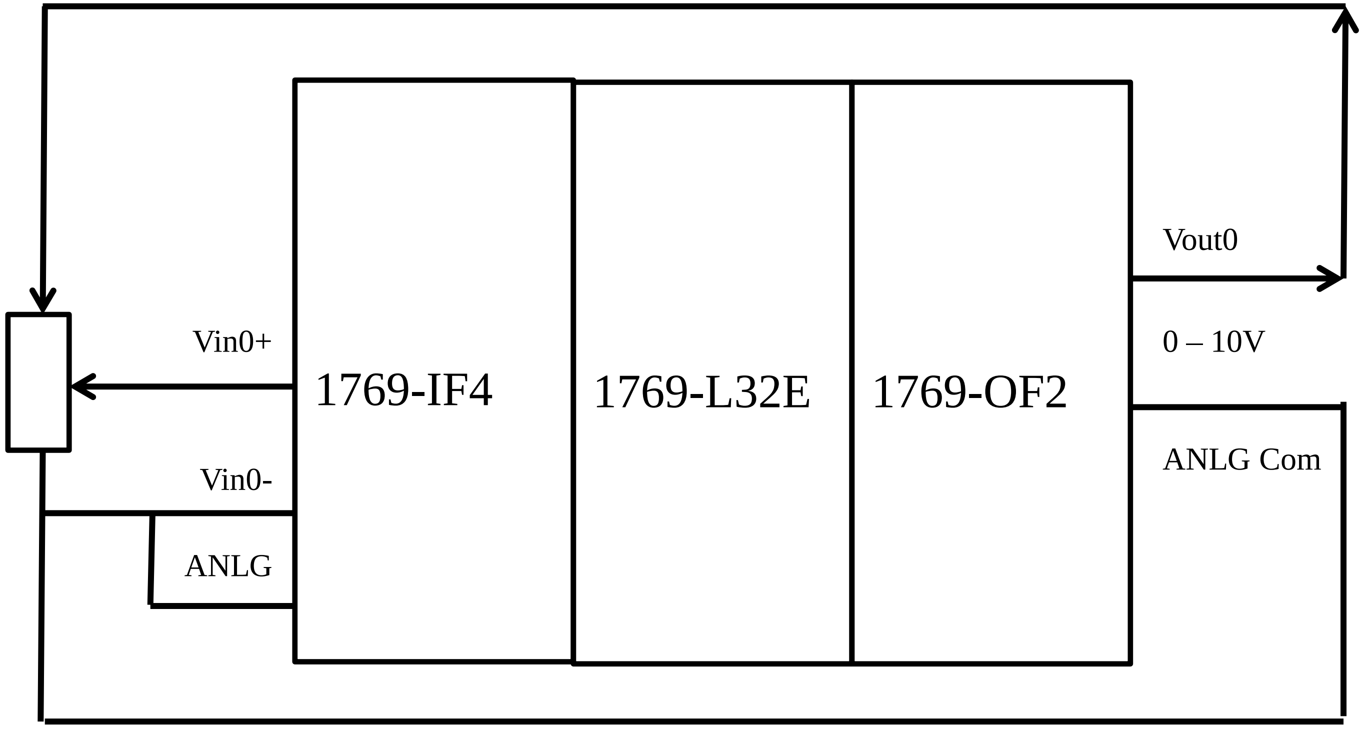


CONNECTING ANALOG INPUT MODULE

EX17: Answer

CONNECTING ANALOG INPUT MODULE

EX18: Program to output 10V at 1769-OF2 module
Connecting an potentiometer to 1769-IF4 and program to calculate voltage at input of the module

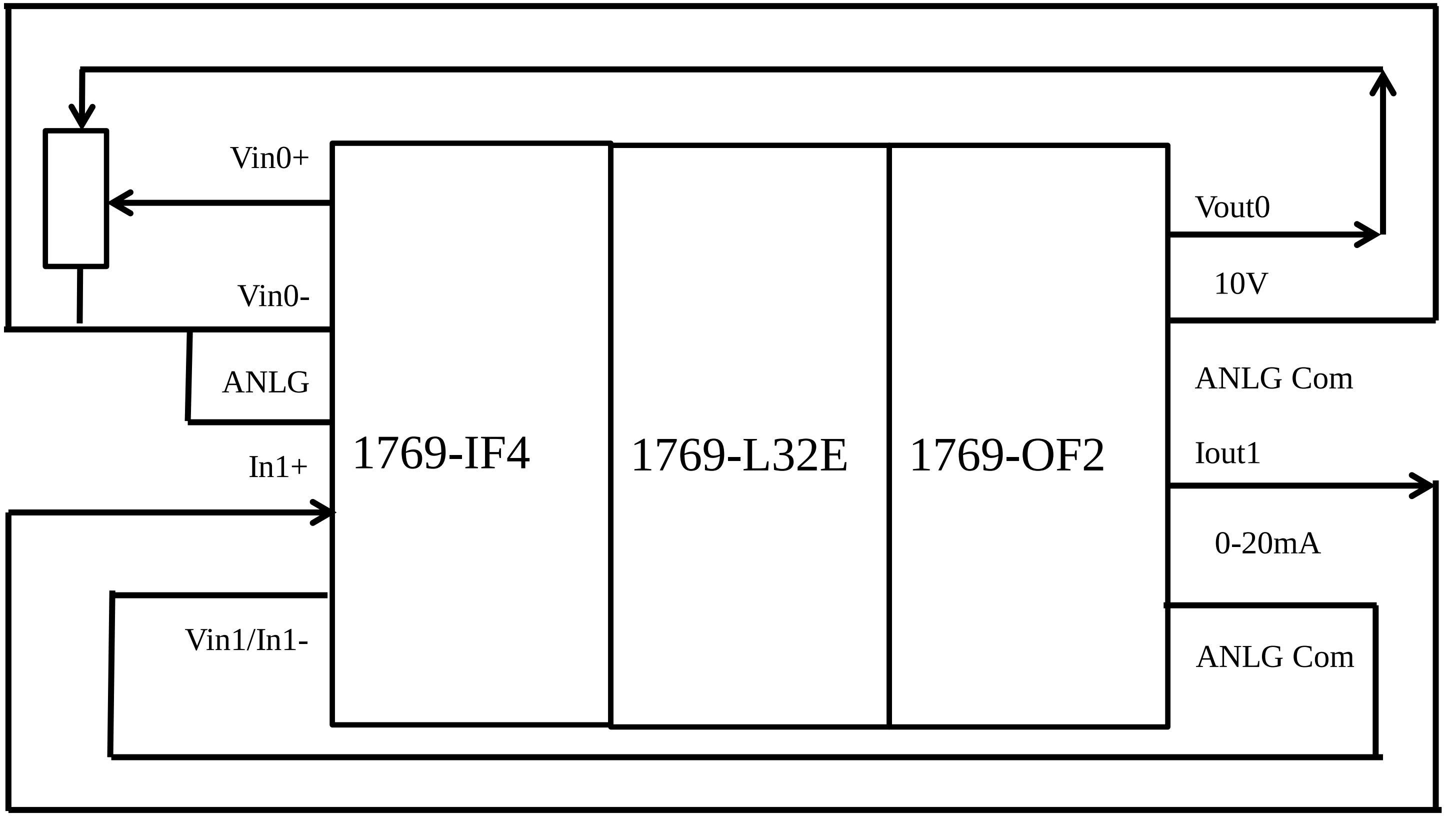


CONNECTING ANALOG INPUT MODULE

EX19: Program to output Votage(10V) at 1769-OF2 module

Connecting an potentiometer to 1769-IF4 and program to output I(mA) at **Iout1** of OF2 module.

Connecting Iout1 to Iin1 and program to calculate I(mA) at input module.



RSLOGIX 5000 CONTROLLER TASKS

A RSLogix 5000 supports three type of tasks

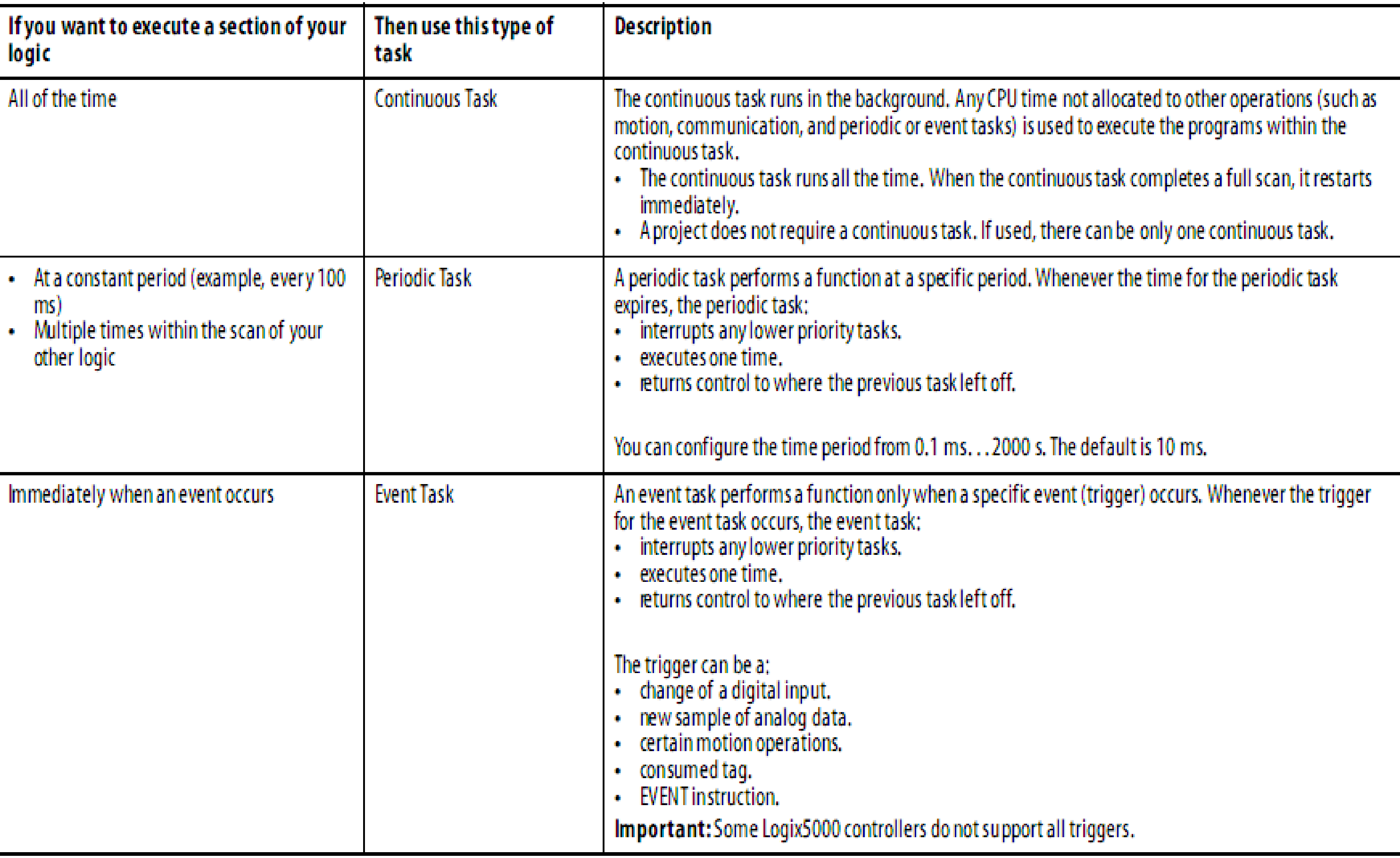
- Continuous Tasks
- Periodic Task
- Event Task

Characteristic of Tasks

- The controller executes only one Task at one time
- A Task can interrupt a different task that is executing and take control if it has high priority
- In any given Task, only one program executes at one time.

RSLOGIX 5000 CONTROLLER TASKS

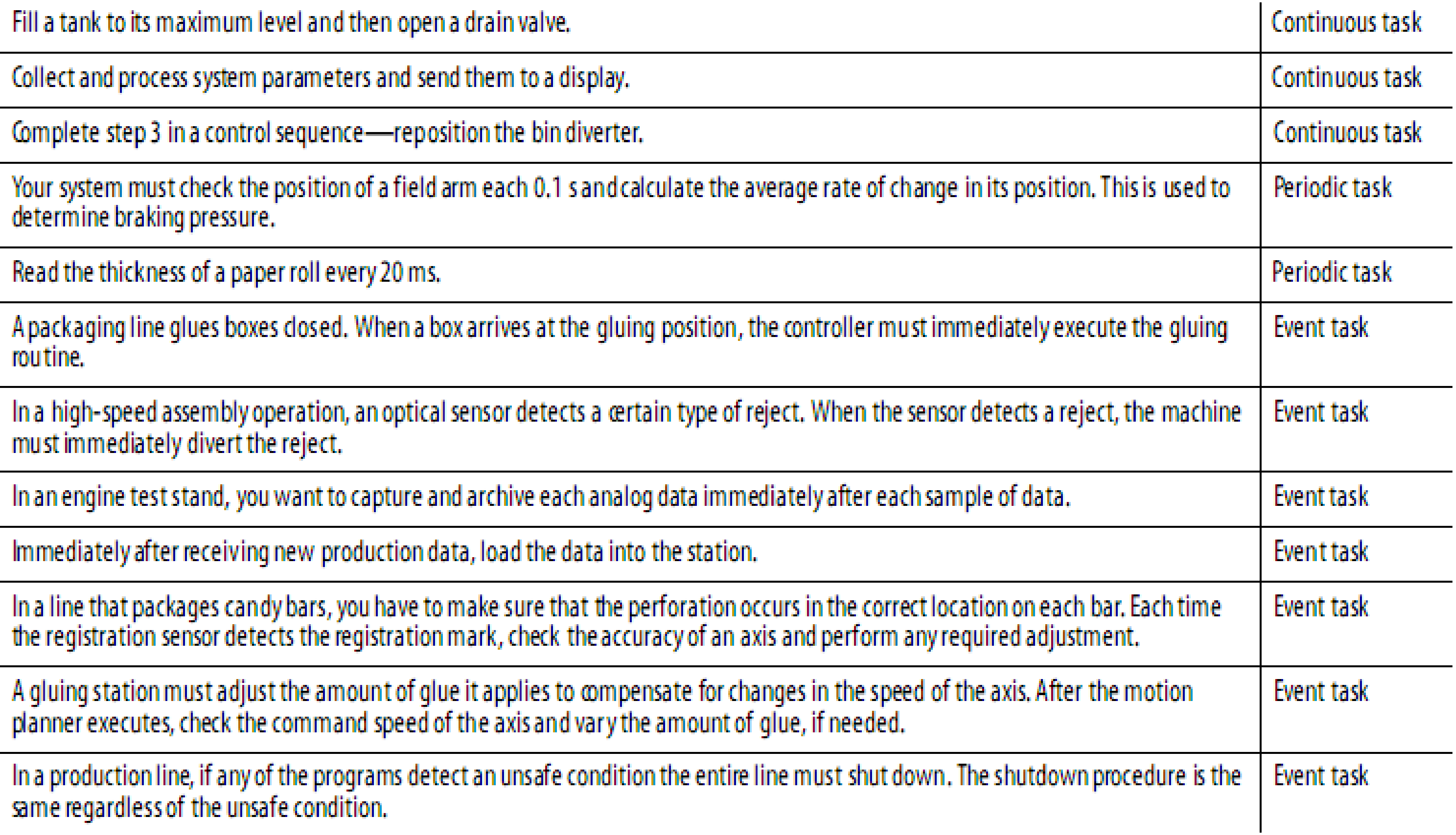
Function of Tasks



If you want to execute a section of your logic	Then use this type of task	Description
All of the time	Continuous Task	The continuous task runs in the background. Any CPU time not allocated to other operations (such as motion, communication, and periodic or event tasks) is used to execute the programs within the continuous task. • The continuous task runs all the time. When the continuous task completes a full scan, it restarts immediately. • A project does not require a continuous task. If used, there can be only one continuous task.
• At a constant period (example, every 100 ms) • Multiple times within the scan of your other logic	Periodic Task	A periodic task performs a function at a specific period. Whenever the time for the periodic task expires, the periodic task: • interrupts any lower priority tasks. • executes one time. • returns control to where the previous task left off. You can configure the time period from 0.1 ms…2000 s. The default is 10 ms.
Immediately when an event occurs	Event Task	An event task performs a function only when a specific event (trigger) occurs. Whenever the trigger for the event task occurs, the event task: • interrupts any lower priority tasks. • executes one time. • returns control to where the previous task left off. The trigger can be a: • change of a digital input. • new sample of analog data. • certain motion operations. • consumed tag. • EVENT instruction. **Important:** Some Logix5000 controllers do not support all triggers.

RSLOGIX 5000 CONTROLLER TASKS

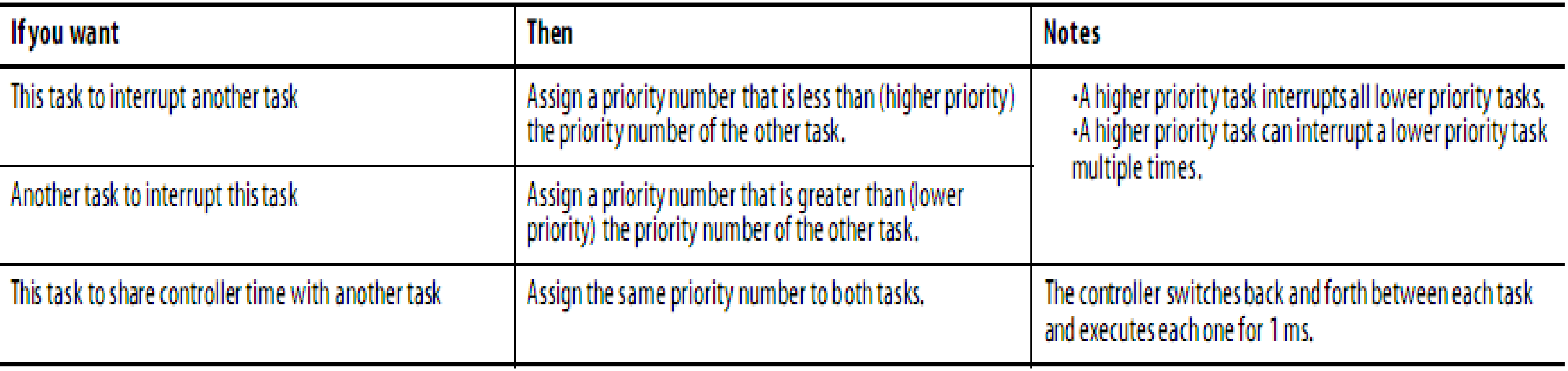
Examples for using Tasks



Fill a tank to its maximum level and then open a drain valve.	Continuous task
Collect and process system parameters and send them to a display.	Continuous task
Complete step 3 in a control sequence—reposition the bin diverter.	Continuous task
Your system must check the position of a field arm each 0.1 s and calculate the average rate of change in its position. This is used to determine braking pressure.	Periodic task
Read the thickness of a paper roll every 20 ms.	Periodic task
A packaging line glues boxes closed. When a box arrives at the gluing position, the controller must immediately execute the gluing routine.	Event task
In a high-speed assembly operation, an optical sensor detects a certain type of reject. When the sensor detects a reject, the machine must immediately divert the reject.	Event task
In an engine test stand, you want to capture and archive each analog data immediately after each sample of data.	Event task
Immediately after receiving new production data, load the data into the station.	Event task
In a line that packages candy bars, you have to make sure that the perforation occurs in the correct location on each bar. Each time the registration sensor detects the registration mark, check the accuracy of an axis and perform any required adjustment.	Event task
A gluing station must adjust the amount of glue it applies to compensate for changes in the speed of the axis. After the motion planner executes, check the command speed of the axis and vary the amount of glue, if needed.	Event task
In a production line, if any of the programs detect an unsafe condition the entire line must shut down. The shutdown procedure is the same regardless of the unsafe condition.	Event task

RSLOGIX 5000 CONTROLLER TASKS

Priority Periodic and Event Tasks: The priority of each task tells the controller what to do

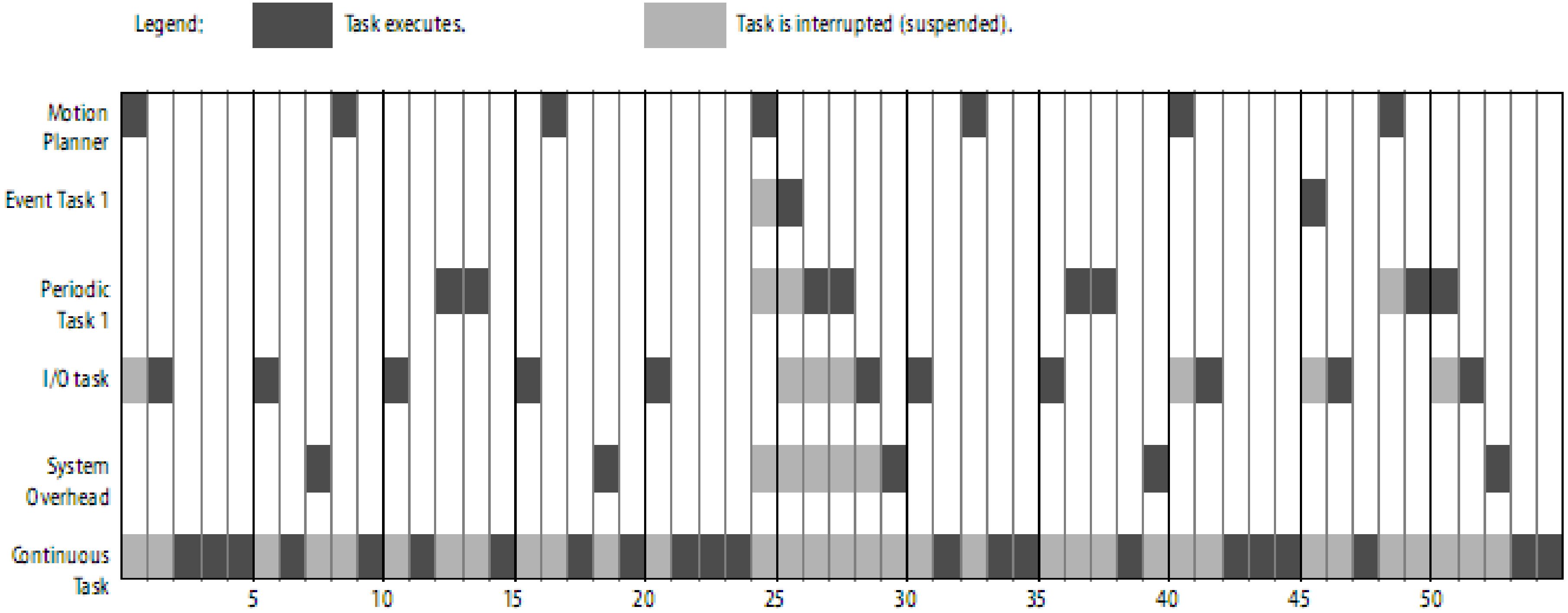


If you want	Then	Notes
This task to interrupt another task	Assign a priority number that is less than (higher priority) the priority number of the other task.	•A higher priority task interrupts all lower priority tasks. •A higher priority task can interrupt a lower priority task multiple times.
Another task to interrupt this task	Assign a priority number that is greater than (lower priority) the priority number of the other task.	
This task to share controller time with another task	Assign the same priority number to both tasks.	The controller switches back and forth between each task and executes each one for 1 ms.

RSLOGIX 5000 CONTROLLER TASKS

This example depicts execution of a project with three tasks

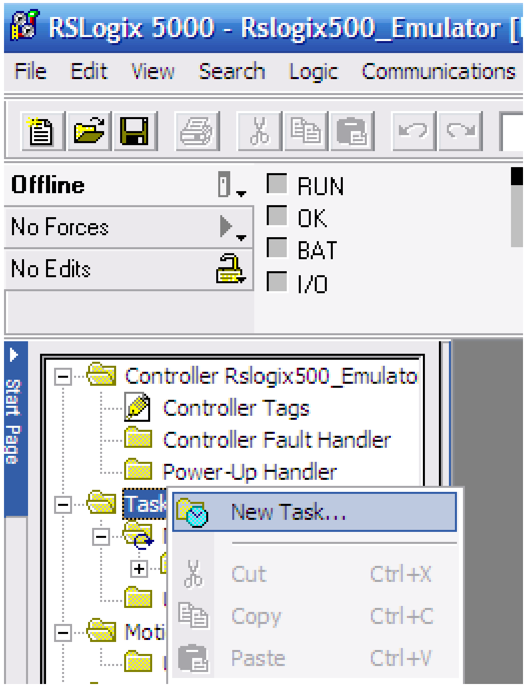
Task	Priority	Period	Execution time	Duration
Motion planner	N/A	8 ms (course update rate)	1 ms	1 ms
Event task 1	1	N/A	1 ms	1…2 ms
Periodic task 1	2	12 ms	2 ms	2…4 ms
I/O task—n/a to ControlLogix and SoftLogix controllers. See page 11.	7	5 ms (fastest RPI)	1 ms	1…5 ms
System overhead	N/A	Time slice = 20%	1 ms	1…6 ms
Continuous task	N/A	N/A	20 ms	48 ms

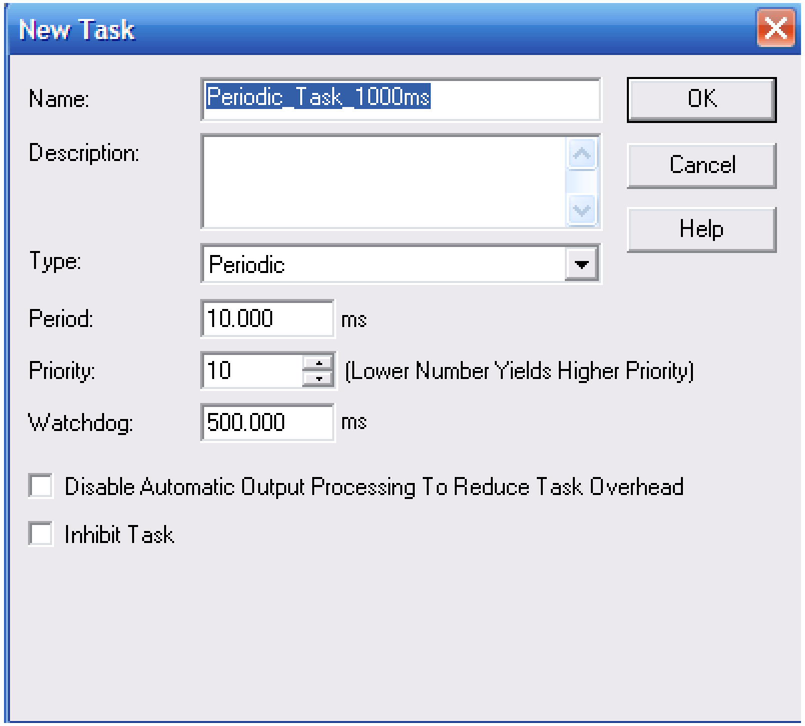


PROGRAM FOR PERIODIC TASKS

Create a Periodic Task, Put an appropriate name, select Task Type, Periodic and Priority, create a program and write a logic program

- **Create a Periodic Task, enter an appropriate name**




New Task

Name: Periodic_Task_1000ms

Description:

Type: Periodic

Period: 10.000 ms

Priority: 10 (Lower Number Yields Higher Priority)

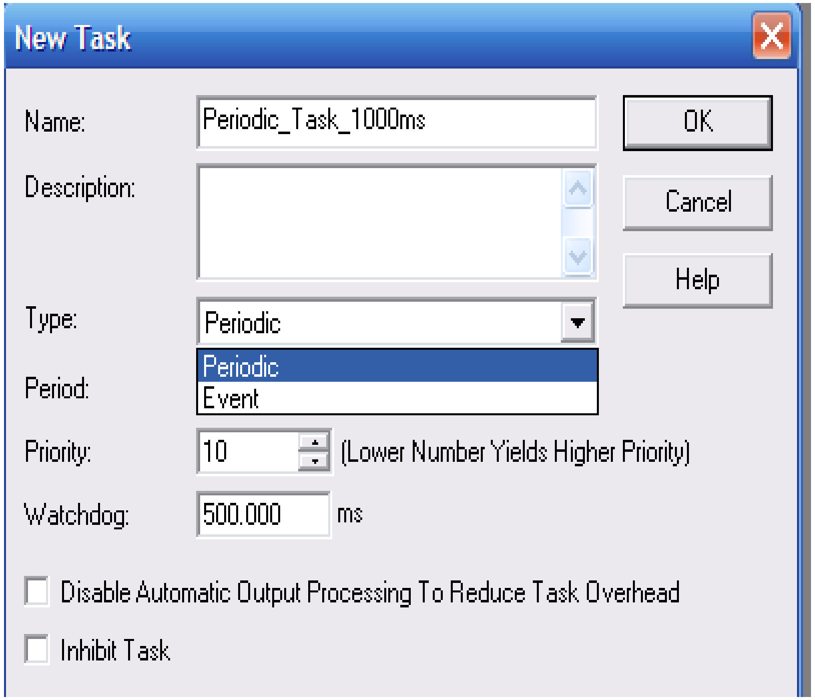
Watchdog: 500.000 ms

□ Disable Automatic Output Processing To Reduce Task Overhead

□ Inhibit Task

PROGRAM FOR PERIODIC TASKS

- **Select Task Type, Periodic and Priority**



New Task

Name: Periodic_Task_1000ms

Description:

Type: Periodic

Periodic
Event

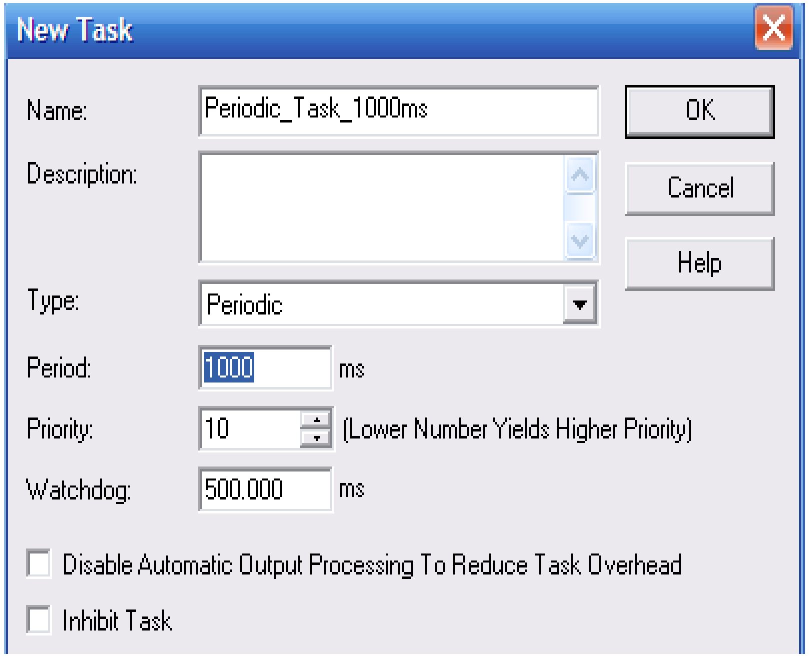
Period:

Priority: 10 (Lower Number Yields Higher Priority)

Watchdog: 500.000 ms

□ Disable Automatic Output Processing To Reduce Task Overhead

□ Inhibit Task



New Task

Name: Periodic_Task_1000ms

Description:

Type: Periodic

Period: 1000 ms

Priority: 10 (Lower Number Yields Higher Priority)

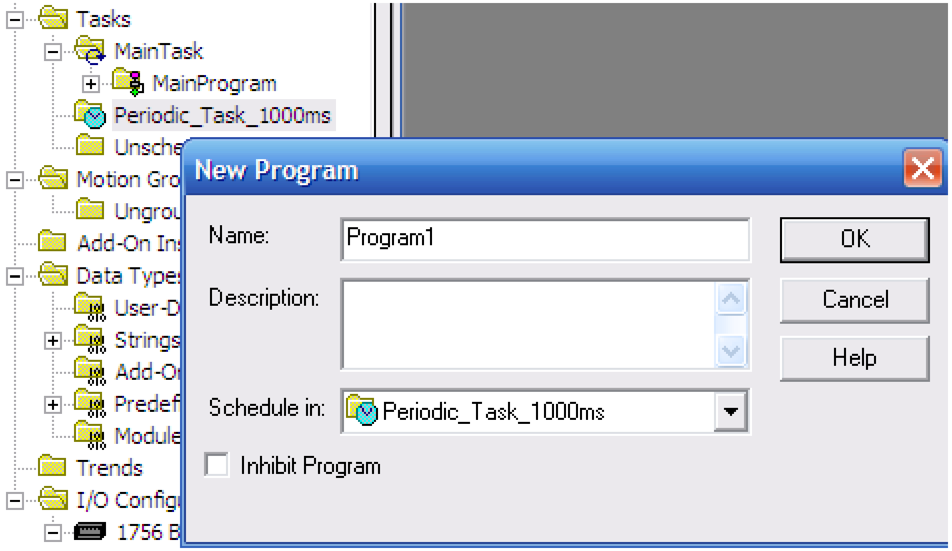
Watchdog: 500.000 ms

□ Disable Automatic Output Processing To Reduce Task Overhead

□ Inhibit Task

PROGRAM FOR PERIODIC TASKS

- Create a new Program with appropriate name and a new routine



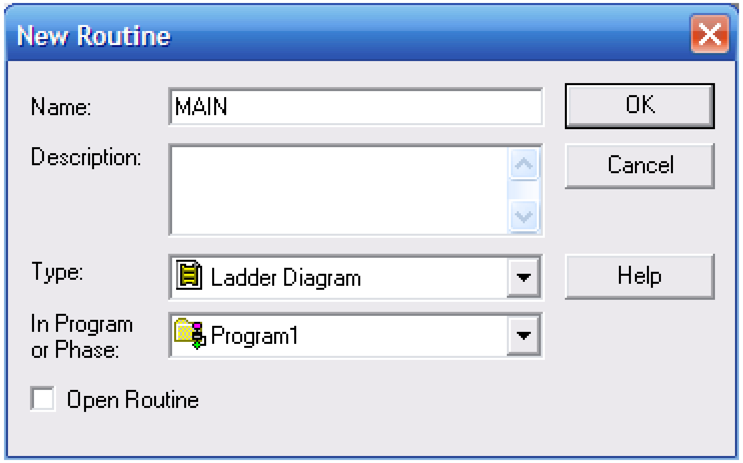
New Program

Name: Program1

Description:

Schedule in: Periodic_Task_1000ms

□ Inhibit Program



New Routine

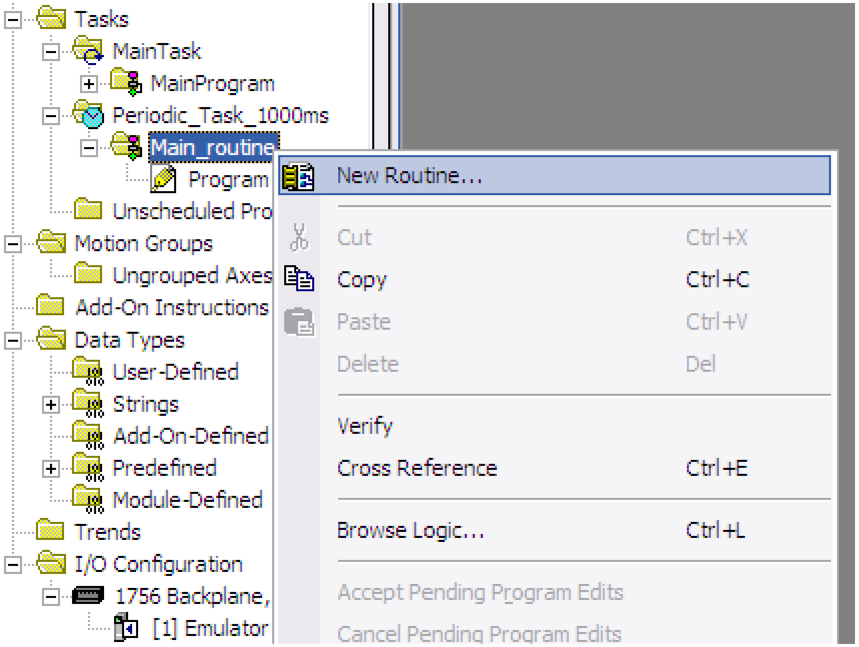
Name: MAIN

Description:

Type: Ladder Diagram

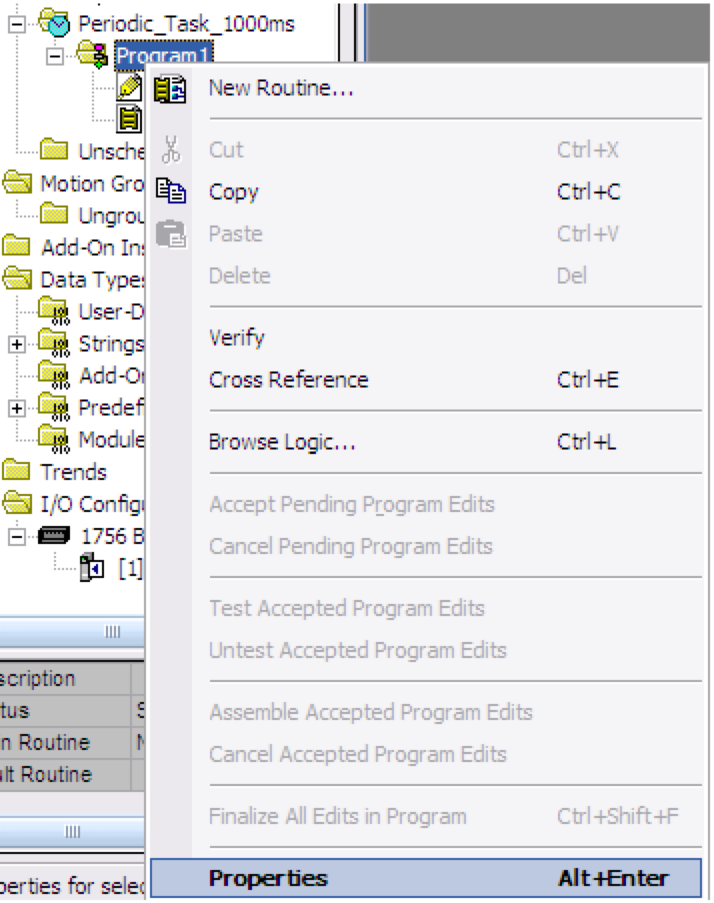
In Program or Phase: Program1

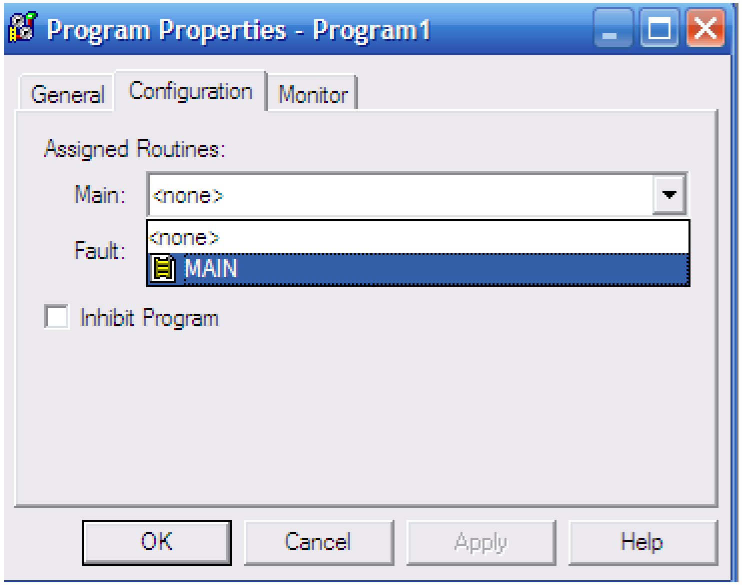
□ Open Routine



PROGRAM FOR PERIODIC TASKS

- **Select Main Routine for writing logic program**





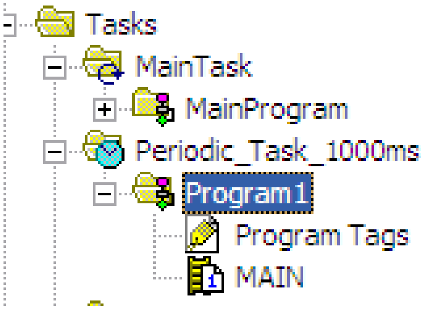
Assigned Routines:

Main: <none>

<none>
MAIN

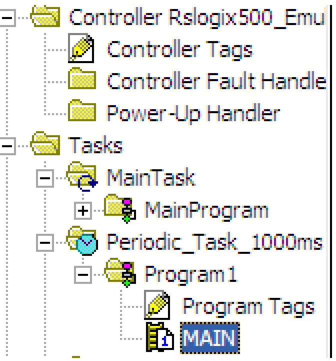
Fault:

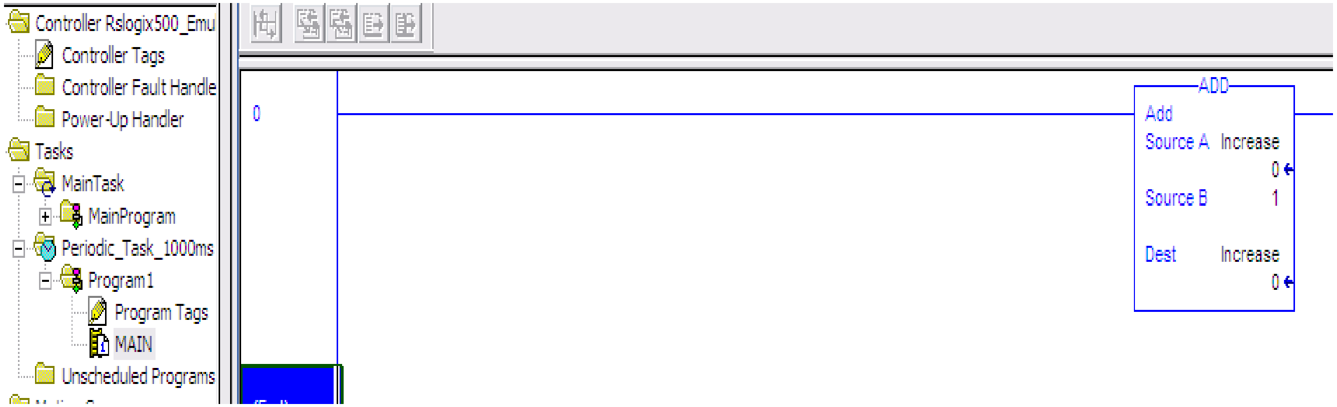
□ Inhibit Program



PROGRAM FOR PERIODIC TASKS

- **Select Main Routine for writing a Program**

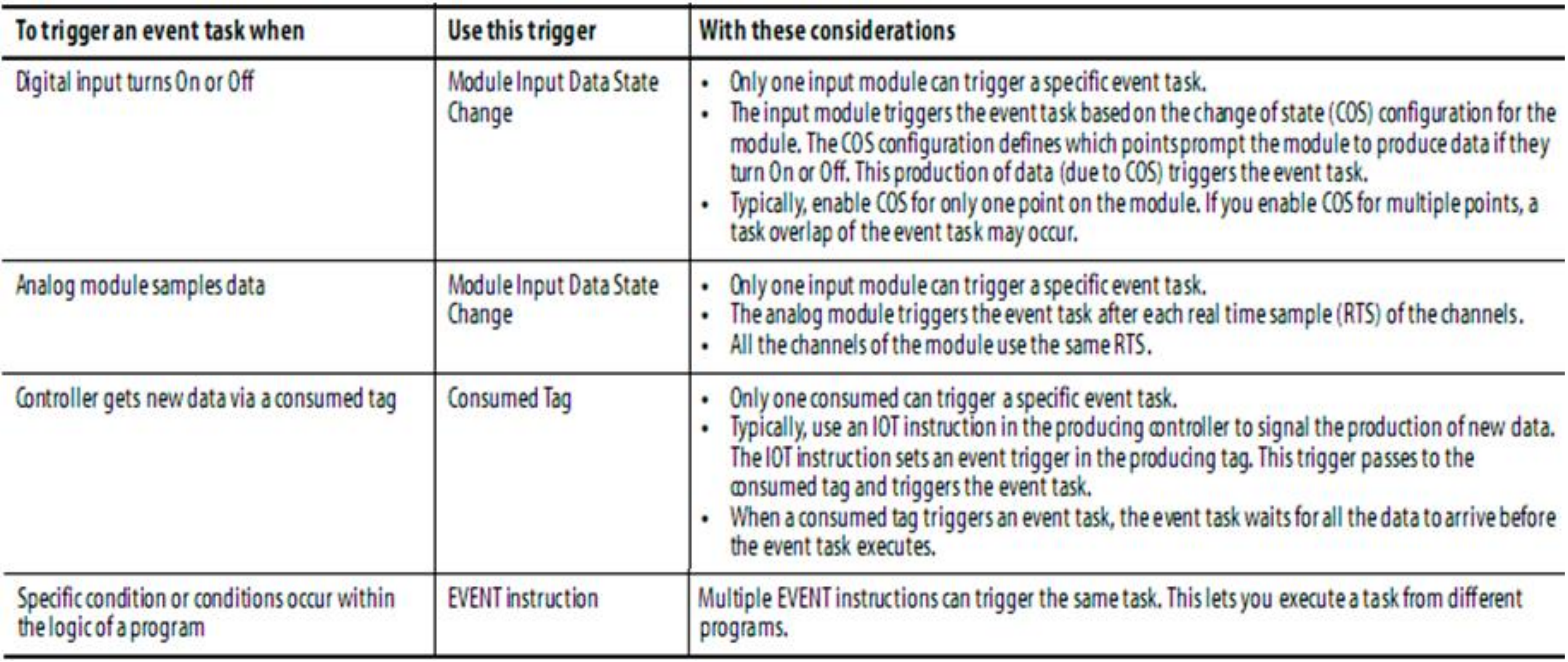




Add Instruction will executed one every 1000ms

MANAGE EVENT TASKS

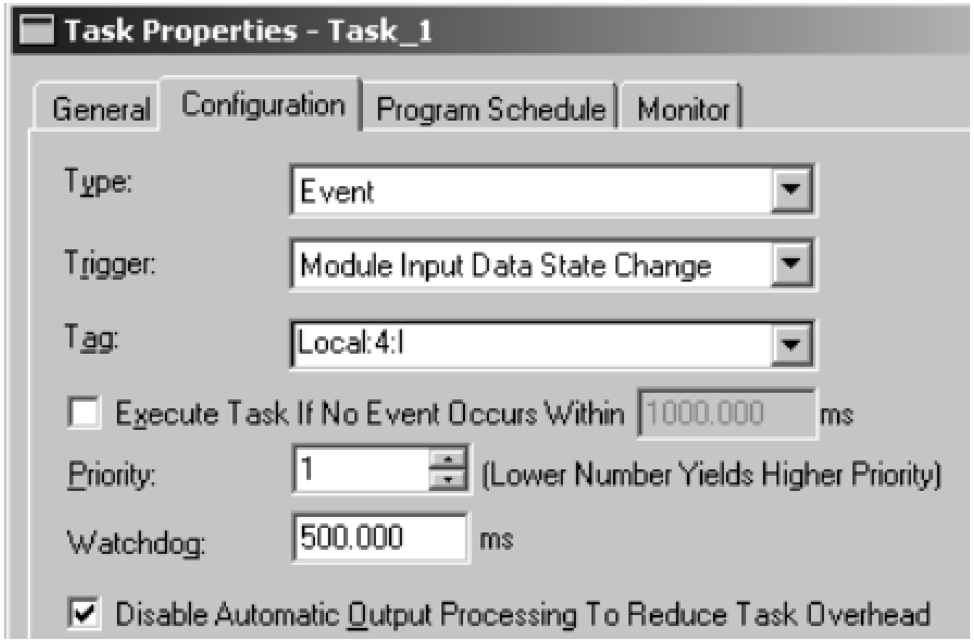
Choose the Trigger for an Event Task



To trigger an event task when	Use this trigger	With these considerations
Digital input turns On or Off	Module Input Data State Change	• Only one input module can trigger a specific event task. • The input module triggers the event task based on the change of state (COS) configuration for the module. The COS configuration defines which points prompt the module to produce data if they turn On or Off. This production of data (due to COS) triggers the event task. • Typically, enable COS for only one point on the module. If you enable COS for multiple points, a task overlap of the event task may occur.
Analog module samples data	Module Input Data State Change	• Only one input module can trigger a specific event task. • The analog module triggers the event task after each real time sample (RTS) of the channels. • All the channels of the module use the same RTS.
Controller gets new data via a consumed tag	Consumed Tag	• Only one consumed can trigger a specific event task. • Typically, use an IOT instruction in the producing controller to signal the production of new data. The IOT instruction sets an event trigger in the producing tag. This trigger passes to the consumed tag and triggers the event task. • When a consumed tag triggers an event task, the event task waits for all the data to arrive before the event task executes.
Specific condition or conditions occur within the logic of a program	EVENT instruction	Multiple EVENT instructions can trigger the same task. This lets you execute a task from different programs.

MANAGE EVENT TASKS

Module Input Data State Change Trigger



Type: Event

Trigger: Module Input Data State Change

Tag: Local:4:I

□ Execute Task If No Event Occurs Within 1000.000 ms

Priority: 1 (Lower Number Yields Higher Priority)

Watchdog: 500.000 ms

☑ Disable Automatic Output Processing To Reduce Task Overhead

Let an event trigger this task. →

Let data from an input module trigger the task. →

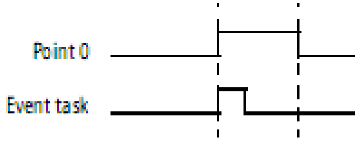
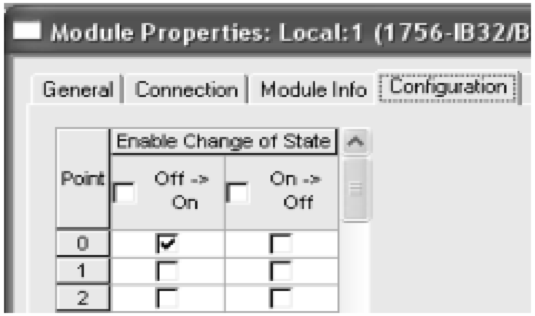
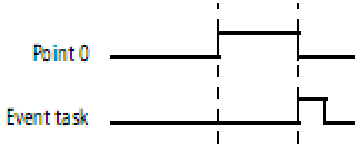
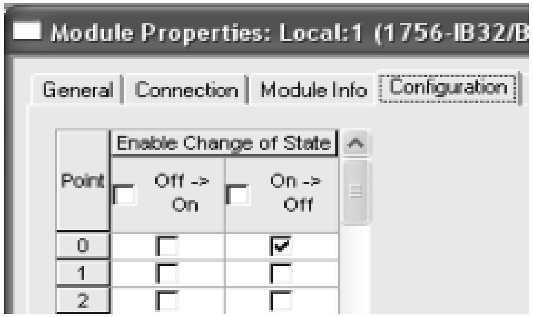
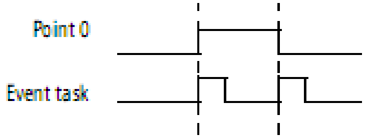
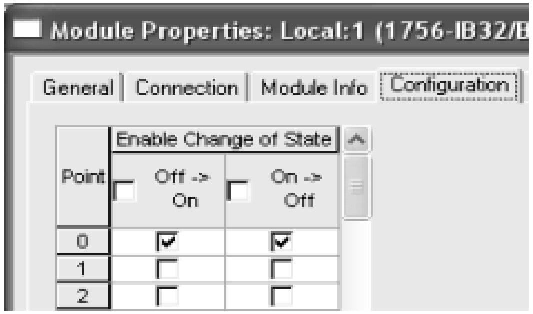
Let this input tag trigger the task. →

When the task is done, do not update digital outputs in the local chassis. →

Event Task is trigged whenever data from input change

MANAGE EVENT TASKS

Choose Trigger for Module Input State

If you want this	Then configure the input module like this (Point 0 is an example)
	Change of State No Change of State for Remaining Points  Module Properties: Local:1 (1756-IB32/B General | Connection | Module Info | Configuration Point | Enable Change of State: □ Off -> On | □ On -> Off 0 | ☑ | □ 1 | □ | □ 2 | □ | □
	Change of State No Change of State for Remaining Points  Module Properties: Local:1 (1756-IB32/B General | Connection | Module Info | Configuration Point | Enable Change of State: □ Off -> On | □ On -> Off 0 | □ | ☑ 1 | □ | □ 2 | □ | □
	Change of State No Change of State for Remaining Points  Module Properties: Local:1 (1756-IB32/B General | Connection | Module Info | Configuration Point | Enable Change of State: □ Off -> On | □ On -> Off 0 | ☑ | ☑ 1 | □ | □ 2 | □ | □

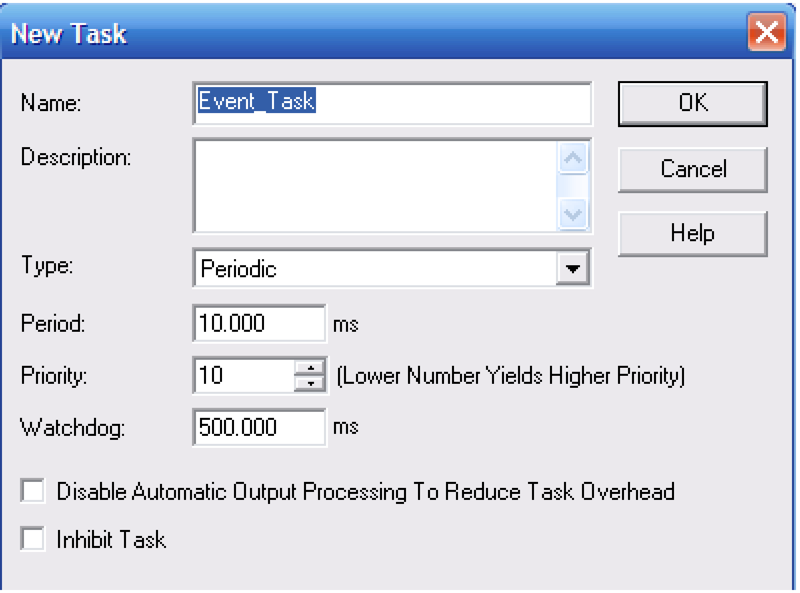
Event Task is trigged whenever data from input change

PROGRAM FOR EVENT TASKS

Create a ***Event Task***, enter an appropriate name, Select Task Type, event

And Priority, create a Program and write a logic program

- ➢Create a **Event Task**, enter an appropriate **name, Type of Task , Trigger and Priority**



New Task

Name: Event_Task

Description:

Type: Periodic

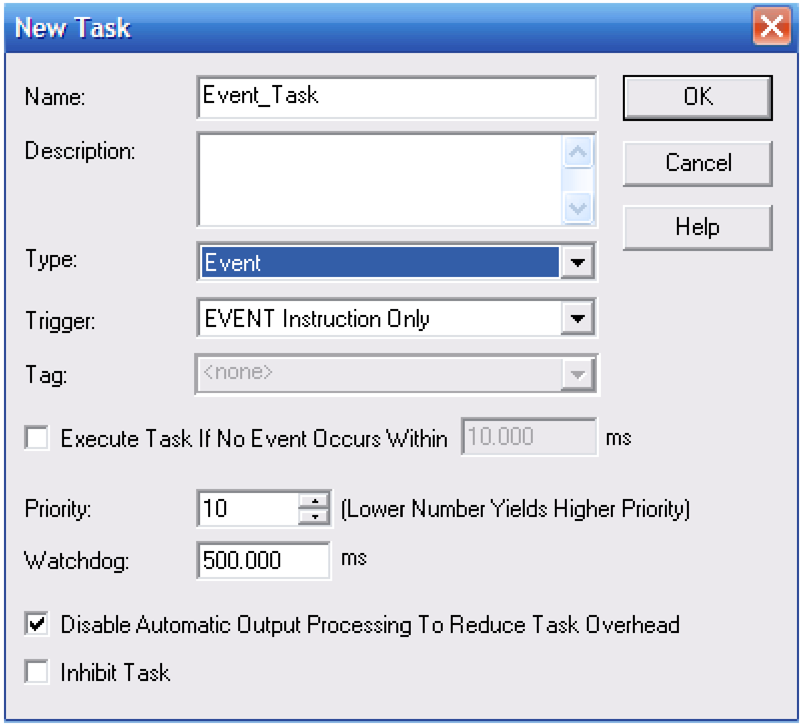
Period: 10.000 ms

Priority: 10 (Lower Number Yields Higher Priority)

Watchdog: 500.000 ms

□ Disable Automatic Output Processing To Reduce Task Overhead

□ Inhibit Task



New Task

Name: Event_Task

Description:

Type: Event

Trigger: EVENT Instruction Only

Tag: <none>

□ Execute Task If No Event Occurs Within 10.000 ms

Priority: 10 (Lower Number Yields Higher Priority)

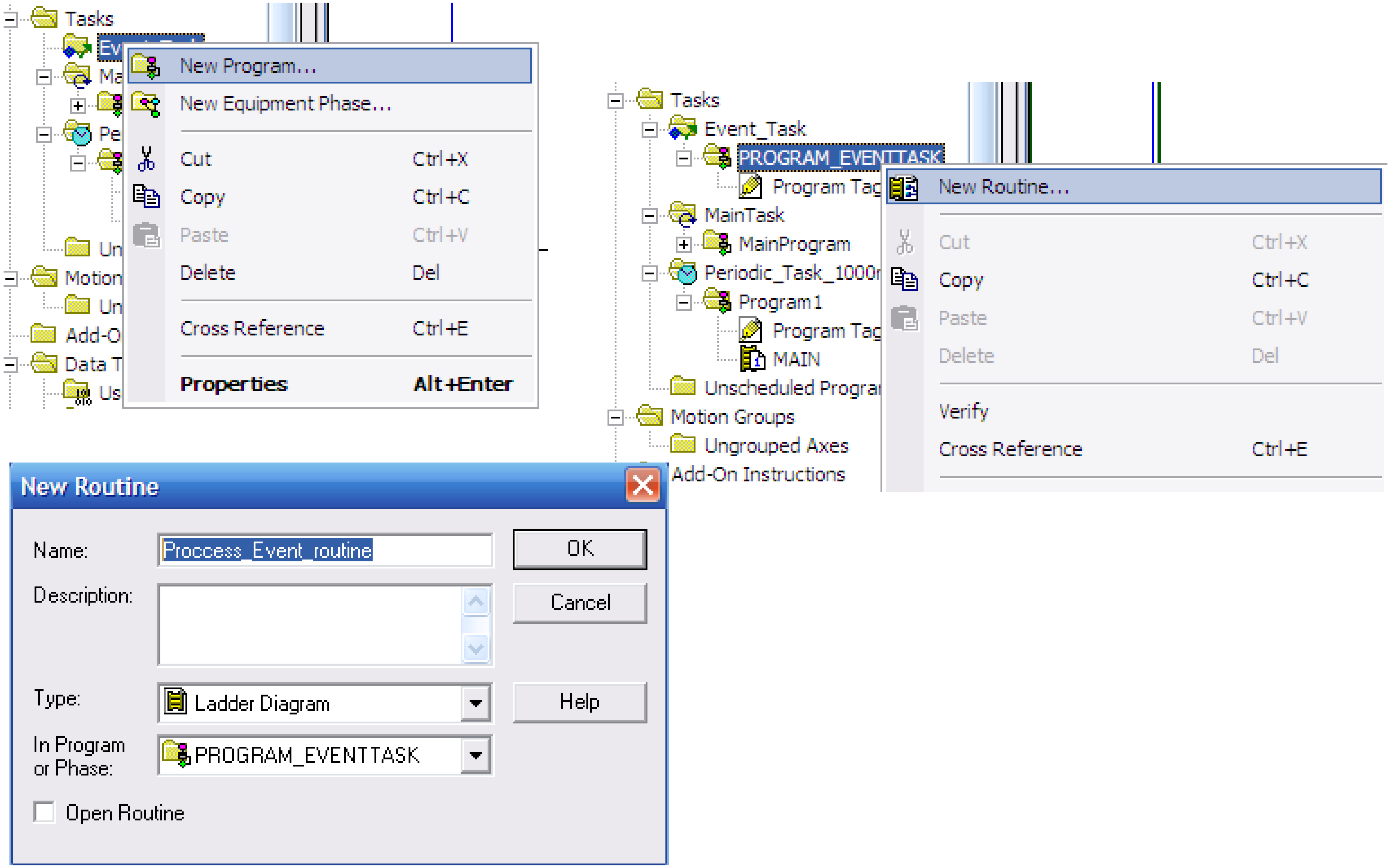
Watchdog: 500.000 ms

☑ Disable Automatic Output Processing To Reduce Task Overhead

□ Inhibit Task

PROGRAM FOR EVENT TASKS

- Create a new Program with appropriate name and a new routine



New Routine

Name: Proccess_Event_routine

Description:

Type: Ladder Diagram

In Program or Phase: PROGRAM_EVENTTASK

☐ Open Routine

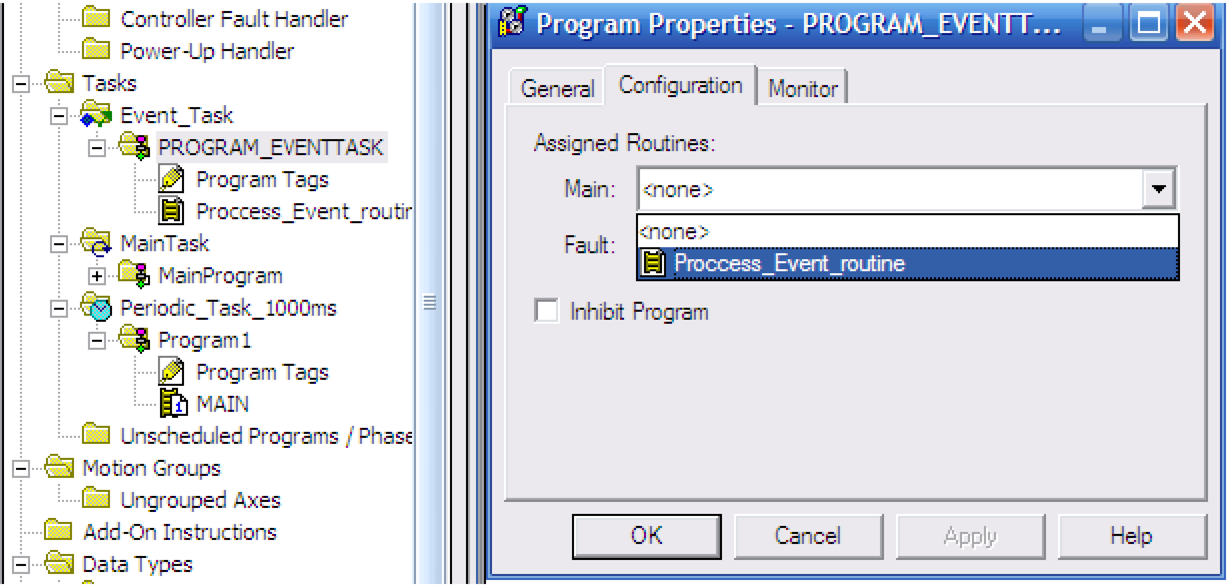
OK

Cancel

Help

PROGRAM FOR EVENT TASKS

- Select Main Routine in Event Task to write logic program



Controller Fault Handler
Power-Up Handler
Tasks
Event_Task
PROGRAM_EVENTTASK
Program Tags
Proccess_Event_routir
MainTask
MainProgram
Periodic_Task_1000ms
Program1
Program Tags
MAIN
Unscheduled Programs / Phase
Motion Groups
Ungrouped Axes
Add-On Instructions
Data Types

Program Properties - PROGRAM_EVENTT...

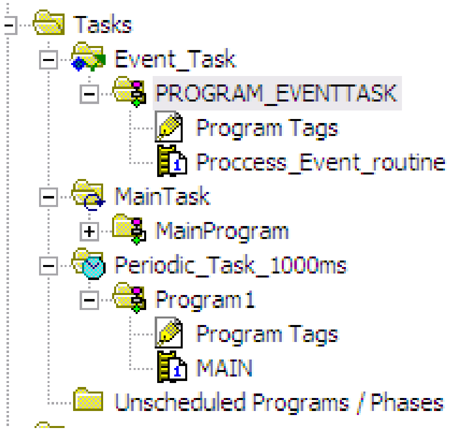
General | Configuration | Monitor

Assigned Routines:

Main: <none>

Fault: <none>
Proccess_Event_routine

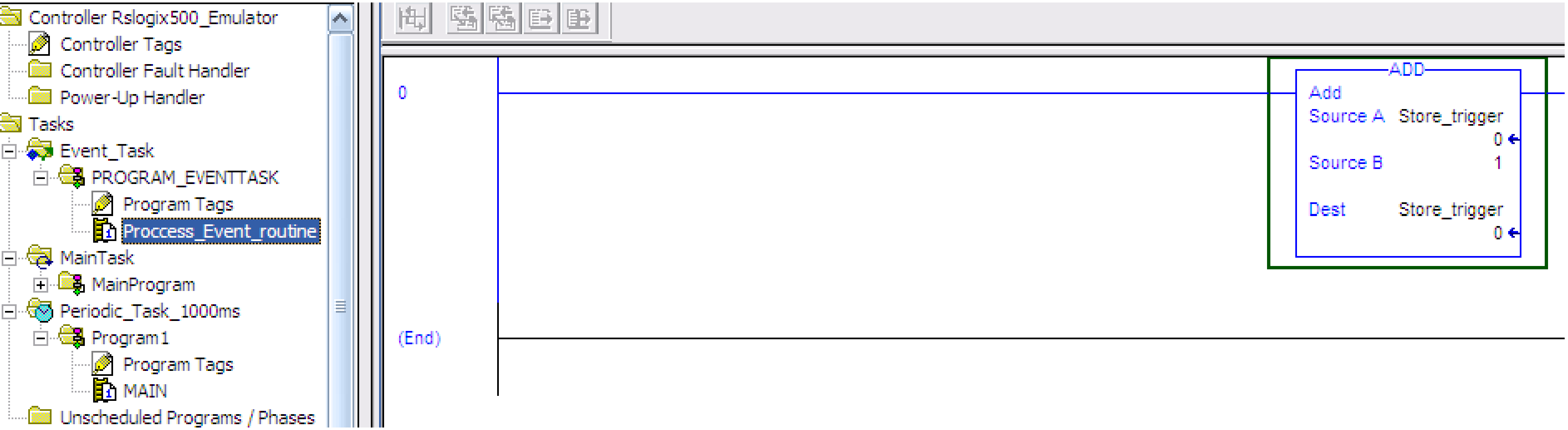
□ Inhibit Program



Tasks
Event_Task
PROGRAM_EVENTTASK
Program Tags
Proccess_Event_routine
MainTask
MainProgram
Periodic_Task_1000ms
Program1
Program Tags
MAIN
Unscheduled Programs / Phases

PROGRAM FOR EVENT TASKS

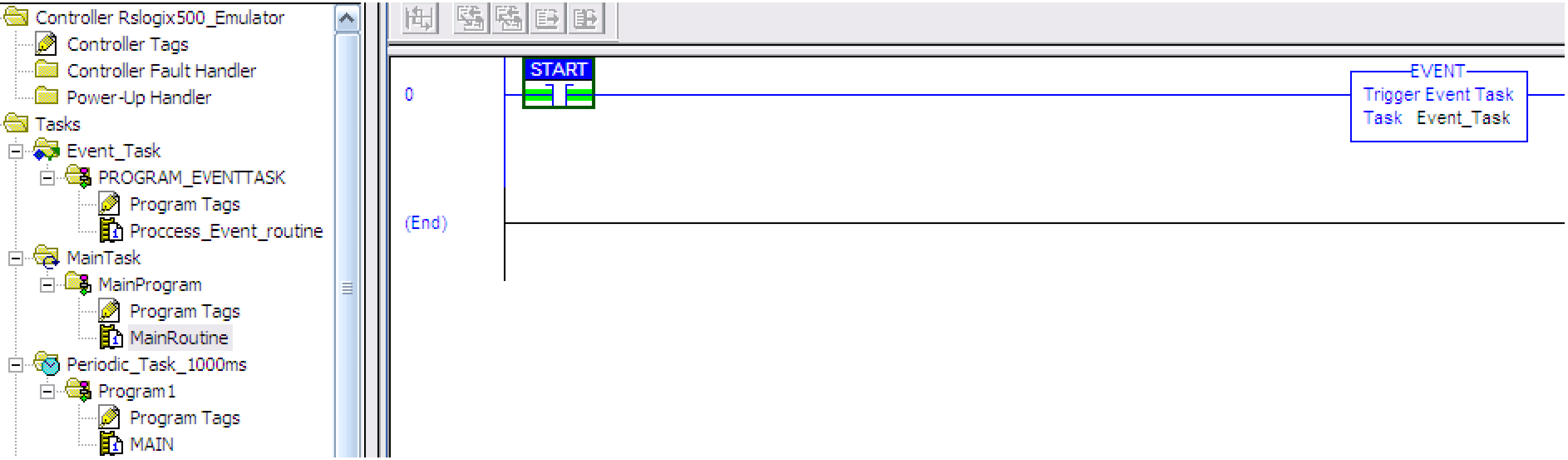
- Select Main Routine in Event Task to write a Program



Add Instruction will executed whenever Event Task is Called

PROGRAM FOR EVENT TASKS

Use Trigger Event Instruction to call Event_Task



Trigger Task Instruction is placed in another Task.

MINOR AND MAJOR FAULT

Minor Fault: CPU does not go in stop mode with fault

- Periodic Task overlap.
- Load from nonvolatile memory.
- Problem with serial port.
- Low battery…..

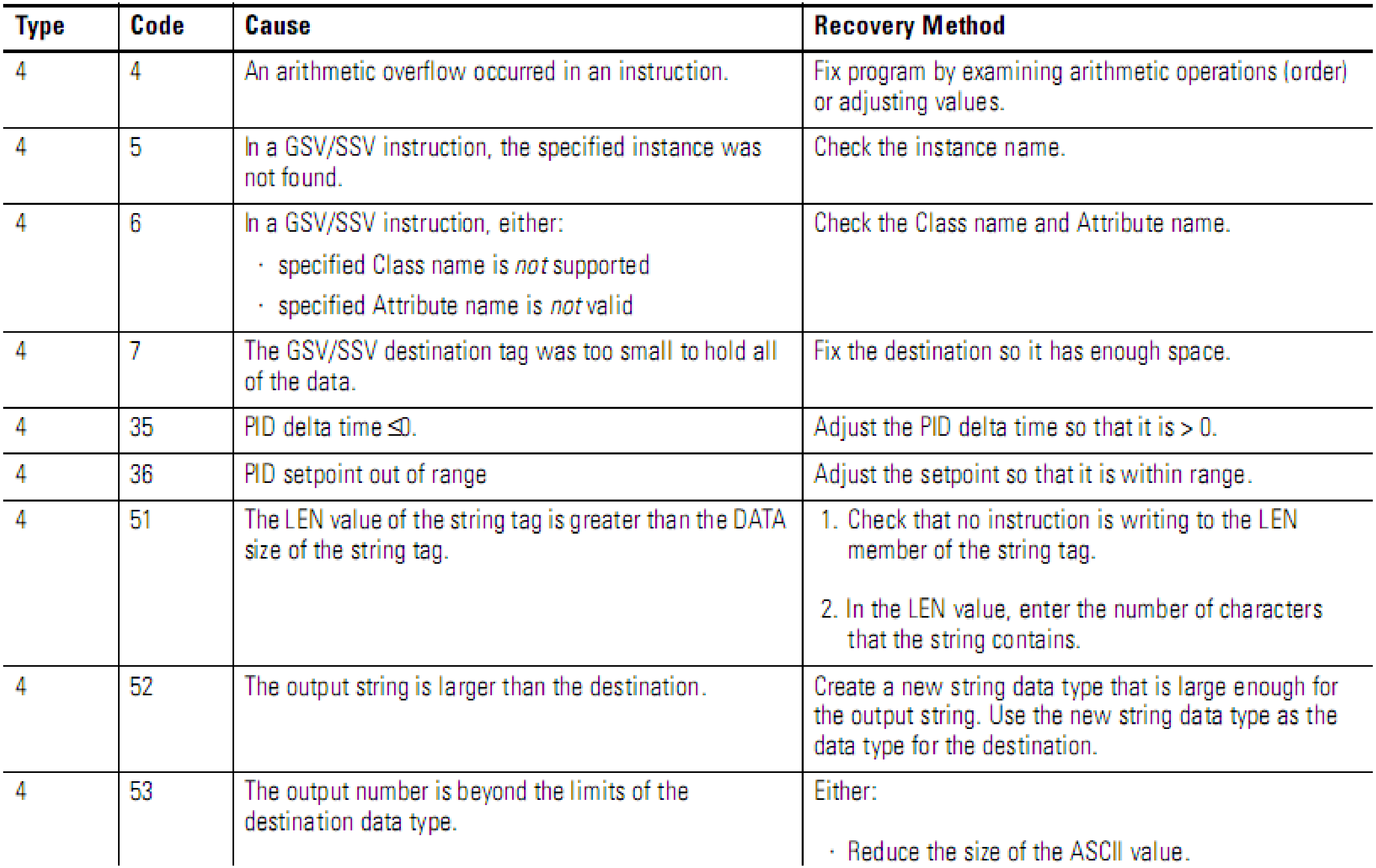
Major Fault: CPU goes in stop mode with fault

The CPU powered on in run mode.

A required I/O module connection failed.

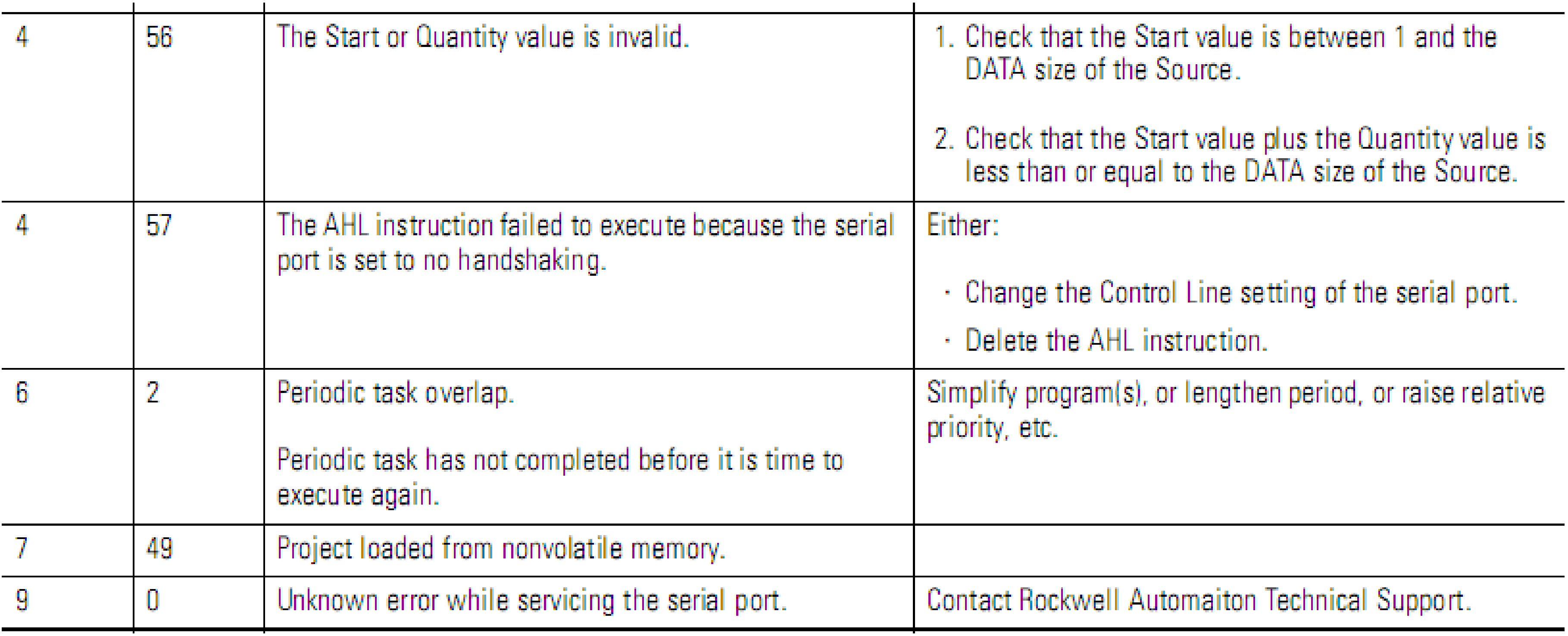
Configuration fault occurred…..

MINOR FAULT CODES



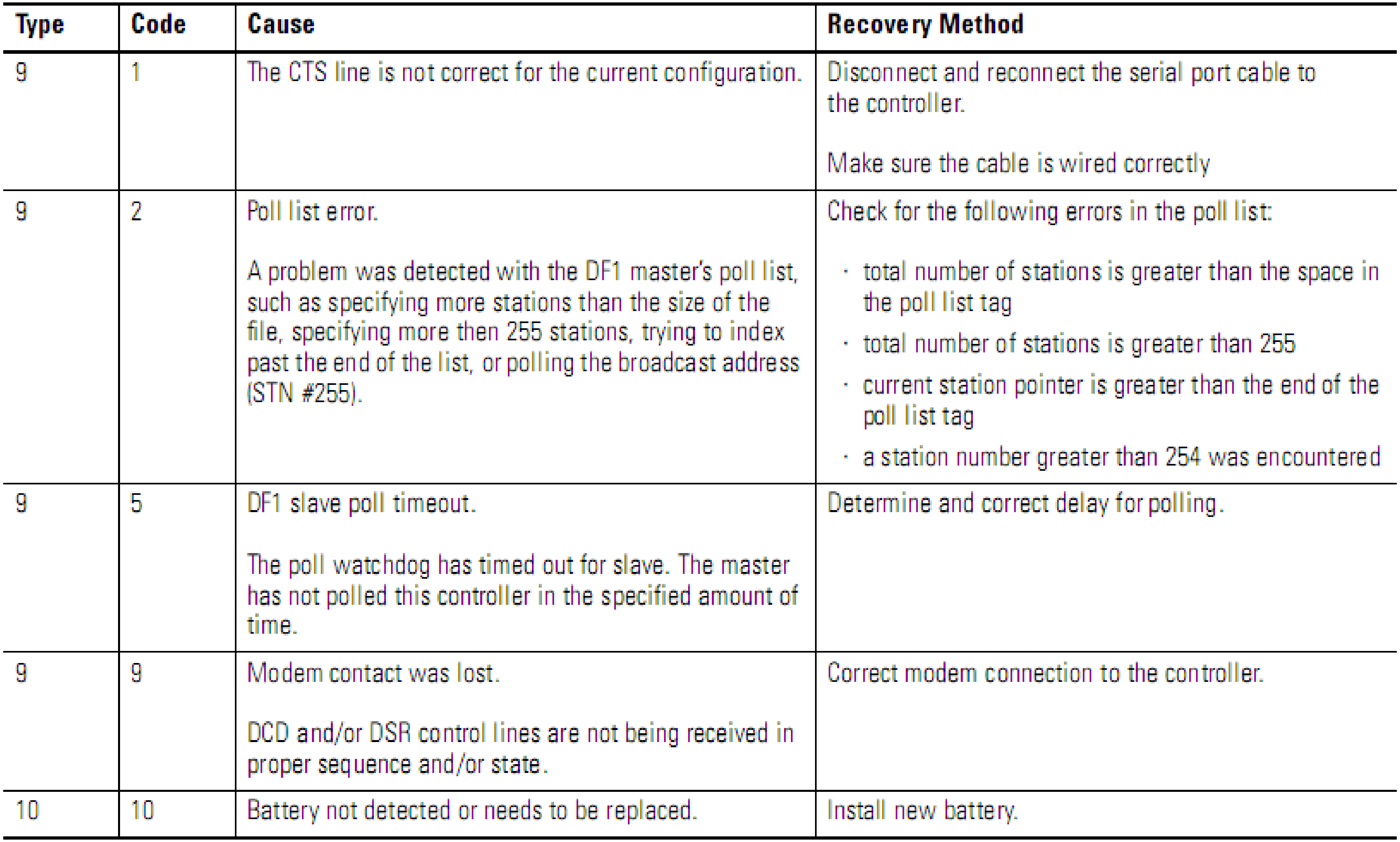
Type	Code	Cause	Recovery Method
4	4	An arithmetic overflow occurred in an instruction.	Fix program by examining arithmetic operations (order) or adjusting values.
4	5	In a GSV/SSV instruction, the specified instance was not found.	Check the instance name.
4	6	In a GSV/SSV instruction, either: · specified Class name is *not* supported · specified Attribute name is *not* valid	Check the Class name and Attribute name.
4	7	The GSV/SSV destination tag was too small to hold all of the data.	Fix the destination so it has enough space.
4	35	PID delta time ≤0.	Adjust the PID delta time so that it is > 0.
4	36	PID setpoint out of range	Adjust the setpoint so that it is within range.
4	51	The LEN value of the string tag is greater than the DATA size of the string tag.	1. Check that no instruction is writing to the LEN member of the string tag. 2. In the LEN value, enter the number of characters that the string contains.
4	52	The output string is larger than the destination.	Create a new string data type that is large enough for the output string. Use the new string data type as the data type for the destination.
4	53	The output number is beyond the limits of the destination data type.	Either: · Reduce the size of the ASCII value.

MINOR FAULT CODES



4	56	The Start or Quantity value is invalid.	1. Check that the Start value is between 1 and the DATA size of the Source. 2. Check that the Start value plus the Quantity value is less than or equal to the DATA size of the Source.
4	57	The AHL instruction failed to execute because the serial port is set to no handshaking.	Either: · Change the Control Line setting of the serial port. · Delete the AHL instruction.
6	2	Periodic task overlap. Periodic task has not completed before it is time to execute again.	Simplify program(s), or lengthen period, or raise relative priority, etc.
7	49	Project loaded from nonvolatile memory.	
9	0	Unknown error while servicing the serial port.	Contact Rockwell Automaiton Technical Support.

MINOR FAULT CODES



Type	Code	Cause	Recovery Method
9	1	The CTS line is not correct for the current configuration.	Disconnect and reconnect the serial port cable to the controller. Make sure the cable is wired correctly
9	2	Poll list error. A problem was detected with the DF1 master's poll list, such as specifying more stations than the size of the file, specifying more then 255 stations, trying to index past the end of the list, or polling the broadcast address (STN #255).	Check for the following errors in the poll list: · total number of stations is greater than the space in the poll list tag · total number of stations is greater than 255 · current station pointer is greater than the end of the poll list tag · a station number greater than 254 was encountered
9	5	DF1 slave poll timeout. The poll watchdog has timed out for slave. The master has not polled this controller in the specified amount of time.	Determine and correct delay for polling.
9	9	Modem contact was lost. DCD and/or DSR control lines are not being received in proper sequence and/or state.	Correct modem connection to the controller.
10	10	Battery not detected or needs to be replaced.	Install new battery.

MINOR FAULT CODES

Handle Minor Fault

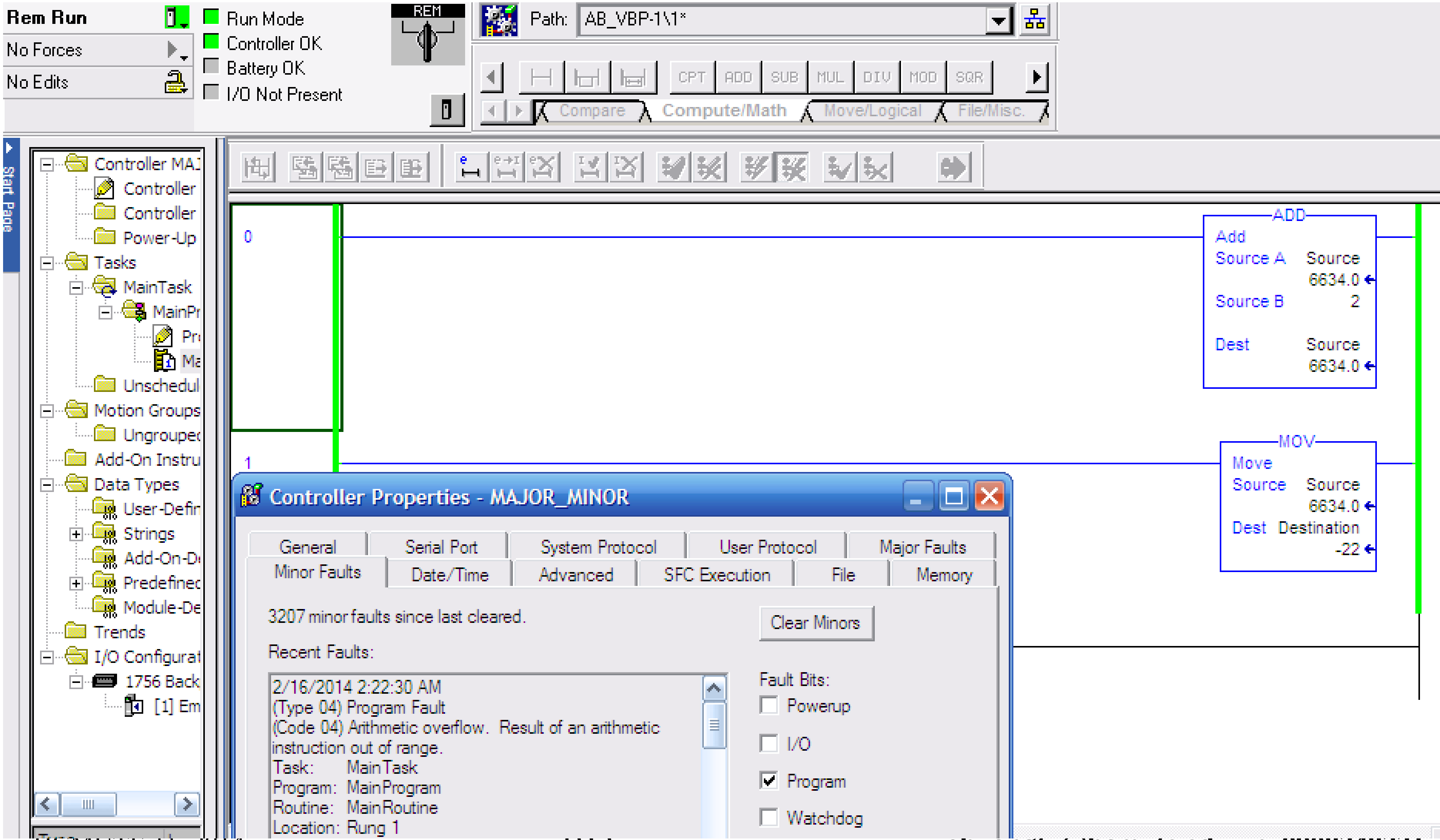
EX: Arithmetic overflow, result of arithmetic instruction is out of range(Type =4, code =4)

- Create a tag, named **source** with **real** type and another named **Destination** with **integer type** .
- Write an instruction to increase data of source tag.
- Write an instruction to move data from **Source tag** to **Destination tag**
- Download program to the CPU, run CPU
- Slect the CPU/ Properties and minor fault to view **Type** and **Code**.

MINOR FAULT CODES

Ex: Arithmetic overflow, result of arithmetic instruction is out of range(

Type =4, code =4)



3207 minor faults since last cleared.

Recent Faults:

2/16/2014 2:22:30 AM
(Type 04) Program Fault
(Code 04) Arithmetic overflow. Result of an arithmetic instruction out of range.
Task: MainTask
Program: MainProgram
Routine: MainRoutine
Location: Rung 1

Fault Bits:
□ Powerup
□ I/O
☑ Program
□ Watchdog

MINOR FAULT CODES

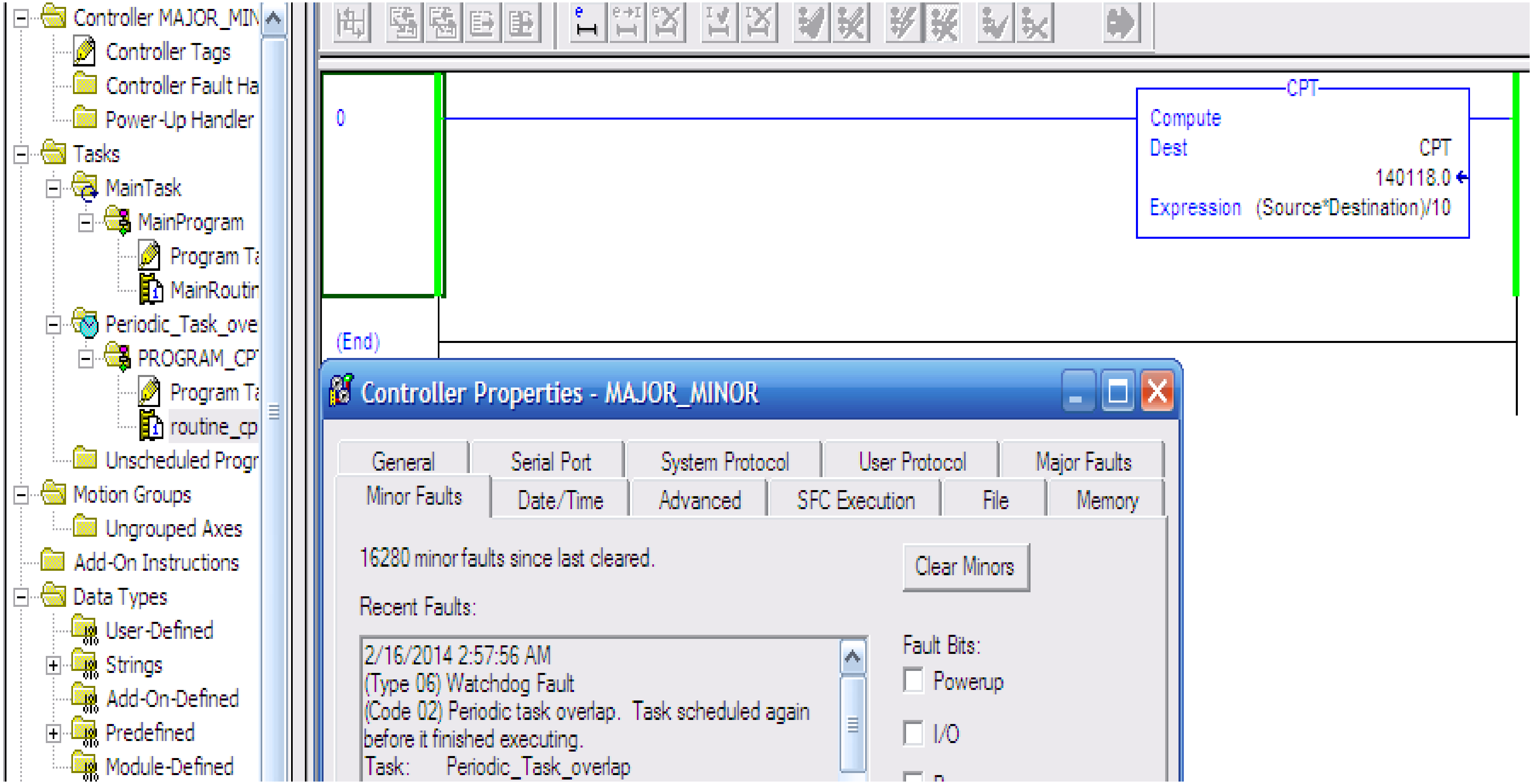
Monitor Minor Fault

EX: Periodic task overlap, Task scheduled again before it finished executing(Type =6, code =2)

- Create a tag, named **CPT** with data type is real, two tag named **Source**(real) and **Destination**(Sint).
- Create a Periodic Task with period 1ms and a routine
- Use CPT instruction to multi **Source tag** and **Destination tag,** the result is placed in **CPT** tag.
- Download program to the CPU, run CPU
- Slect the CPU/ Properties and minor fault tab to view **Type** and **Code**.

MINOR FAULT CODES

EX: Periodic task overlap, Task scheduled again before it finished executing(Type =6, code =2)



16280 minor faults since last cleared.

Recent Faults:

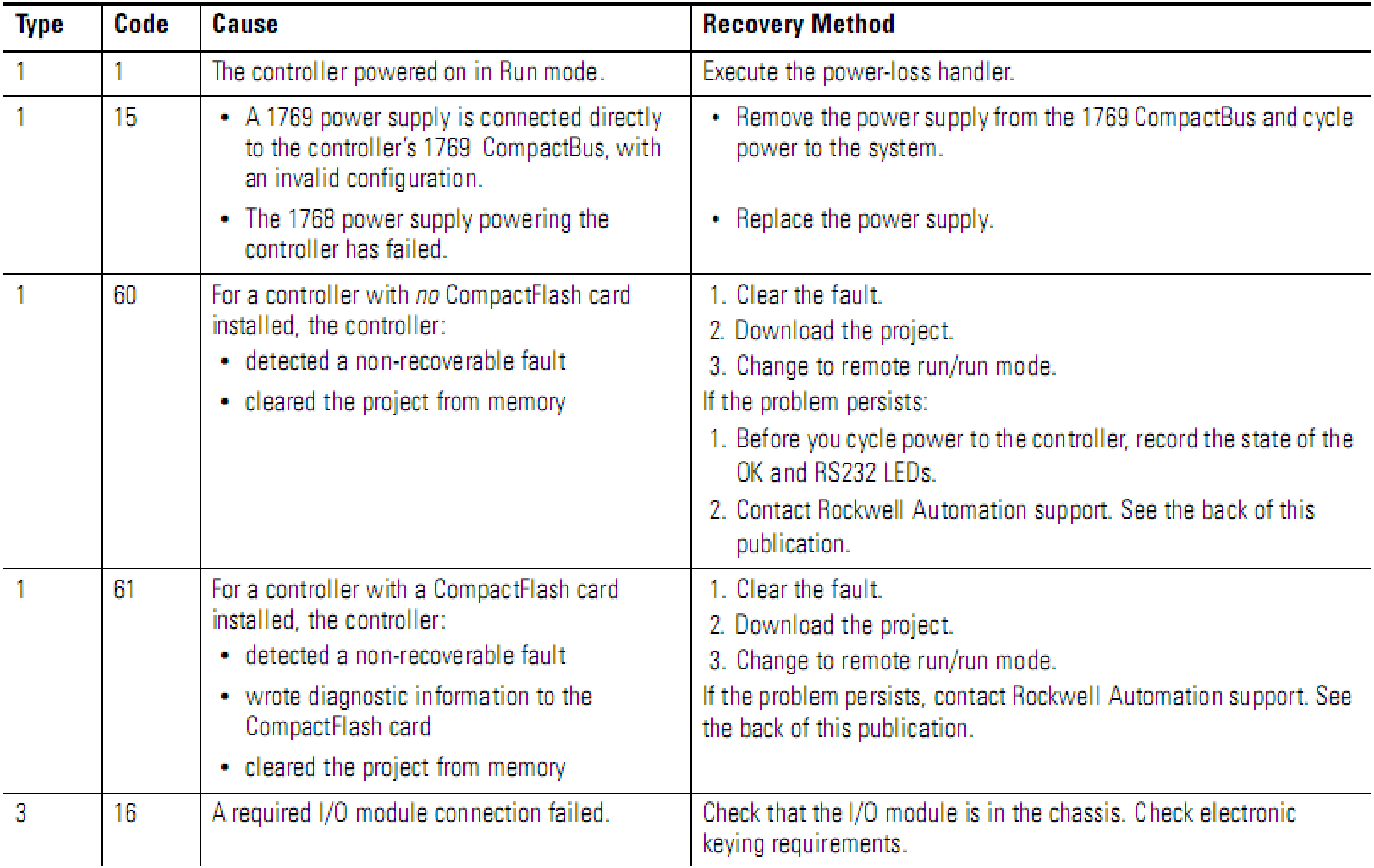
```
2/16/2014 2:57:56 AM
(Type 06) Watchdog Fault
(Code 02) Periodic task overlap.  Task scheduled again
before it finished executing.
Task:     Periodic_Task_overlap
```

Fault Bits:

□ Powerup

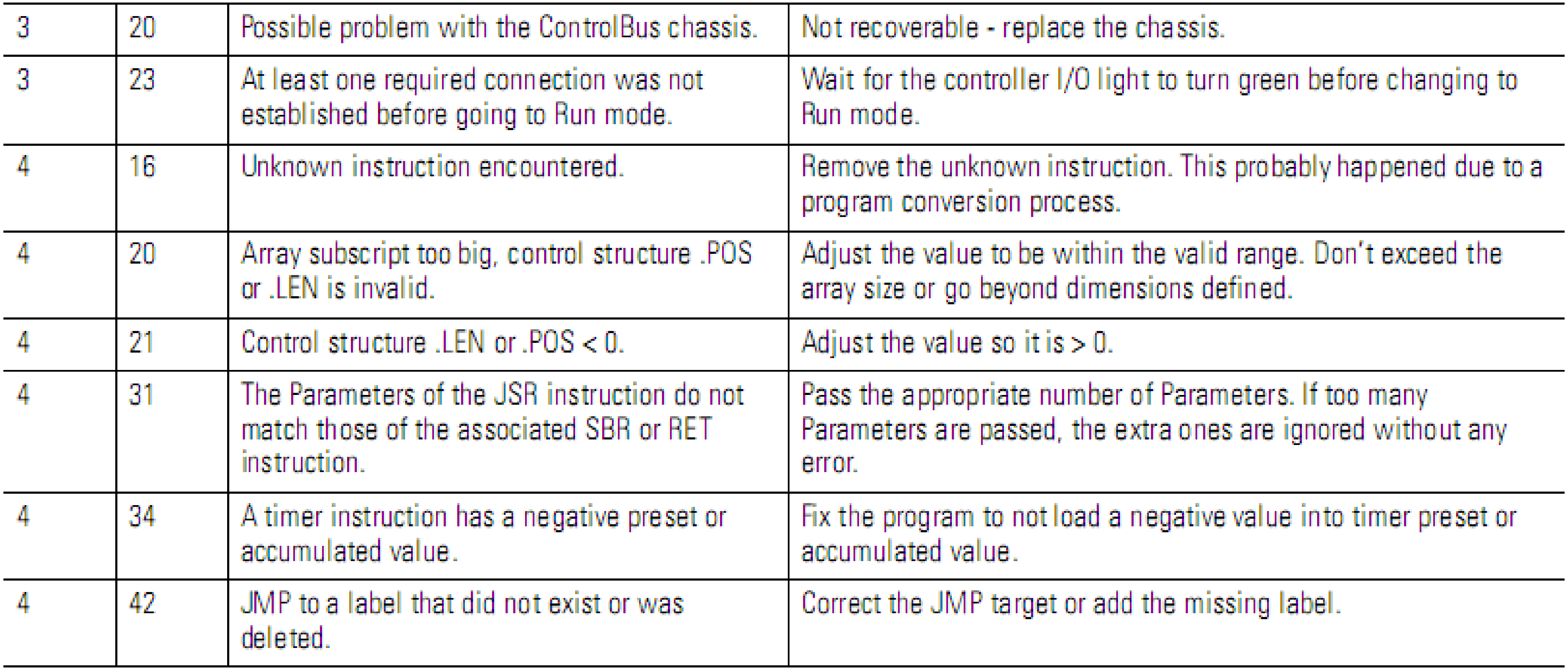
□ I/O

MAJOR FAULT CODES



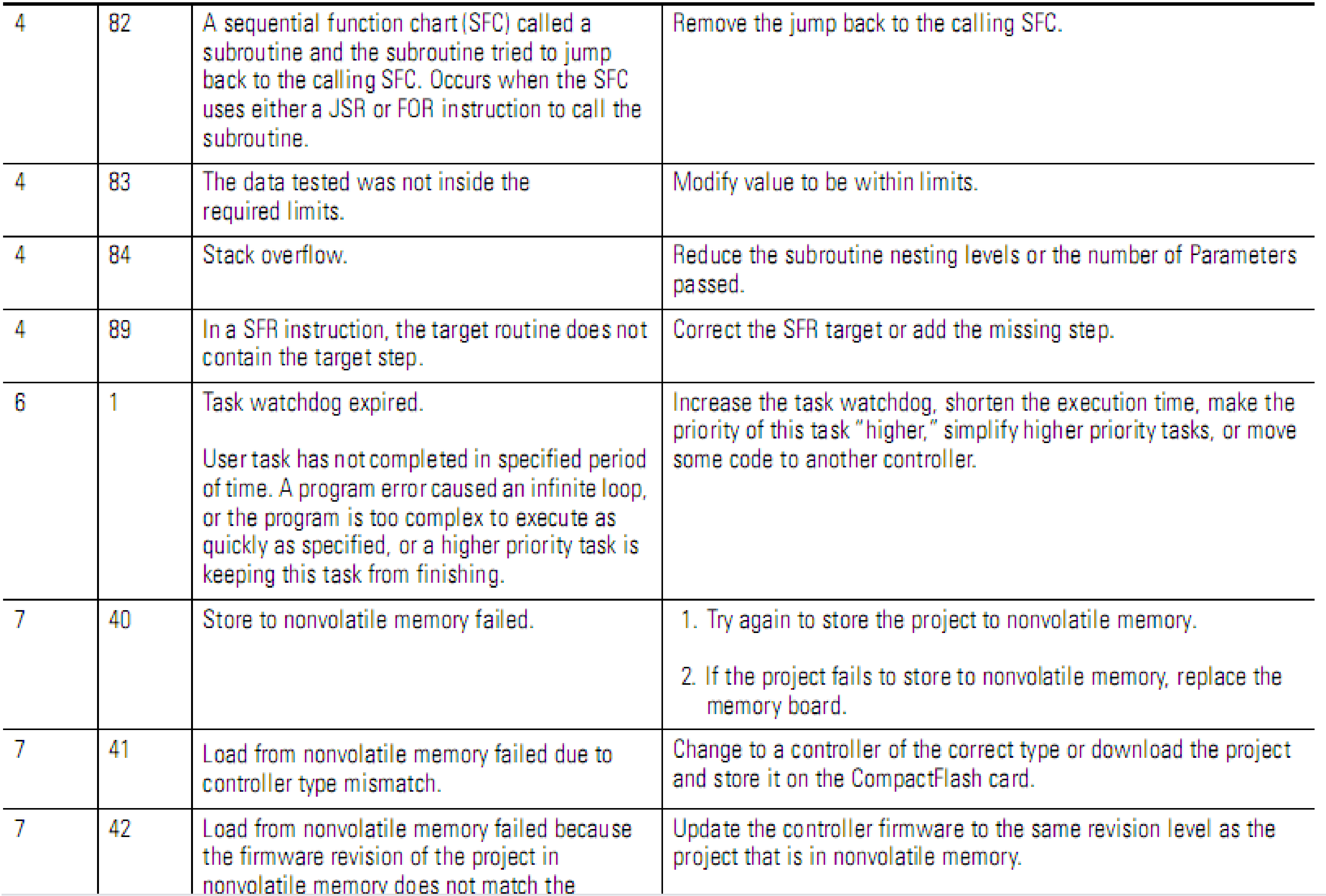
Type	Code	Cause	Recovery Method
1	1	The controller powered on in Run mode.	Execute the power-loss handler.
1	15	• A 1769 power supply is connected directly to the controller's 1769 CompactBus, with an invalid configuration. • The 1768 power supply powering the controller has failed.	• Remove the power supply from the 1769 CompactBus and cycle power to the system. • Replace the power supply.
1	60	For a controller with *no* CompactFlash card installed, the controller: • detected a non-recoverable fault • cleared the project from memory	1. Clear the fault. 2. Download the project. 3. Change to remote run/run mode. If the problem persists: 1. Before you cycle power to the controller, record the state of the OK and RS232 LEDs. 2. Contact Rockwell Automation support. See the back of this publication.
1	61	For a controller with a CompactFlash card installed, the controller: • detected a non-recoverable fault • wrote diagnostic information to the CompactFlash card • cleared the project from memory	1. Clear the fault. 2. Download the project. 3. Change to remote run/run mode. If the problem persists, contact Rockwell Automation support. See the back of this publication.
3	16	A required I/O module connection failed.	Check that the I/O module is in the chassis. Check electronic keying requirements.

MAJOR FAULT CODES



3	20	Possible problem with the ControlBus chassis.	Not recoverable - replace the chassis.
3	23	At least one required connection was not established before going to Run mode.	Wait for the controller I/O light to turn green before changing to Run mode.
4	16	Unknown instruction encountered.	Remove the unknown instruction. This probably happened due to a program conversion process.
4	20	Array subscript too big, control structure .POS or .LEN is invalid.	Adjust the value to be within the valid range. Don't exceed the array size or go beyond dimensions defined.
4	21	Control structure .LEN or .POS < 0.	Adjust the value so it is > 0.
4	31	The Parameters of the JSR instruction do not match those of the associated SBR or RET instruction.	Pass the appropriate number of Parameters. If too many Parameters are passed, the extra ones are ignored without any error.
4	34	A timer instruction has a negative preset or accumulated value.	Fix the program to not load a negative value into timer preset or accumulated value.
4	42	JMP to a label that did not exist or was deleted.	Correct the JMP target or add the missing label.

MAJOR FAULT CODES

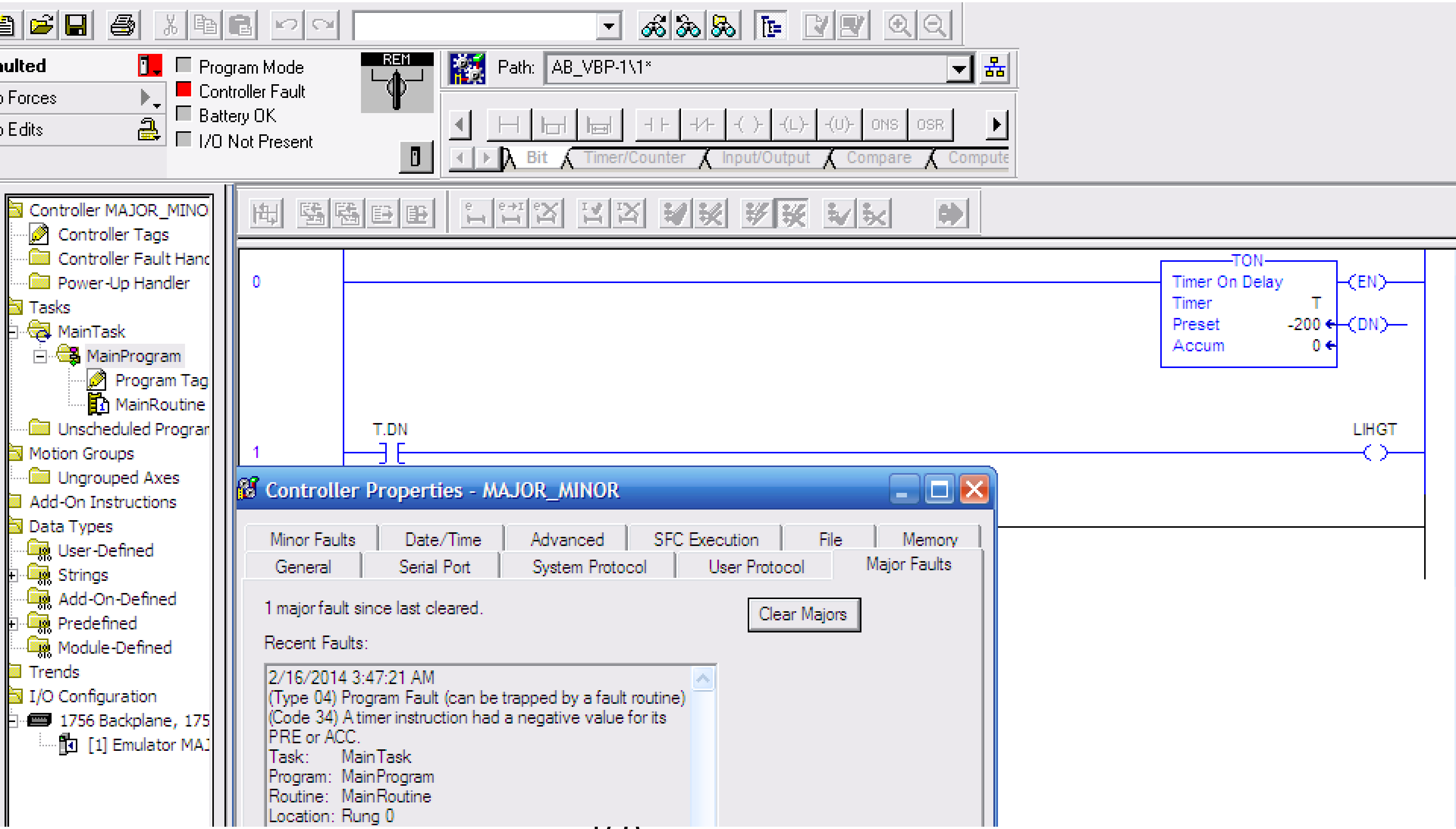


4	82	A sequential function chart (SFC) called a subroutine and the subroutine tried to jump back to the calling SFC. Occurs when the SFC uses either a JSR or FOR instruction to call the subroutine.	Remove the jump back to the calling SFC.
4	83	The data tested was not inside the required limits.	Modify value to be within limits.
4	84	Stack overflow.	Reduce the subroutine nesting levels or the number of Parameters passed.
4	89	In a SFR instruction, the target routine does not contain the target step.	Correct the SFR target or add the missing step.
6	1	Task watchdog expired. User task has not completed in specified period of time. A program error caused an infinite loop, or the program is too complex to execute as quickly as specified, or a higher priority task is keeping this task from finishing.	Increase the task watchdog, shorten the execution time, make the priority of this task "higher," simplify higher priority tasks, or move some code to another controller.
7	40	Store to nonvolatile memory failed.	1. Try again to store the project to nonvolatile memory. 2. If the project fails to store to nonvolatile memory, replace the memory board.
7	41	Load from nonvolatile memory failed due to controller type mismatch.	Change to a controller of the correct type or download the project and store it on the CompactFlash card.
7	42	Load from nonvolatile memory failed because the firmware revision of the project in nonvolatile memory does not match the	Update the controller firmware to the same revision level as the project that is in nonvolatile memory.

MAJOR FAULT CODES

Example about Major Fault: Timer with a negative value preset for its Pre

(Type =04, code =34)



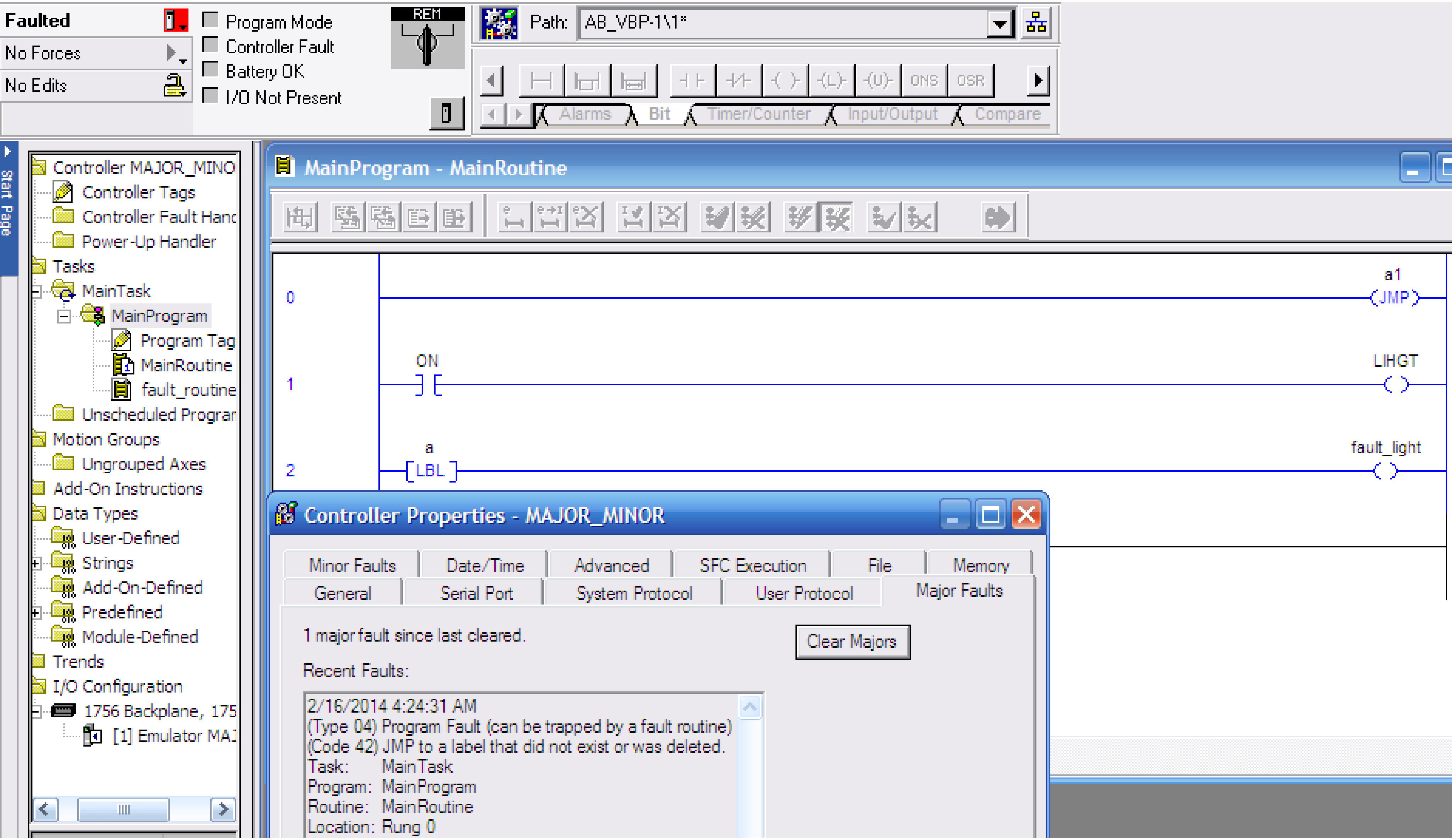
1 major fault since last cleared.

Recent Faults:

```
2/16/2014 3:47:21 AM
(Type 04) Program Fault (can be trapped by a fault routine)
(Code 34) A timer instruction had a negative value for its
PRE or ACC.
Task:      MainTask
Program:   MainProgram
Routine:   MainRoutine
Location:  Rung 0
```

MAJOR FAULT CODES

Example about Major Fault: JMP to a label that do not exits (Type =04, code =42)



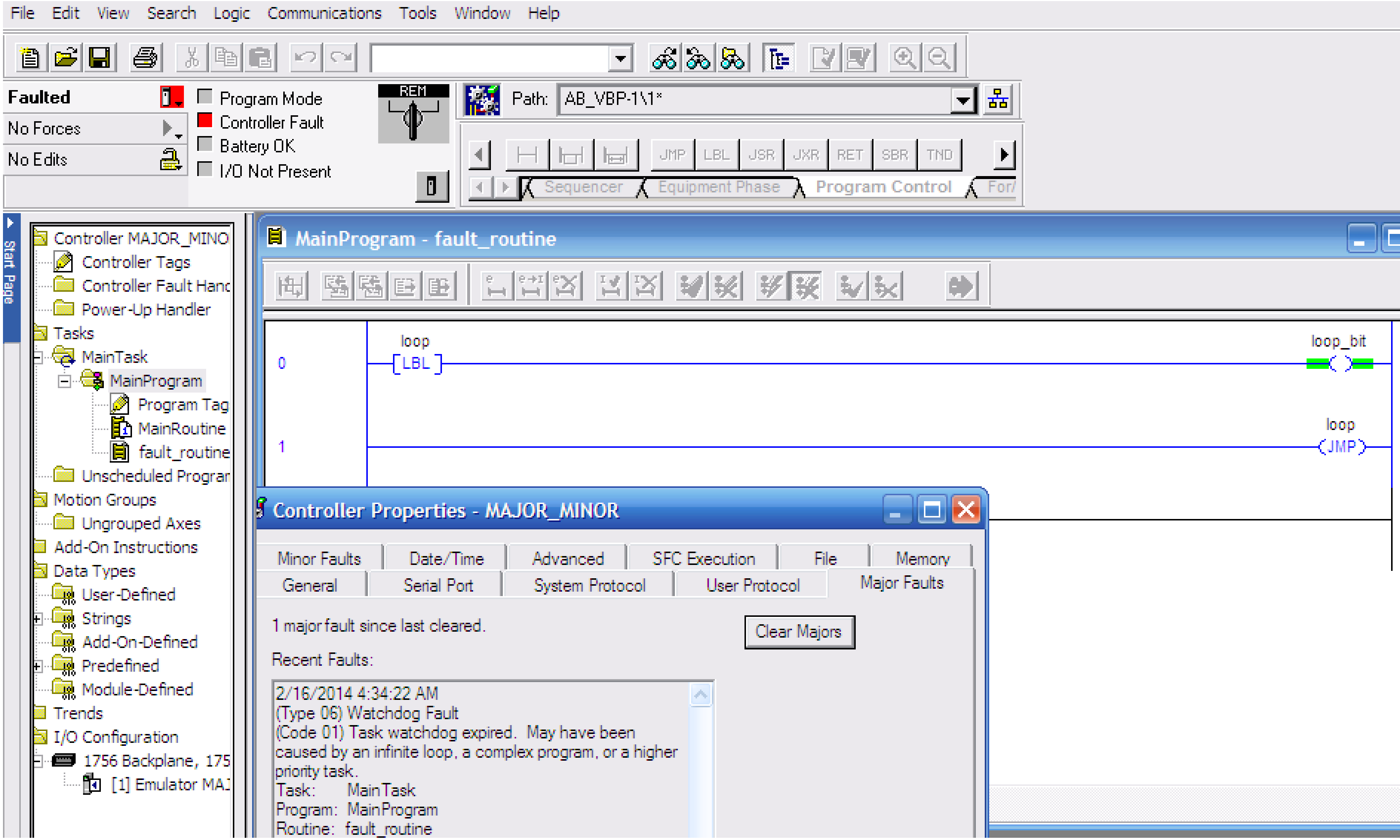
1 major fault since last cleared.

Recent Faults:

```
2/16/2014 4:24:31 AM
(Type 04) Program Fault (can be trapped by a fault routine)
(Code 42) JMP to a label that did not exist or was deleted.
Task:      MainTask
Program:   MainProgram
Routine:   MainRoutine
Location:  Rung 0
```

MAJOR FAULT CODES

Example about Major Fault: Task watchdog expired(Type =06, code =01)



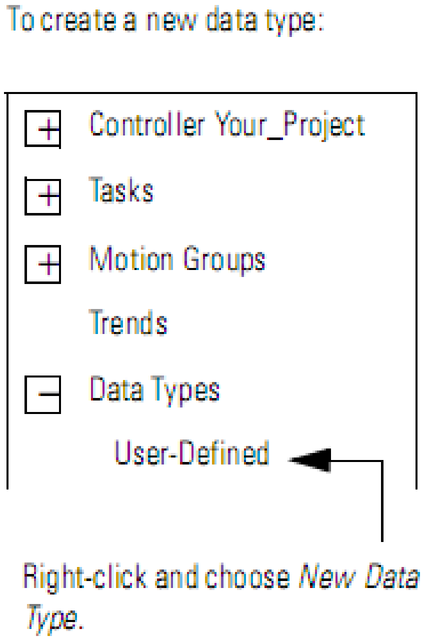
1 major fault since last cleared.

Recent Faults:

```
2/16/2014 4:34:22 AM
(Type 06) Watchdog Fault
(Code 01) Task watchdog expired.  May have been
caused by an infinite loop, a complex program, or a higher
priority task.
Task:      MainTask
Program:  MainProgram
Routine:  fault_routine
```

HANDLE FAULTs

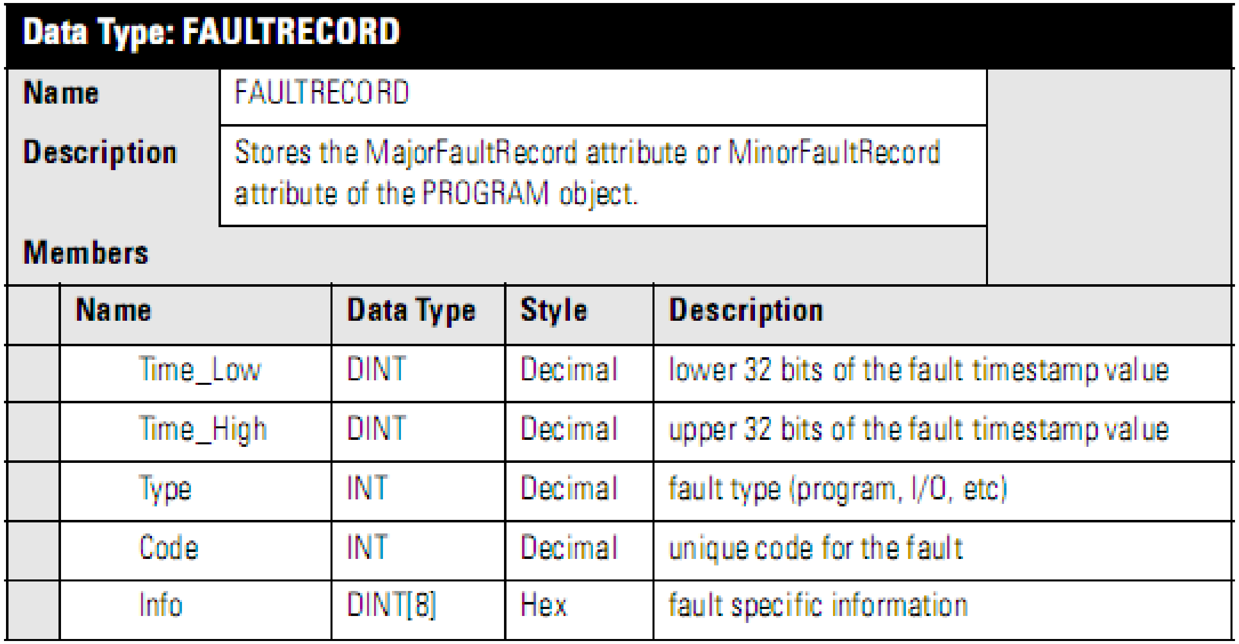
Create a Data Type to Store the fault information.



To create a new data type:

- ⊞ Controller Your_Project
- ⊞ Tasks
- ⊞ Motion Groups
- Trends
- ⊟ Data Types
  - User-Defined

Right-click and choose *New Data Type*.



Data Type: FAULTRECORD

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

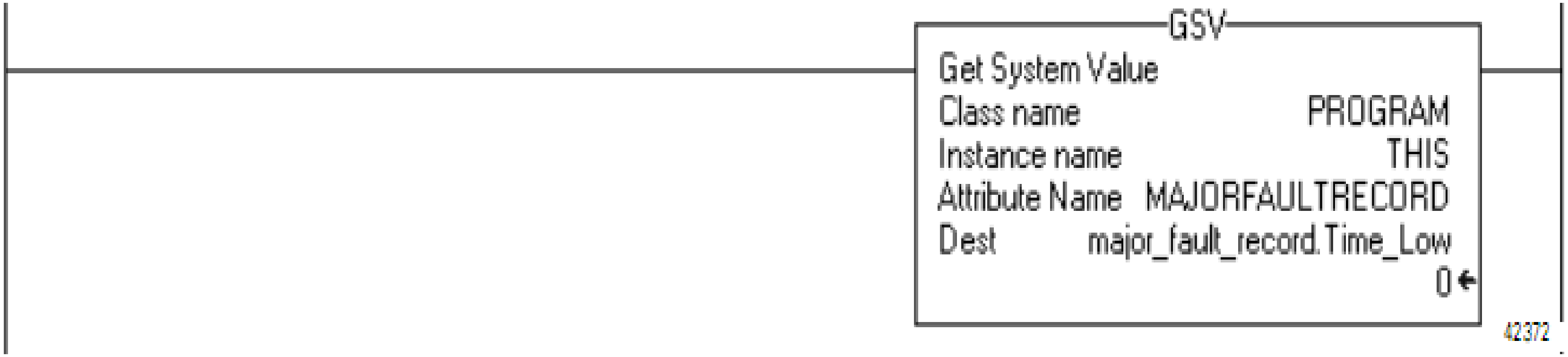
Members

Name	Data Type	Style	Description
Time_Low	DINT	Decimal	lower 32 bits of the fault timestamp value
Time_High	DINT	Decimal	upper 32 bits of the fault timestamp value
Type	INT	Decimal	fault type (program, I/O, etc)
Code	INT	Decimal	unique code for the fault
Info	DINT[8]	Hex	fault specific information

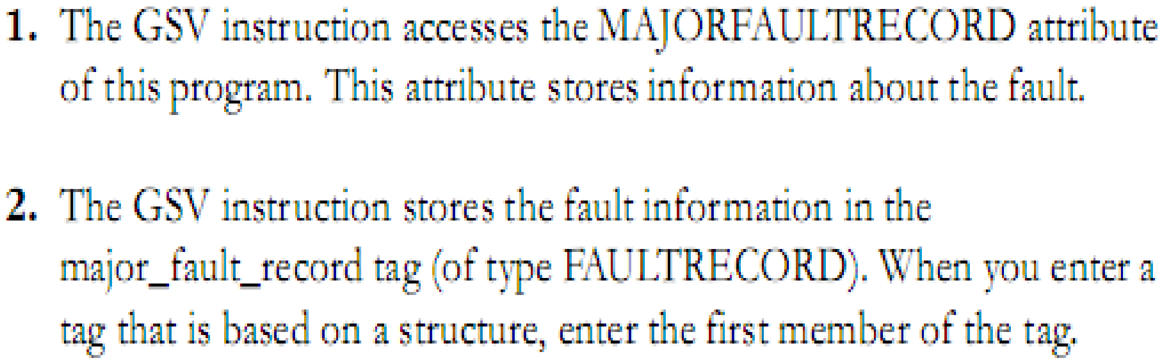
- To access system information, use GSV(Get System Value) and SSV(Set System Value) Instruction.
- For status information about a program, access the program Objects.
- For fault information, access these attribute of the program Object

HANDLE FAULTs

Get the fault Type and Code.



GSV
Get System Value
Class name PROGRAM
Instance name THIS
Attribute Name MAJORFAULTRECORD
Dest major_fault_record.Time_Low
0 ←

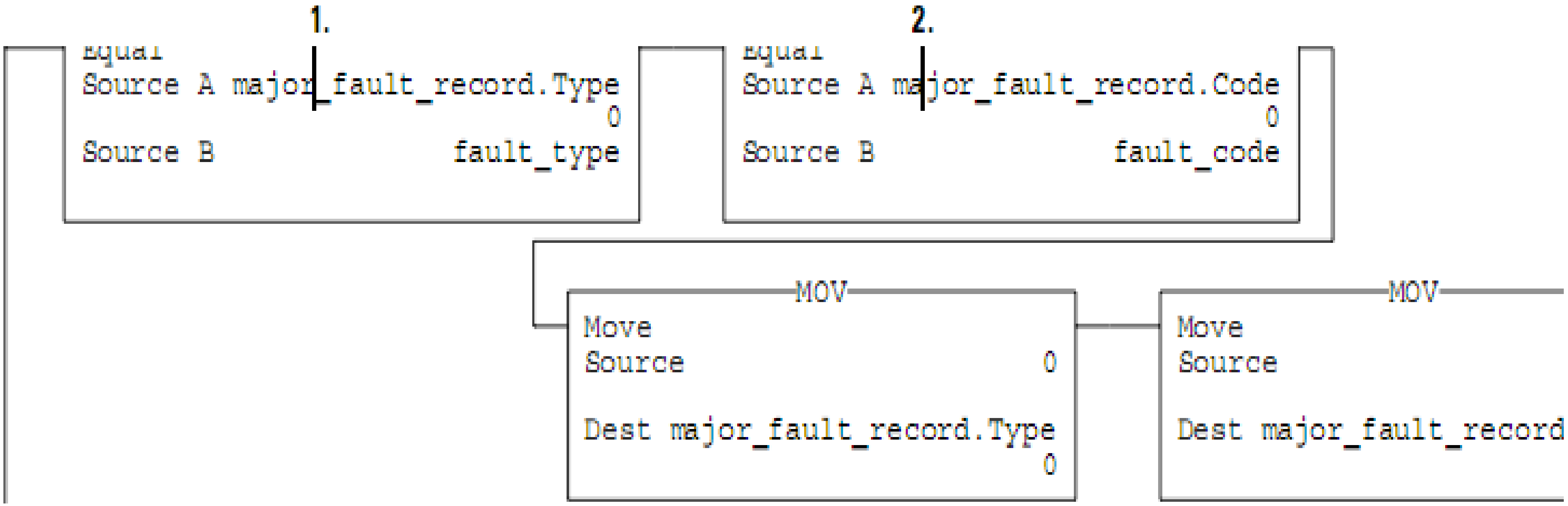


1. The GSV instruction accesses the MAJORFAULTRECORD attribute of this program. This attribute stores information about the fault.

2. The GSV instruction stores the fault information in the major_fault_record tag (of type FAULTRECORD). When you enter a tag that is based on a structure, enter the first member of the tag.

HANDLE FAULTs

Check for Specific fault.



1.

Equal
Source A major_fault_record.Type
0
Source B fault_type

2.

Equal
Source A major_fault_record.Code
0
Source B fault_code

MOV
Move
Source 0
Dest major_fault_record.Type
0

MOV
Move
Source
Dest major_fault_record

1. The first EQU instruction checks for a specific type of fault, such as program, I/O. In Source B, enter the value for the type of fault that you want to clear.

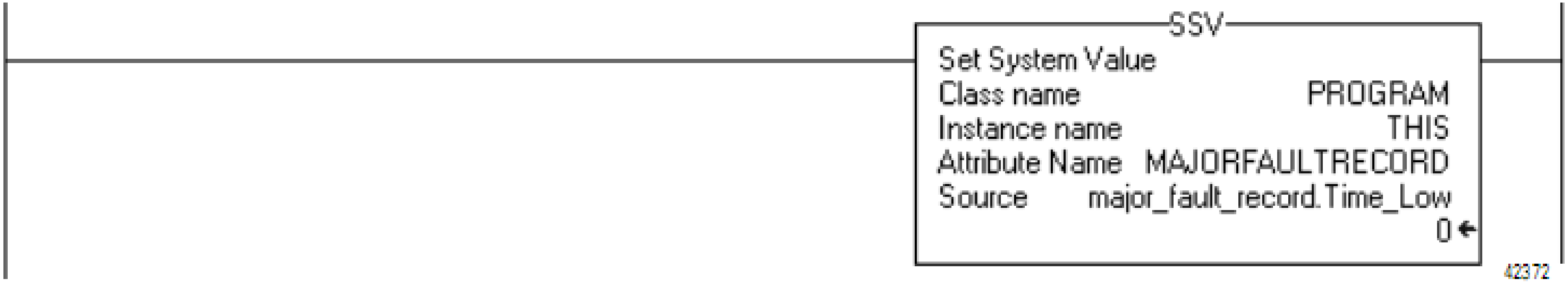
2. The second EQU instruction checks for a specific fault code. In Source B, enter the value for the code that you want to clear.

3. The first CLR instruction sets to zero the value of the fault type in the major_fault_record tag.

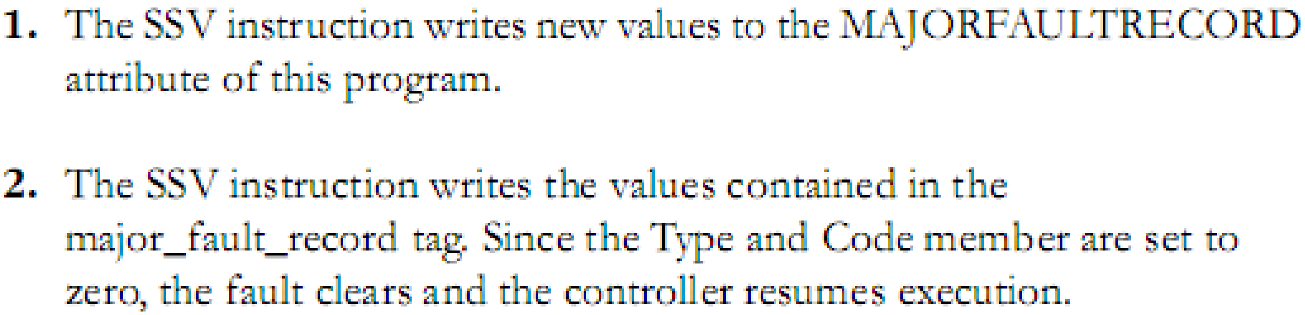
4. The second CLR instruction sets to zero the value of the fault code in the major_fault_record tag.

HANDLE FAULTs

Clear Fault.



SSV
Set System Value
Class name PROGRAM
Instance name THIS
Attribute Name MAJORFAULTRECORD
Source major_fault_record.Time_Low
0 ←

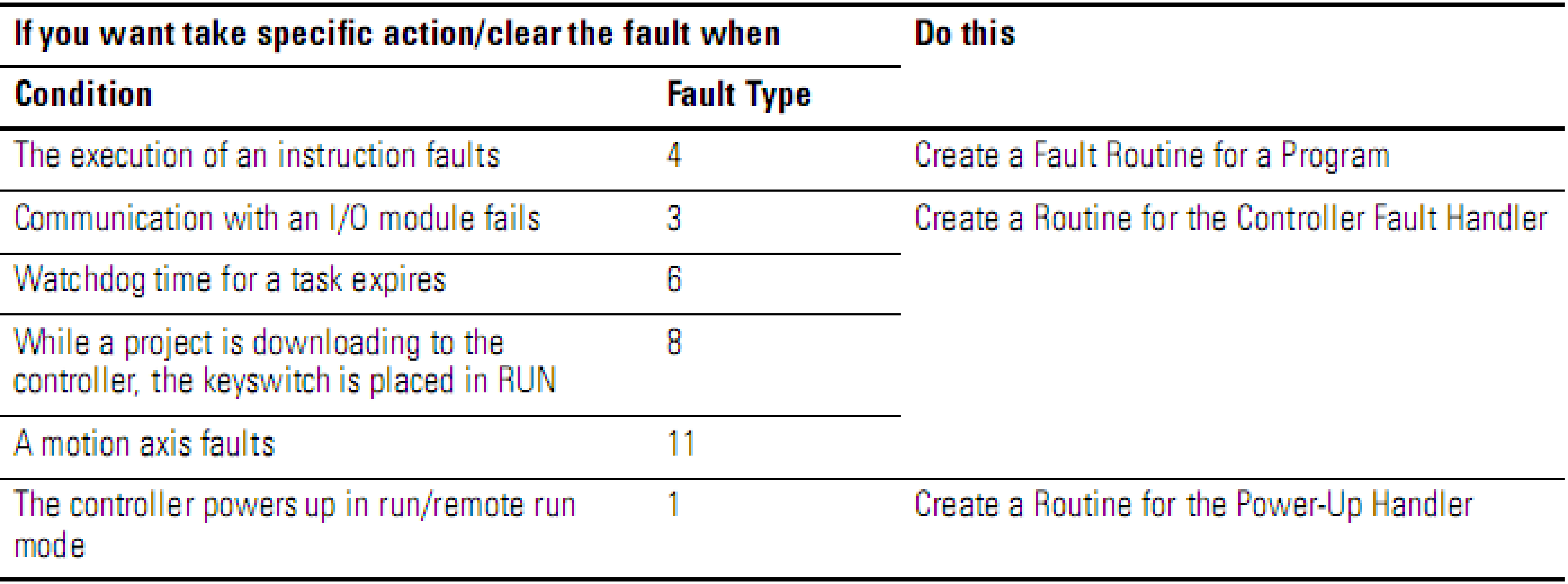


1. The SSV instruction writes new values to the MAJORFAULTRECORD attribute of this program.

2. The SSV instruction writes the values contained in the major_fault_record tag. Since the Type and Code member are set to zero, the fault clears and the controller resumes execution.

HANDLE FAULTs

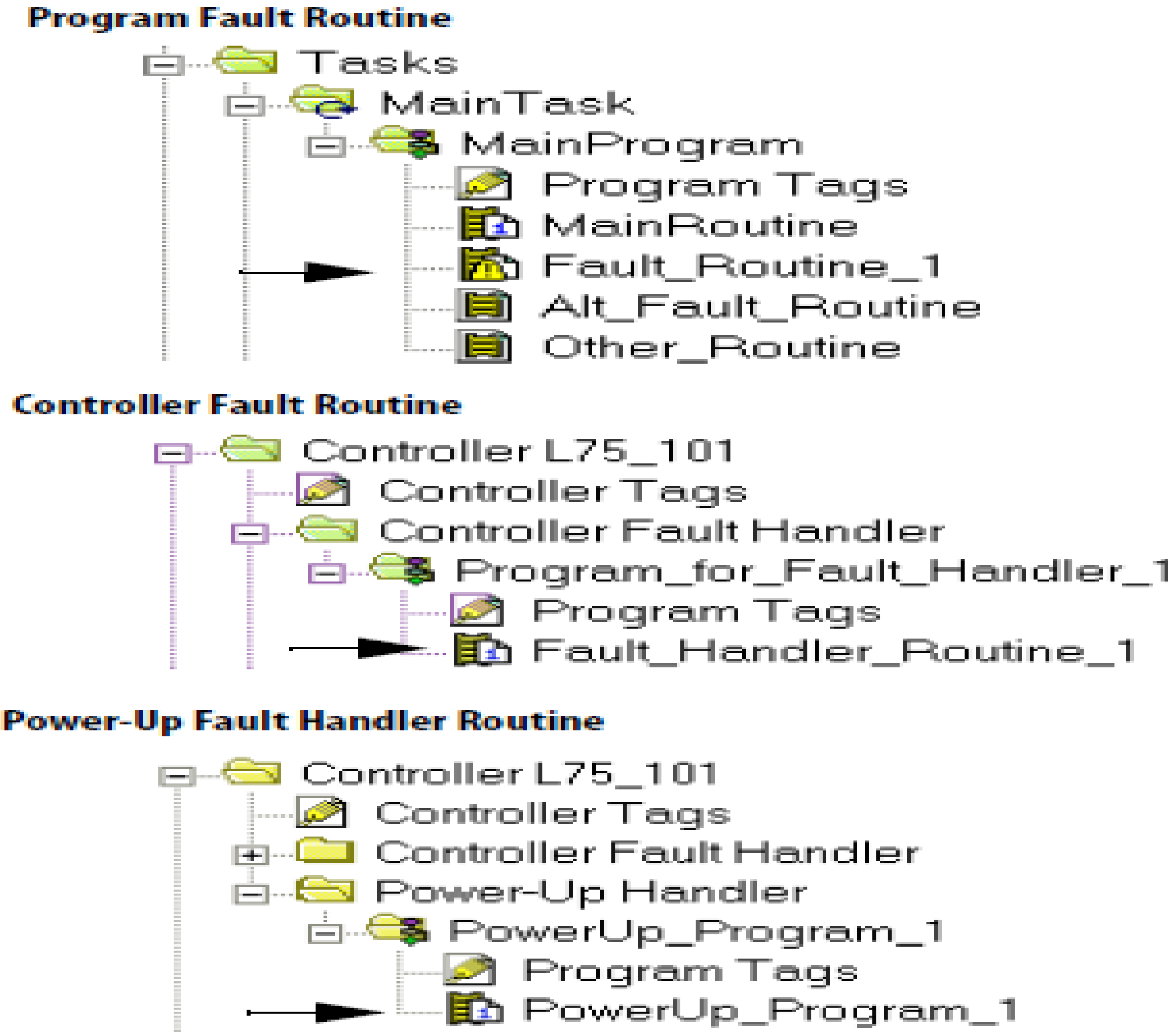
Choose Where To Place The Fault Routine .



If you want take specific action/clear the fault when		Do this
Condition	Fault Type	
The execution of an instruction faults	4	Create a Fault Routine for a Program
Communication with an I/O module fails	3	Create a Routine for the Controller Fault Handler
Watchdog time for a task expires	6	
While a project is downloading to the controller, the keyswitch is placed in RUN	8	
A motion axis faults	11	
The controller powers up in run/remote run mode	1	Create a Routine for the Power-Up Handler

HANDLE FAULTs

Choose Where To Place The Fault Routine .



Program Fault Routine

- Tasks
  - MainTask
    - MainProgram
      - Program Tags
      - MainRoutine
      - Fault_Routine_1
      - Alt_Fault_Routine
      - Other_Routine

Controller Fault Routine

- Controller L75_101
  - Controller Tags
  - Controller Fault Handler
    - Program_for_Fault_Handler_1
      - Program Tags
      - Fault_Handler_Routine_1

Power-Up Fault Handler Routine

- Controller L75_101
  - Controller Tags
  - Controller Fault Handler
  - Power-Up Handler
    - PowerUp_Program_1
      - Program Tags
      - PowerUp_Program_1

HANDLE FAULTs

Example: Check and clear the fault when CPU powered in run mode: Type =1, Code = 1.

Create a Data type to store fault information of program

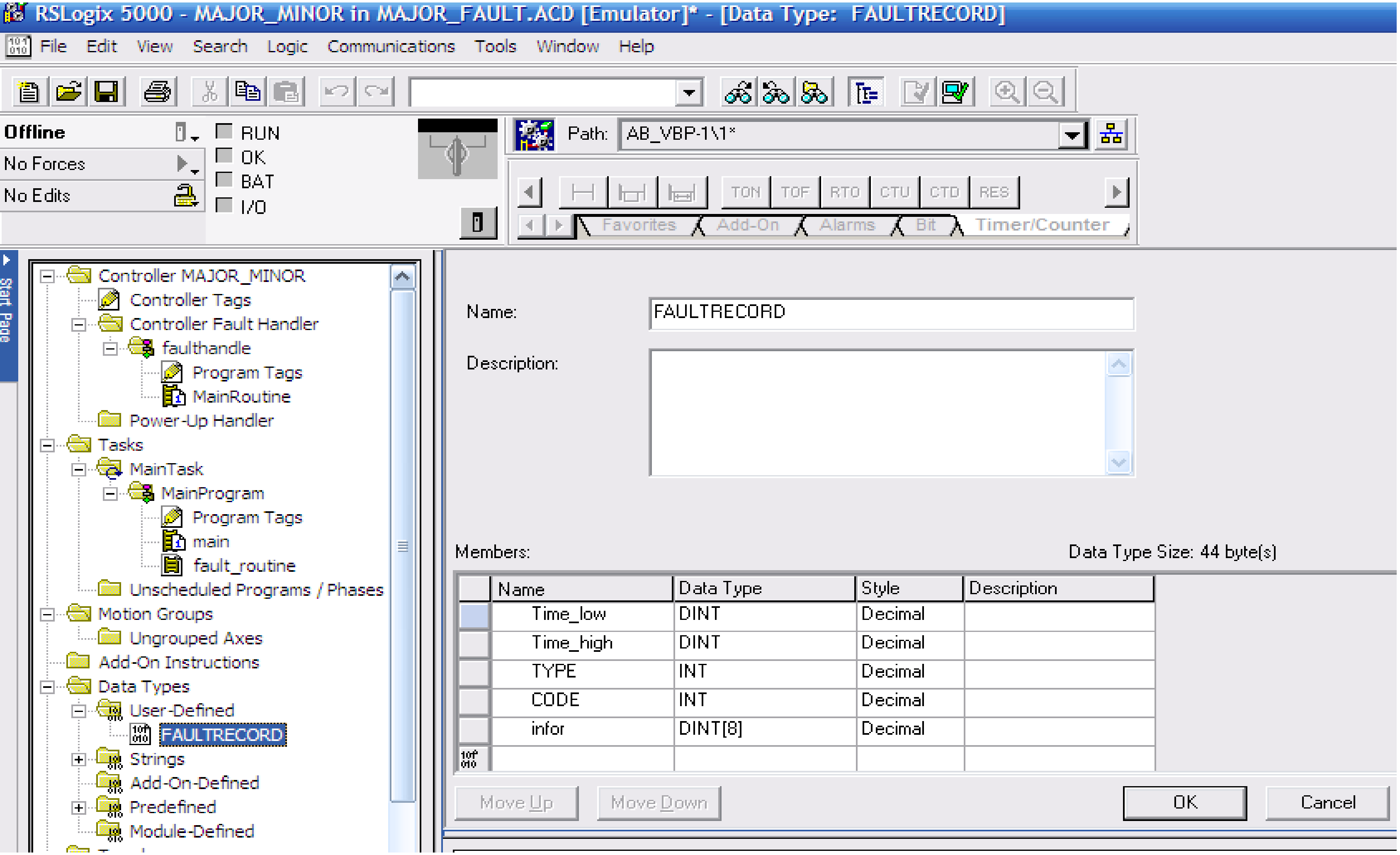
Use GSV instruction to read MAJORFAULTRECORD attribute of the program

Check specific fault code of **Type** and **Code** and clear

Use SSV instruction to write new value to MAJORFAULTRECORD attribute of the program.

HANDLE FAULTs

- Create a Data Type to store fault information of program



RSLogix 5000 - MAJOR_MINOR in MAJOR_FAULT.ACD [Emulator]* - [Data Type: FAULTRECORD]

Name: FAULTRECORD

Description:

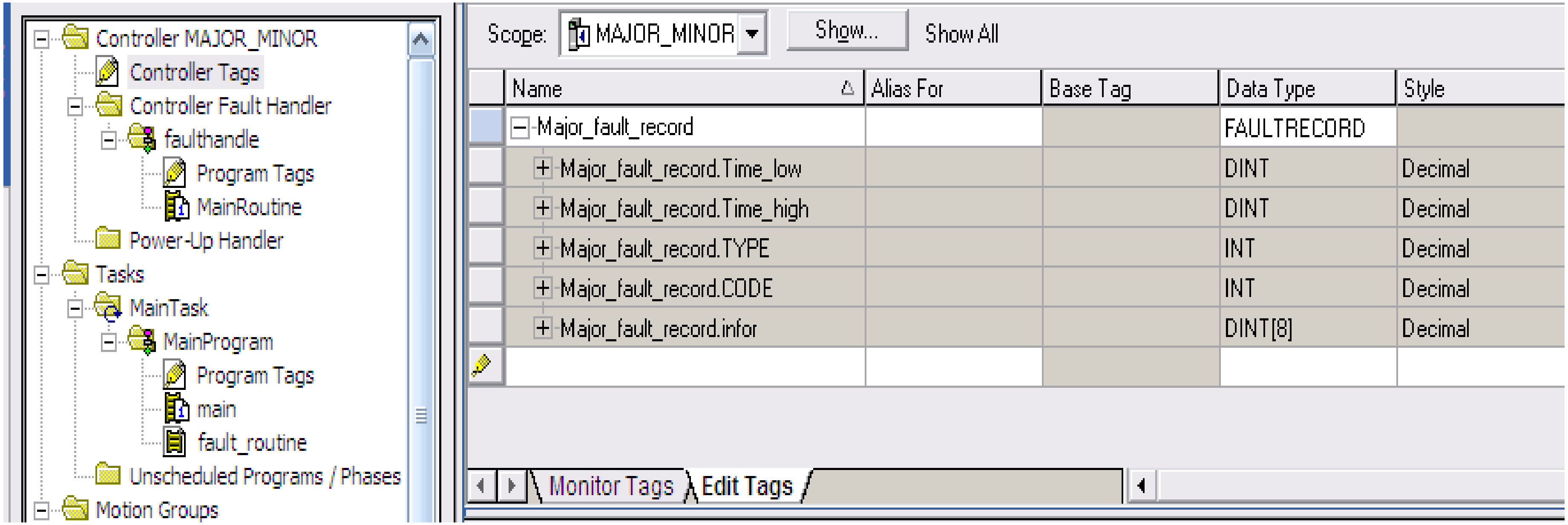
Members:

Data Type Size: 44 byte(s)

Name	Data Type	Style	Description
Time_low	DINT	Decimal	
Time_high	DINT	Decimal	
TYPE	INT	Decimal	
CODE	INT	Decimal	
infor	DINT[8]	Decimal	

HANDLE FAULTs

- Create a tag to store MAJORFAUTRECORD of the program

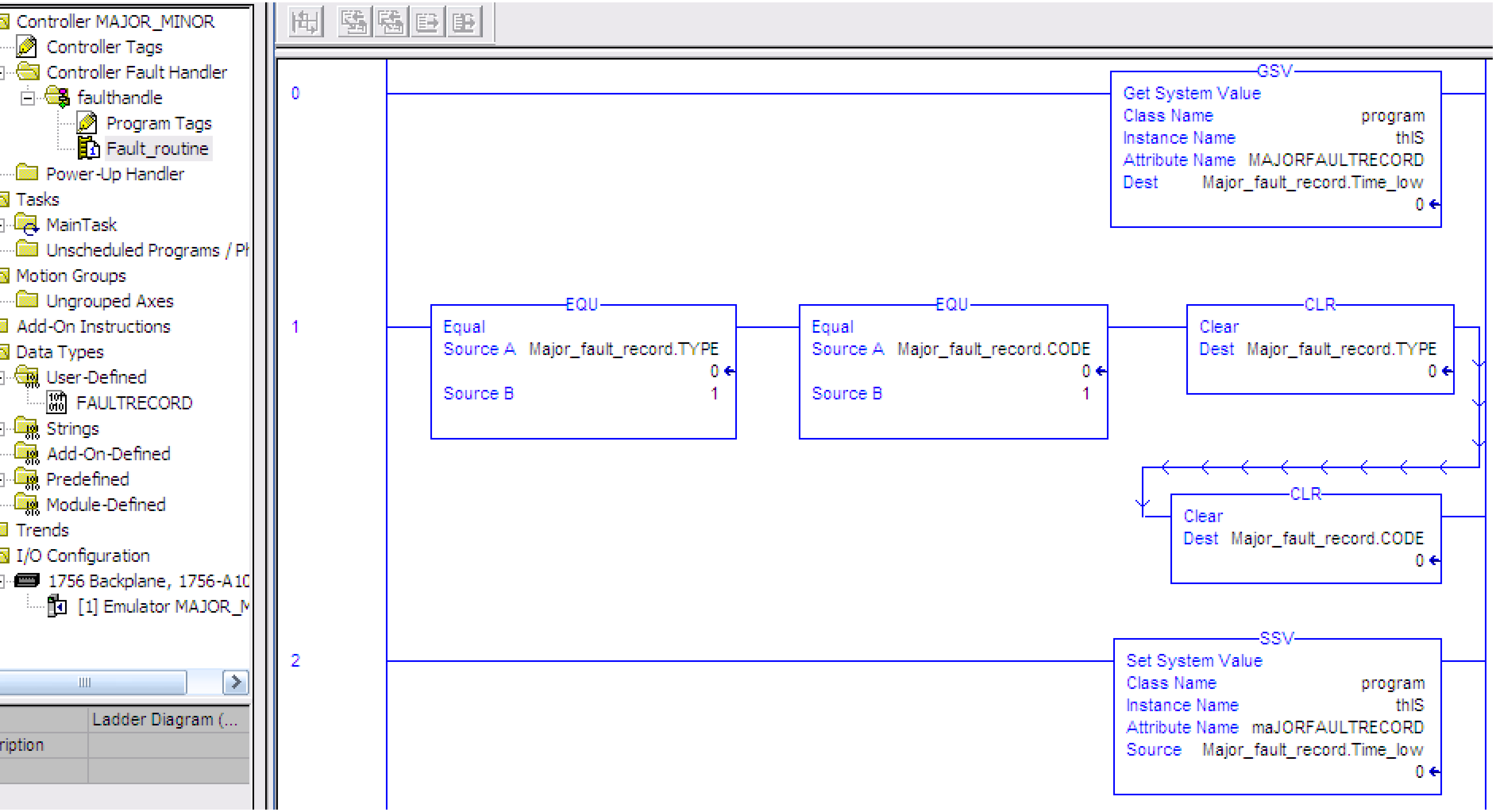


Scope: MAJOR_MINOR

Name	Alias For	Base Tag	Data Type	Style
⊟ Major_fault_record			FAULTRECORD	
⊞ Major_fault_record.Time_low			DINT	Decimal
⊞ Major_fault_record.Time_high			DINT	Decimal
⊞ Major_fault_record.TYPE			INT	Decimal
⊞ Major_fault_record.CODE			INT	Decimal
⊞ Major_fault_record.infor			DINT[8]	Decimal

HANDLE FAULTs

- Create a routine in Controller Fault Handler and write a program as following



```
0   GSV
    Get System Value
    Class Name          program
    Instance Name       thIS
    Attribute Name      MAJORFAULTRECORD
    Dest                Major_fault_record.Time_low
                        0

1   EQU
    Equal
    Source A   Major_fault_record.TYPE
               0
    Source B   1

    EQU
    Equal
    Source A   Major_fault_record.CODE
               0
    Source B   1

    CLR
    Clear
    Dest   Major_fault_record.TYPE
           0

    CLR
    Clear
    Dest   Major_fault_record.CODE
           0

2   SSV
    Set System Value
    Class Name          program
    Instance Name       thIS
    Attribute Name      maJORFAULTRECORD
    Source              Major_fault_record.Time_low
                        0
```

HANDLE FAULTs

EX2: Handle fault when download program to cpu in run mode.

EX3: Handle fault when configure a wrong module