Gæðaskjal (GSK) GSK-1066
Date of issue: 7.1.2021 Revision no.4.0:
Responsible:Torfi Dan Sævarsson
Editors: Hannes Rúnar Herbertsson

07-Instrumentation

PLC - PROGRAMMING

Doc. no.: NA-07-STS009

This standard technical description is subject to change without prior notice. The most current issue will at all times be located on the Nordural web site, www.nordural.is.

Contents

Table of figures

Table of tables

1 Responsibility

This Standard Technical Specification (STS) is of responsibility of the owner. The revision and date of issue are on the front page.

IMPORTANT: All deviations from the specifications must be approved in writing by the Owner.

2 Introduction

This document describes control systems configuration standards that must be used and implemented at Norðurál locations. The document specifically handles the configuration of devices, networks and PLC modules. It also provides directives for Level-0 and Level-1 control systems programming and provides the basis of the various entity tag naming.

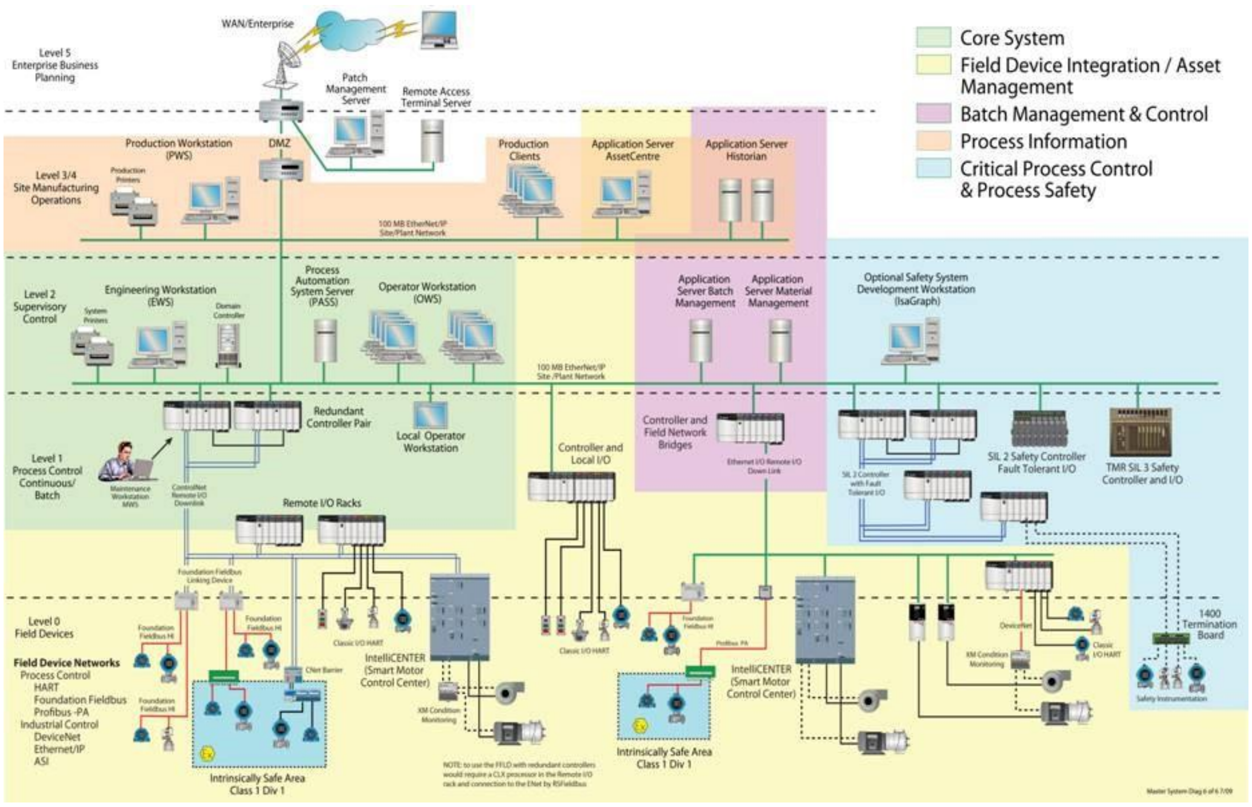

WAN/Enterprise
Level 5 Enterprise Business Planning
Patch Management Server
Remote Access Terminal Server
Core System
Field Device Integration / Asset Management
Batch Management & Control
Process Information
Critical Process Control & Process Safety
Production Workstation (PWS)
DMZ
Production Clients
Application Server AssetCentre
Application Server Historian
Level 3/4 Site Manufacturing Operations
Production Printers
100 MB EtherNet/IP Site/Plant Network
Level 2 Supervisory Control
Engineering Workstation (EWS)
System Printers
Domain Controller
Process Automation System Server (PASS)
Operator Workstation (OWS)
Application Server Batch Management
Application Server Material Management
Optional Safety System Development Workstation (IsaGraph)
100 MB EtherNet/IP Site /Plant Network
Level 1 Process Control Continuous/ Batch
Redundant Controller Pair
Local Operator Workstation
Maintenance Workstation MWS
ControlNet Remote I/O Downlink
Remote I/O Racks
Controller and Local I/O
Controller and Field Network Bridges
Ethernet I/O Remote I/O Down Link
SIL 2 Controller with Fault Tolerant I/O
SIL 2 Safety Controller Fault Tolerant I/O
TMR SIL 3 Safety Controller and I/O
Foundation Fieldbus Linking Device
Level 0 Field Devices
Foundation Fieldbus HI
Field Device Networks
Process Control
HART
Foundation Fieldbus
Profibus -PA
Industrial Control
DeviceNet
Ethernet/IP
ASI
Classic I/O HART
IntelliCENTER (Smart Motor Control Center)
XM Condition Monitoring
Intrinsically Safe Area Class 1 Div 1
Profibus PA
DeviceNet
1400 Termination Board
Safety Instrumentation

Figure 1 – Plant System Architecture Levels.

All new installations, setups and arrangements shall follow the instructions provided by this standard. This also applies for all new additions i.e. modules that are installed onto a preexisting system, network, remote or local racks, etc. Even if the preexisting system is not configured according to this standard.

This document is the master standard for PLC programming and should be consider the principal standard. Supporting documents either support specific statements made in this document, such as appendices, guidelines, program version criteria, or test documents. Supporting documents will complement the maser standard and can be issued more rapidly.

Norðurál uses automation equipment from various suppliers, having Allen Bradley Rockwell Automation as the primary supplier. All new installations or migration projects must therefore take that into a consideration. Norðurál will allow themselves to decline a solution as they see fit, solely, if the solution does not use equipment from Allen Bradley Rockwell Automation.

This document namely contains the following standards;

- ControlLogix configuration and programming standards.
- CompactLogix configuration and programming standards.
- Ethernet configuration standards.
- PLC Programming Handbook.

The level-0 and level-1 communication network protocols for the Norðurál Site shall be Ethernet/IP using DLR technology. Other communication protocols require owner's approval.

Password protecting any part of a PLC code or network application is strictly prohibited and will be resolved by rejection of handover by Norðurál.

All additions, corrections done for a PLC software previously handed to Norðurál, and/or new installations, must be done through Norðurál's FactoryTalk AssetCentre. Only during initial testing and SAT[1] is it allowed to work directly connected to the PLC, and only then if the PLC does not handle other function specified in the project (that is the sole purpose of the PLC is one specific equipment, machine or area not formerly established within Norðurál' s site).

This document is the master document to be supported by a series of other documents such as;

Standard Name	Description
AKS Handbook [ISL]	Description of the AKS coding system [Icelandic]
AKS Handbook [ENG]	Description of the AKS coding system [English]
AKS Key Total Plant	Function Key Main Group
AKS Key 1 System	AKS Key 1 system
AKS Key 2 Aggregates	AKS Key 2 Aggregates
AKS Key 3 Components	AKS Key 3 Components

Table 1 – Norðurál AKS Coding system (STS004)

The AKS coding system is designed with the objective of obtaining control systems that will operate with optimum efficiency, reliability, and low cost of ownership. To obtain a plant wide uniformity of design and configuration of the control systems, all requirements of these standards are to be strictly followed, regardless of which group or entity is involved or in charge of the design. Any change or deviation from the standards herein shall be considered on a case by case basis and submitted for evaluation to the owner.

2.1 UPS Requirements

All PLC applications shall be connected through a UPS and shall receive a signal from the UPS indicating the state of the UPS to safely put the PLC controlled system in a safe state after power loss. This must be done so that PLC system will take appropriate action when power is restored. The restart state of equipment or machines can wary depending on the application.

[1] Site Acceptance Testing

3 Software types

In this document, the generic use of the term software can designate one of the following specific elements;

- Firmware - This software is specific to the device or machine and controls the hardware directly, e.g. Control Flash CLX version 19, Intel BIOS.
- Infrastructure software – This software shall be for platform level hardware and run directly on the firmware, BIOS or basic operating system of platform, e.g. Windows 10, Windows Server, ADS, DNS, DHCP and WINs.
- Application Infrastructure software - This software shall run on a base layer of infrastructure and provides application support for a specific application, e.g. FactoryTalk Linx, I.I.S (Internet Information Service), FactoryTalk directory.
- Development software – Concerns the software used for the development of application code and configuration duties for Level-0 and Level-1 automation systems, e.g. Studio 5000 Logix Designer, FTView Studio, and RSNetWorx.
- Application Software - Consists in the software developed by the control systems designers and vendors using any development software package.

3.1 File type standards

The following file types shall be used for;

Name	Type
ControlLogix, RSLogix 5000	*.ACD
RSNetWorx Ethernet/IP	*.enet

Table 2 – File type standards

3.2 Version Control

This project shall fix its initial development on the following versions of software and hardware as standard.

Software	Version
Studio 5000 Logix Designer	30.00.00 (CPR 9 SR 9)
RSLogix 5000	20.01.00 (CPR 9 SR 5)
RSLinx-classic	4.00.00 (CPR 9 SR 10.0)
RSLogix Emulator 5000	20.01.00 (CPR 9 SR 5)
RSNetWorx for ControlNet	27.00.00 (CPR 9 SR 10)
RSNetWorx for DeviceNet	27.00.00 (CPR 9 SR 10)
RSNetWorx for Ethernet	27.00.00 (CPR 9 SR 10)
Connecting Components Workbench	11.00.00

Table 3 – Software versions

Hardware and software requirements can change annually, it is therefore vital that vendors ensure that they develop their applications with the appropriate versions, that are available to Norðurál at the time of handover.

3.3 PLC Vendor Mandates

The PLC vendors must provide the following documentation with a unique document number according to vendor coding system;

Title	Type
Functional Description	Document
Simulation procedure	Document
Factory Acceptance Test (FAT)	Document
Site Acceptance Test (SAT)	Document
Fault and Alarm List	Excel
PLC Tag List	CSV
Custom UDT Definition	Document
Interlock list	Document
Custom AOI Definition	Document
All other documents included in the contract	

Table 4 – PLC Vendor Mandates

The vendor could be requested to provide the following services, as per contract;

- Support MES[2] integration i.e. installation, commissioning, debugging etc.
- Support for the HMI / SCADA configuration and testing.
- All other activities included in the contract

[2] MES: Manufacturing Execution System

4 Control module assignment standards

The following control module assignment standards shall apply for the Norðurál Site.

All new installations, setups and arrangements need to follow this standard. This also implies if new modules are installed within a preexisting system, network, remote or local racks etc. Notice that this also applies if the preexisting system is not configured according to this standard.

In general, CompactLogix controller series can be used as a controller for an individual machine and/or equipment. While the extended use of remote racks is designed as a control strategy, ControlLogix controller series are preferred. This concerns the design of the local rack and main CPU, mainly. Chassis Sizes and Modules Slot Assignment

The processor shall be placed in the left-hand slot (slot 0) of the chassis (the local rack). Where multiple processors are placed within a single chassis, they are to be placed in priority order starting from the left-hand side. Empty slots should be closed off with blank cover module (1756-N2).

The physical location of the module within a chassis must follow certain rules in order make the control systems at Norðurál homogenous and to optimize the maintenance of the control system. The assignment of module within the rack slightly varies depending on the function of the rack and whether it is:

- Local Rack (series 1756)
- Remote Rack (series 1756)
- Compact Logix (series 1768)
- Remote Rack with Flex I/O (series 1794)
- Remote Rack with Flex 5000 I/O (series 5094)
- Remote Rack with Point I/O (series 1734)
- Remote Safety Point IO (series 1734)

4.1 Local Rack Modules Position

Modules are listed in Table 5 and in Figure 2 by position importance in the rack from left to right.

If multiple modules of any type are required, they should be lined up in sequence and the following modules should be moved respectively.

If a certain module type is not used, the next module should start in the next available slot, e.g. if a rack only contains Digital Inputs the first input should start in the first empty slot after the controller or the communication adapter.

1756 Local Rack Modules	Control Level	Slot	Comment
Processor Modules	1	0	
Ethernet Modules	2	1	
Ethernet Modules	1	2	Depending on use
Specialty I/O[3] Modules	0-1	3	Depending on use
Analogue I/O Modules	0	4-5	Depending on use
Discrete I/O Modules	0	6-7	Depending on use

Table 5 – Local Rack Configuration

[3] I/O - Inputs/Outputs. Specialty I/O[3] Modules - High Speed Counter, Multi-Vendor Interface Module, Thermocouple Input Module etc.

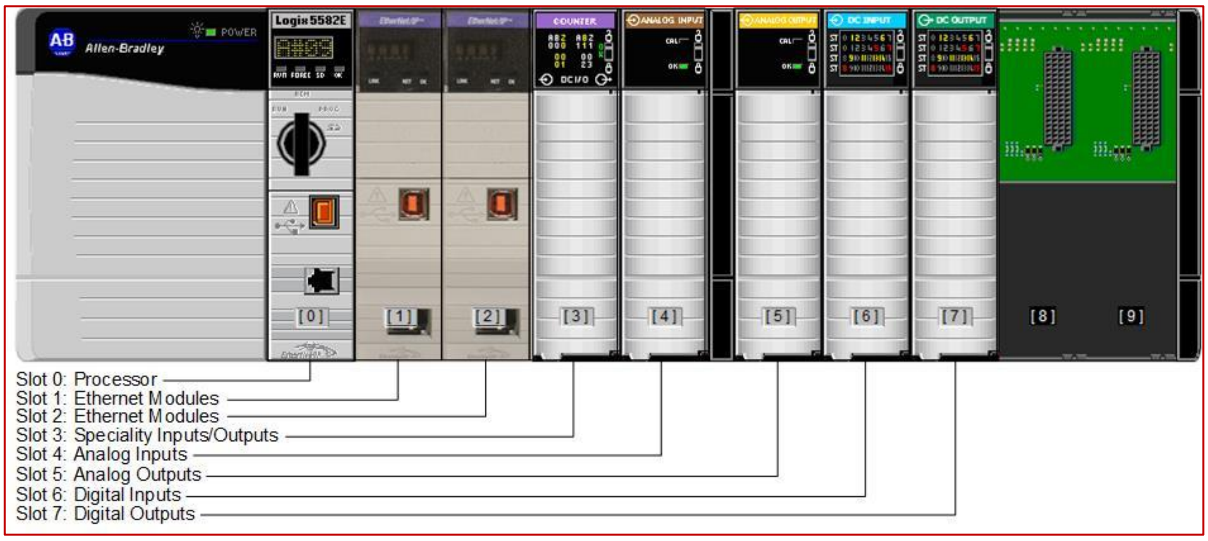


Figure 2 – 1756 ControlLogix Rack Configuration.

In safety setup applications, the safety partner is always in slot number 1. All subsequent slot numbers are increased by one.

4.2 Remote I/O Rack Modules Position

The modules are listed below by position importance in the rack from left to right.

If multiple modules of any type are required, they should be lined up in sequence and the following modules should be moved respectively.

If a certain module type is not used, the modules should start in the next available slot, e.g. if a rack only contains Digital Inputs the first input should start in the first empty slot after the communication adapter.

4.2.1 ControlLogix Remote Rack Modules Position

Module	Type	Control Level	Slot	Comment
Communication Adapter	Ethernet	1	0	
Specialty I/O Modules		0	1	Depending on use
Analogue I/O Modules		0	2-3	Depending on use
Discrete I/O Modules		0	4-5	Depending on use

Table 6 – ControlLogix Remote Rack Configuration

4.2.2 Flex I/O Remote Rack Module Position

Module	Type	Control Level	Slot	Comment
Communication Adapter	Ethernet	1	NaN	
Specialty I/O Modules		0	0	Depending on use
Analogue I/O Modules		0	1	Depending on use
Discrete I/O Modules		0	2	Depending on use

Table 7 – Flex Remote Rack Configuration

4.2.3 Point I/O Remote Rack Module Position

Module	Type	Control Level	Slot	Comment
Communication Adapter	Ethernet	1	NaN	
Specialty I/O Modules		0	0	Depending on use
Analogue I/O Modules		0	1	Depending on use
Discrete I/O Modules		0	2	Depending on use

Table 8 – Point IO Remote Rack Configuration

4.3 Spare requirements

The vendors shall guaranty that there is a minimum of 20% spare or available slots for further expansions. This applies for all new installations, new control panels, local and remote racks etc. the following minimum should be guaranteed;

- Control panels should have space for expansions of 20% or more. This involves cabling, space for increase number of slots (expansions) in the local and/or remote rack, terminal blocks and circuit breakers etc.
- Number of slots (expansions) in local and remote racks should be able to be expended 20 % of installed module count.
- Minimum of 20% empty slot for spare modules in all racks shall be available.
- Modules should have minimum of 20% spare inputs/outputs. This applies to digital and analog modules, not specialty modules like; counter modules, high speed counter modules, communication modules etc.

5 Naming Standards

At Norðurál, a software tag based on the following naming standard must be given to each device in the Studio 5000 Logix Designer software i.e. processor, communication modules, I/O module etc.

Area Code	Description
00	General
10	Utilities
20	Administration
30	Material Handling
40	Power
50	Reduction
60	Anode Production
70	Casting
80	Environmental
90	Automation Standard

Table 9 – Area Codes.

The tag structure explained in this section uses a DEVICE type code. The table below provides a list of these DEVICE type codes with their description.

Device Type Code	Device Type Description
ACN	ControlNet Adapter Module
ACS	ABB ACS (VFD)
ADN	DeviceNet Adapter Module
AENT	Flex Ethernet Adapter
AI	Analogue Input Module
AO	Analogue Output Module
CLX	ControlLogix CPU
CN2(R)	ControlNet Bridge Module (Redundant)
CNB(R)	ControlNet Bridge Module (Redundant)
CPL	CompactLogix CPU
DI	Digital Input Module
DNB	DeviceNet Scanner Bridge Module
DO	Digital Output Module
EN2T	Ethernet Bridge Module
ENBT	Ethernet Bridge Module
ETAP	Ethernet Tap module
GLX	GuardLogix CPU
HI	Hart Input Module
HO	Hart Output Module
HSC	High Speed Counter module
ICE	Inview CE
IDI	Isolated Digital Input Module
IDO	Isolated Digital Output Module
LOR	Local Rack
MVI	Serial Communication Module
PM	Power Monitor
PVP	PanelView Plus (HMI)
RR	Remote Rack
TMQ	Telemecanique
VFD	Variable Frequency Drive

Table 10 – Device Type Codes.

5.1 Processor Naming

All new installed processors shall be named according to the following format convention:

“GRTAC_AKxxx_DTCnnn_Description”

GRT:	Norðurál Grundartangi (GRT is always present).
AC:	Refers to the area code where the application is located (see **Error! Reference source not found.**).
AK:	Norðurál AKS coding system for equipment naming convention, view reference document **Error! Reference source not found.** for further details.
xxx:	Refers to the system number. View reference document listed in **Error! Reference source not found.**, for further details.
DTC:	Refers to the device type code according to Table 10 – Device Type Codes.**Error! Reference source not found.**.
nnn:	Refers to a sequential number to ensure that no duplicate names are created within an area.
Description:	Short description of the application purpose, max. 21 characters.

The Processor module naming for the gas treatment center for pot line 1 could therefore be:

“GRT80_FD100_CLX001_GasTreatmentCenter”

Vendor is not allowed to change name of any preexisting processors. This highly applies to projects where new configurations are added to preexisting systems, as communication paths can be affected by the modification.

5.2 Rack Naming

A Local Rack always contains the processor, and a Remote Rack is controlled over an Ethernet/IP communication module (adapter or a bridge) depending of the I/O series used. A rack (that doesn’t include a processor) is considered as a Remote Rack even if the rack is in the same control panel as the Local Rack.

Racks shall be named according to the following format convention:

- LOR001, where LOR is the Device Type Code used for Local Rack and the 3-digit numbers will always be 001 as the local rack will always have the IP address x.x.x.001.
  - Generally, Redundancy controllers are not required on the Norðurál Site. Their use will be handled case by case.
- ENR**002** to ENR**nnn** where ENR stands for Ethernet Rack, and **nnn** are integers that represent the device’s host ID (last number of the IP address). IP address assigned to the rack (incrementing from **002**).

5.3 Network Communication Module Naming

This section defines the naming convention to be used for network communication modules (1756-ENBT, 1756-EN2T, 1734-AENT). **Error! Reference source not found.** provides the Network Identifiers (NetID) code established on the project. Notice the NetID is often a part of a module name.

IMPORTANT: ControlNet and DeviceNet communication modules are not allowed in new system installations nor in new configurations. These modules are used in legacy systems at Norðurál, and the conversion of migrating a legacy system with a valid communication module must be considered for all new additions to legacy systems. Moreover, as ControlNet and DeviceNet communication modules are obsolete products according to Norðurál standards (**Error! Reference source not found.**).

NetID	Network Identification Description	Status
EN	Ethernet/IP used for an isolated standalone device 01-254	Valid
CN	ControlNet used for Control and Remote I/O level (xx is the network number 01-99)	Obsolete
DN	DeviceNet used for Distributed devices (xx is the network number 01-99)	Obsolete

Table 11 – Network Identification Description

5.3.1 Ethernet/IP Communication Module in a Local Rack

The module naming shall be as per the following format:

"IDxxx_RANnnn_SmmDTC"

ID:	Refers to the communication module network ID. **EN** is identified as I/O Ethernet/IP Network ID.
xxx:	The network ID integer number (last 3-digits of the module corresponding IP address).
RAN:	Refers to the rack naming convention described in section 0.
nnn:	The sequential number **nnn** is always 001 for a local rack since it will always be number one, "LOR001".
S:	Label **S** stands for Slot and is always present.
mm:	Integer number that represents the slot number of the communication module in the corresponding Local Rack, S is always present.
DTC:	The Device Type Code acronym used for an Ethernet Bridge module ENB.

Following example shows a naming convention for an Ethernet/IP module:

"EN001_LOR001_S01ENB"

5.3.2 Ethernet/IP Communication Module in a Remote Rack

If the Ethernet module on Ethernet number 1 in local rack number 1 connects to a remote rack with a Flex communication adapter the name would be as shown.

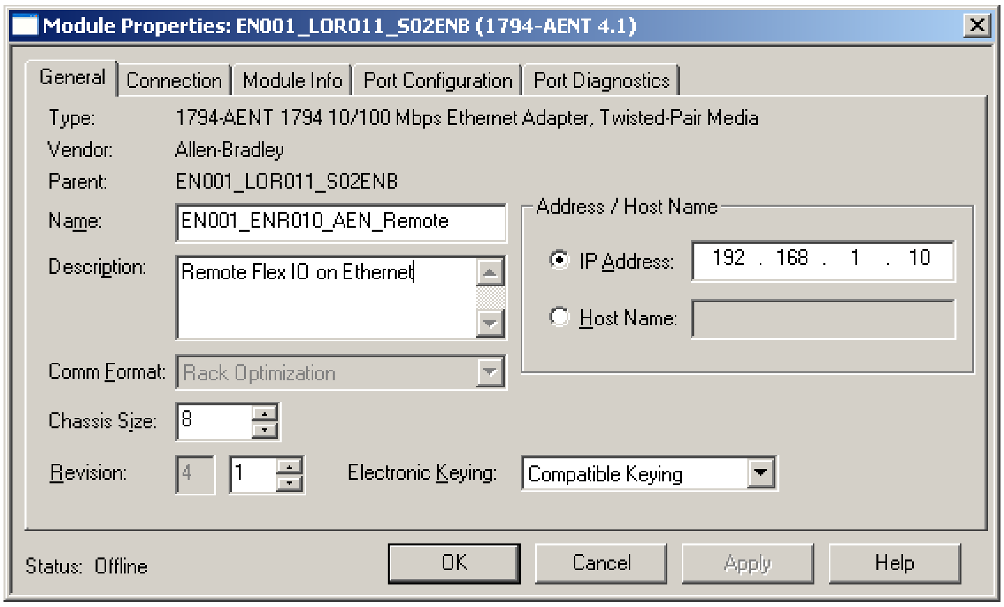
"IDxxx_RANnnn_DTC_Description"

ID:	Refers to the communication module network ID. **EN** is identified as I/O Ethernet/IP Network ID.
xxx:	The network ID integer number (last 3-digits of the module corresponding IP address).
RAN:	Refers to the rack naming convention described in section 0.
nnn:	Remote Ethernet Rack ID integer number (last 3-digits of the module corresponding IP address).
DTC:	The Device Type Code acronym used for an Ethernet adapter module AEN.
Description:	Short description of the remote location

Following example shows a naming convention for Ethernet module:

"EN001_ENR010_AEN_FilterTop"

The configuration screen for an Ethernet Adapter installed in a remote rack is shown in **Error! Reference source not found.**.


Module Properties: EN001_LOR011_S02ENB (1794-AENT 4.1)

General | Connection | Module Info | Port Configuration | Port Diagnostics

Type: 1794-AENT 1794 10/100 Mbps Ethernet Adapter, Twisted-Pair Media
Vendor: Allen-Bradley
Parent: EN001_LOR011_S02ENB
Name: EN001_ENR010_AEN_Remote
Description: Remote Flex IO on Ethernet
Comm Format: Rack Optimization
Chassis Size: 8
Revision: 4 1
Electronic Keying: Compatible Keying

Address / Host Name
(●) IP Address: 192 . 168 . 1 . 10
( ) Host Name:

Status: Offline

Figure 3 – Ethernet module configuration.

5.3.3 Directly Connected Devices on Ethernet

In addition to Ethernet/IP modules devices can be connected directly on the Ethernet network such as VFD's, MCC's etc.

The naming convention for directly connected equipment on Ethernet shall be as per the following format:

"IDxxx_DTCnnn_EQT"

For example, if we have a VFD (PowerFlex drive) connected on the Ethernet network EN01 and configured with the Ethernet address last number 012, its name will be:

ID: Refers to the communication module network ID. **EN** is identified as the corresponding Ethernet/IP Network ID that the VFD is connected.

xxx: The identified Network ID integer number (last 3-digits of the module corresponding IP address).

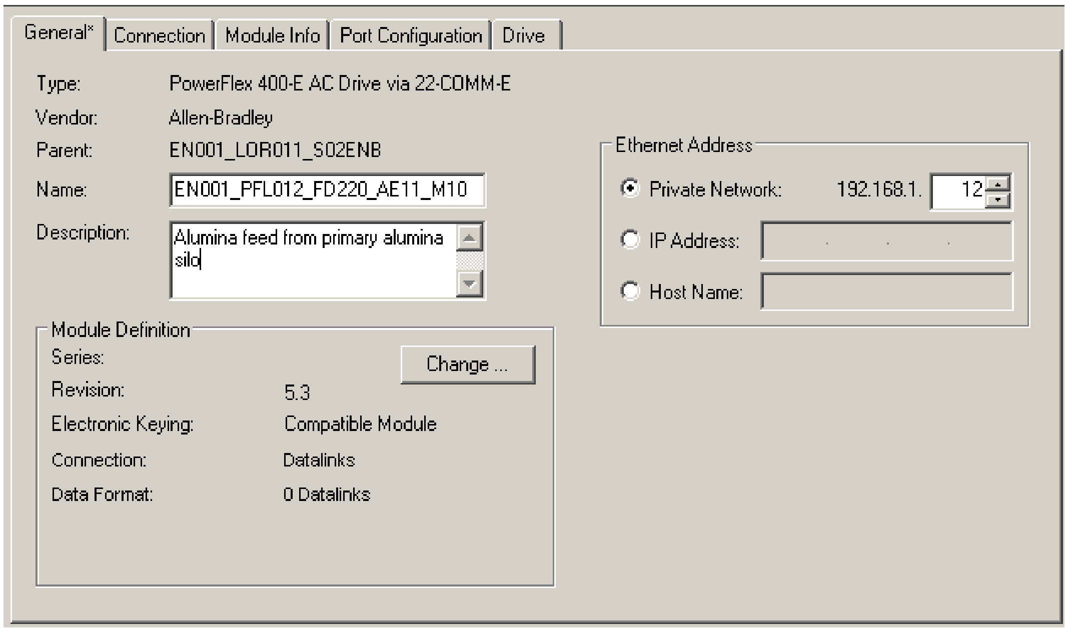
DTC: The Device Type Code acronym used for a PowerFlex drive is PFL.

nnn: Integer number that represents the last 3-digits of the device IP address. For instance, if a PowerFlex drive is configured with the IP address "192.168.1.12" then **nnn** would be labelled **012.**

EQT: Refers to the AKS naming convention to identify type of equipment. In this example to equipment tag to identify a motor.

"EN001_PFL012_FD220_AE11_M10"

The configuration for VFD on Ethernet is shown in Figure 4.


General* | Connection | Module Info | Port Configuration | Drive

Type: PowerFlex 400-E AC Drive via 22-COMM-E
Vendor: Allen-Bradley
Parent: EN001_LOR011_S02ENB
Name: EN001_PFL012_FD220_AE11_M10
Description: Alumina feed from primary alumina silo

Module Definition
Series:
Revision: 5.3
Electronic Keying: Compatible Module
Connection: Datalinks
Data Format: 0 Datalinks

Ethernet Address
[✓ Private Network: 192.168.1. 12]
IP Address:
Host Name:

Figure 4 – Directly connected device on Ethernet.

5.4 I/O Module Naming

I/O modules placed within the racks (Local or Remote) shall be labeled upon location and device type as standard, where the location refers to the chassis slot identification.

5.4.1 Module Configuration

Error! Reference source not found. shows the naming conventions for a typical I/O configuration tree in RSLogix 5000.

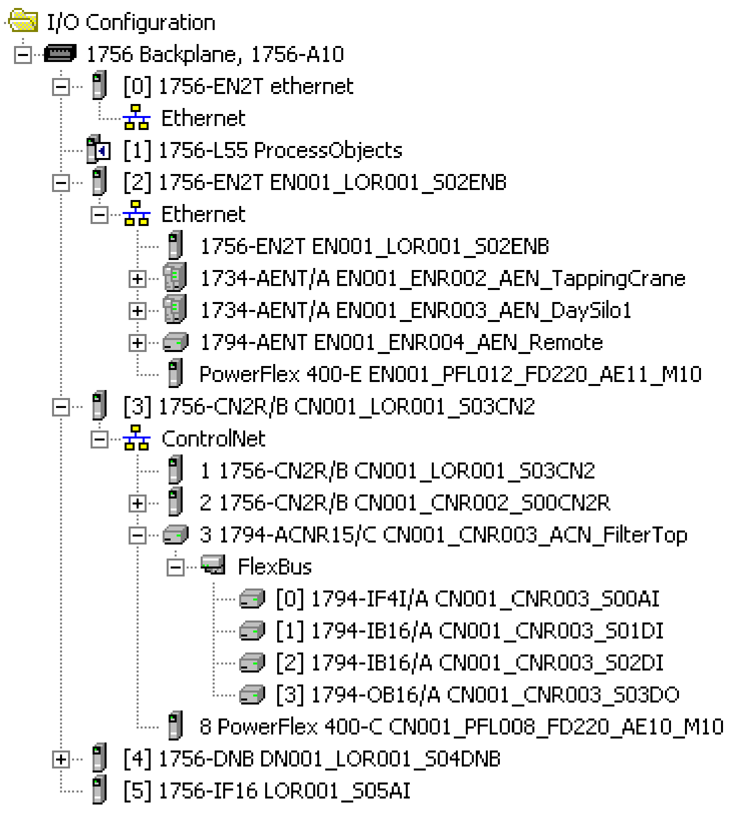

```
I/O Configuration
  1756 Backplane, 1756-A10
    [0] 1756-EN2T ethernet
      Ethernet
    [1] 1756-L55 ProcessObjects
    [2] 1756-EN2T EN001_LOR001_S02ENB
      Ethernet
        1756-EN2T EN001_LOR001_S02ENB
        1734-AENT/A EN001_ENR002_AEN_TappingCrane
        1734-AENT/A EN001_ENR003_AEN_DaySilo1
        1794-AENT EN001_ENR004_AEN_Remote
        PowerFlex 400-E EN001_PFL012_FD220_AE11_M10
    [3] 1756-CN2R/B CN001_LOR001_S03CN2
      ControlNet
        1 1756-CN2R/B CN001_LOR001_S03CN2
        2 1756-CN2R/B CN001_CNR002_S00CN2R
        3 1794-ACNR15/C CN001_CNR003_ACN_FilterTop
          FlexBus
            [0] 1794-IF4I/A CN001_CNR003_S00AI
            [1] 1794-IB16/A CN001_CNR003_S01DI
            [2] 1794-IB16/A CN001_CNR003_S02DI
            [3] 1794-OB16/A CN001_CNR003_S03DO
        8 PowerFlex 400-C CN001_PFL008_FD220_AE10_M10
    [4] 1756-DNB DN001_LOR001_S04DNB
    [5] 1756-IF16 LOR001_S05AI
```

Figure 5 – Module Configuration.

5.4.2 IO Module in a Local Rack

The module naming shall be as the following format the **RANxxx**. Network ID is not required in the naming convention for a Local Rack as it does not control it.

"RANxxx_SmmDTC"

RAN:	Refers to the rack naming convention described in section 0.
nnn:	The sequential number **nnn** is always 001 for a local rack since it will always be number one, "LOR001".
S:	Label **S** stands for Slot and is always present.
mm:	Integer number that represents the slot number of the communication module in the corresponding Local Rack, S is always present.
DTC:	The Device Type Code acronym used for a Digital Input module is DI.

The following name would be used for a digital input module located in slot 05 of the local rack:

"LOR001_S05DI"

5.4.3 IO Module in a Remote Rack

The module naming shall be as the following format:

"IDxxx_RANnnn_SmmDTC"

ID:	Refers to the communication module network ID. **EN** is identified as the corresponding Ethernet/IP Network ID.
xxx:	The network ID integer number (last 3-digits of the module corresponding IP address).
RAN:	Refers to the rack naming convention described in section 0. **ENR** is identified as Ethernet Rack for this example.
nnn:	The sequential number **nnn** is always 001 for a local rack since it will always be number one, "LOR001".
S:	Label **S** stands for Slot and is always present.
mm:	Integer number that represents the slot number of the communication module in the corresponding Remote Rack.
DTC:	The Device Type Code acronym used for a Digital Input module is DI.

The following name would be used for a Digital Input module located in slot 05 of the Ethernet Remote Rack (with IP address where 3 last digits are 021) on an EN001 Network:

"EN001_ENR021_S05DI"

5.5 Task, Program and Routine Naming

The PLC program shall only have English US text and no special character. Tag name separators shall be done using underscore (_).

5.5.1 Task Name

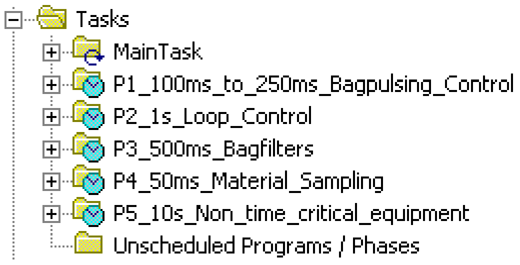
To simplify application navigation, the name of the task must include a short descriptive text that describes the task function.

⚠ The Main Task shall always be named "**MainTask**".

For the cases when periodic tasks are used, the priority and the scan rate of the task shall be included in the task name as shown in **Error! Reference source not found.**. If the scan rate is flexible, and is changed automatically from PLC, then the range of scan rate execution must be given in the task name.

⚠ Manual adjustment of scan rate of periodic tasks from SCADA and/or HMI are not allowed.

IMPORTANT: If periodic tasks are configured with flexible scan rate, the vendor must ensure robustness for the scan rate configured i.e. prevent tasks from overlapping, ensure that the controller CPU usage is under limitations[4] and ensure the task's function stability.



```
Tasks
  MainTask
  P1_100ms_to_250ms_Bagpulsing_Control
  P2_1s_Loop_Control
  P3_500ms_Bagfilters
  P4_50ms_Material_Sampling
  P5_10s_Non_time_critical_equipment
  Unscheduled Programs / Phases
```

Figure 6 – Task Naming Structure.

The "**MainTask**" shall always be the first project defined in tasks hierarchy, located at the top in the tasks tree, and followed by the periodic tasks.

The naming conventions for the periodic tasks shall be as follows:

"Px_Time_Description"

P:	Refers to the task priority level.
x:	The priority level from **P1** to **P15**, given by order of importance.
Time:	The periodic time or the time interval configured for the task, as well as the time unit (10s, 10ms etc.).
Description:	Short description of the task function.

"P1_100ms_to_250ms_Bagpulsing_Control"

⚠ Vendor must ensure that the priorities configured for periodic tasks guarantees system safety and stability. This also applies if scan rates are modified during installation. The priority can only be set once for each periodic task.

5.5.2 Program Name

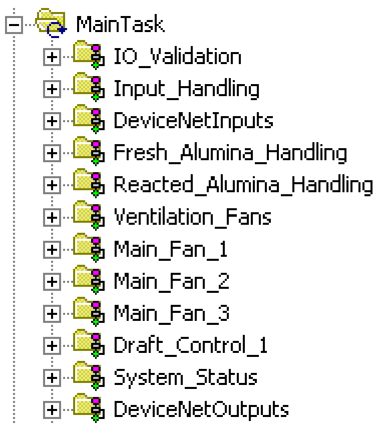
The Program names must be descriptive of their function where system grouping is useful. The number of programs should always be kept to a minimum, without losing the control system overview and integrity. Note that the quantity of declared programs can affect the PLC performance.

If applicable, then programs must always include:

- **IO_Validation** - *I/O Validation for all modules.*
- **Input_Handling** - *Input mapping / Copying.*
- **Inputs** - *Digital / Analog.*

[4] According to Rockwell Automation recommendations.

- **Outputs** - *Digital / Analog.*
- **Communication**
- **System_Status / Utilities**



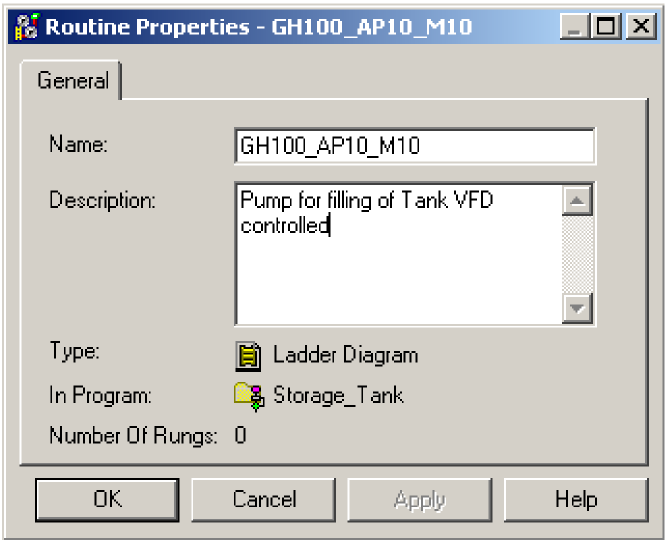
```
MainTask
  IO_Validation
  Input_Handling
  DeviceNetInputs
  Fresh_Alumina_Handling
  Reacted_Alumina_Handling
  Ventilation_Fans
  Main_Fan_1
  Main_Fan_2
  Main_Fan_3
  Draft_Control_1
  System_Status
  DeviceNetOutputs
```

Figure 7 – Program Naming Structure

5.5.3 Routine Name

Each program shall be split up into as many routines as required for the process. Mandatory is to segregate all equipment into separate routines where the routine name represents the equipment tag name. This is done for easy troubleshooting and fault finding of the PLC application.

A short descriptive text is permitted as part of the routine name for equipment. A more detailed description shall be available in the routine description field.



Routine Properties - GH100_AP10_M10

General

Name: GH100_AP10_M10

Description: Pump for filling of Tank VFD controlled

Type: Ladder Diagram

In Program: Storage_Tank

Number Of Rungs: 0

Figure 8 – Routine name and description

The example given below with:

- Filling pump GH100_AP10_M10
- Flow Pump PG010_AP11_M10_PUMP_A
- Filling valve GH100_AA10
- Discharge valve GH100_AA20
- Filling pressure GH100_CP10

- Discharge pressure GH100_CP20
- Level in tank GH100_CL10_XP10_SILO_310

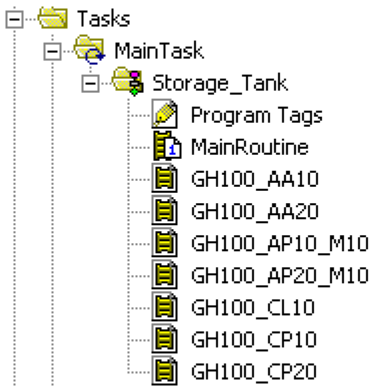


Figure 9 - Routine naming structure

Each individual routine contains all the equipment code including interlocks, alarms and permissive. Sequences and process control shall not be done within the equipment routines. But separate routines shall be created for the sequences as shown in chapter **Error! Reference source not found.**.

5.6 Tag Naming and usage

The Tags created within the processor shall be Controller scoped, for all structures used for interfacing and program task interaction as standard. The vendor shall always follow the following steps when creating a PLC application. Assistive tags, unlikely to be shared between programs or used by SCADA/HMI can be program scope.

- Unused tags should always be deleted.
- Comments are obligatory for all tasks, programs, routines, and tags.
- Comment associated to a Boolean shall reflect the “TRUE” state (Logical “1”).
- Temporary tags should always be deleted.
- Temporary tags and commissioning tags shall be marked with a comment, the developers name and contact information.

Tag naming shall be according to the Norðurál AKS[5] coding standard and shall equipment tags have the same name as given on the P&ID or electrical drawing.

The system tag is split into three components separated with an underscore (_). **Error! Reference source not found.** shows the tag structure for the AKS tag names. **Error! Reference source not found.** shows an example of the tag name structure.

[5] 00/STS004 - AKS Coding system English

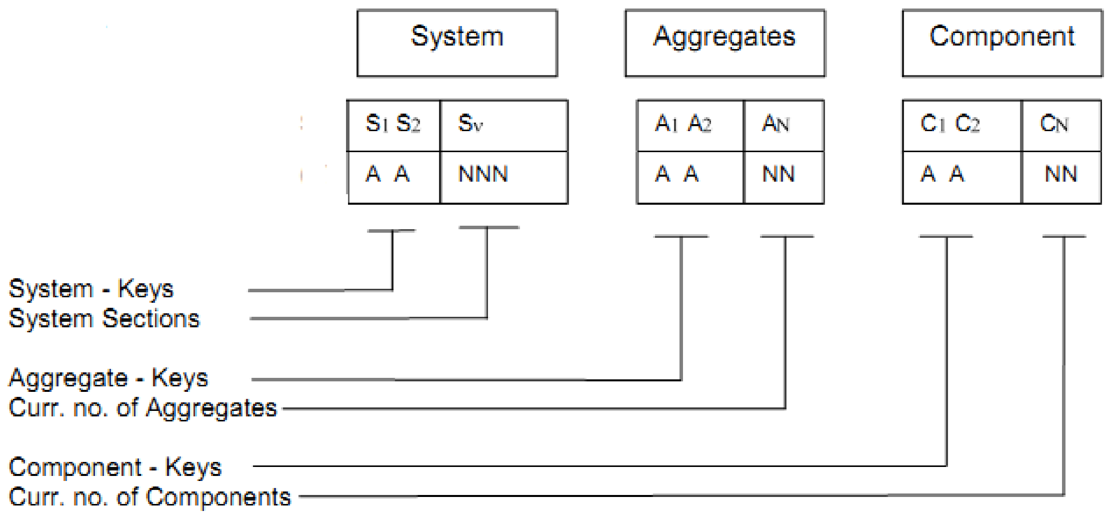


Figure 10 – AKS Mask

Refer to **Error! Reference source not found.** for information on Norðurál AKS coding system

Equipment	Tagname
Pump	GH100_AP10_M10
Fan	FD230_AN10_M10
Valve	FD120_AA10_KA10
Pressure Transmitter	FD120_CP10_XP10
Level Transmitter	FD340_CL10_XL10

Table 12 – Tag naming structure.

5.6.1 Internal PLC Tag Naming

Internal PLC tags created, shall follow the Norðurál AKS coding system. For example, if a PID controller is created then it shall take the name of the primary controlled equipment with the PID extension:

- FD230_AP10_M10 shall be FD230_AP10_M10_PID

A timer for equipment shall include the name of the corresponding equipment and including a Timer extension:

- FD230_AP10_M10 shall be FD230_AP10_M10_TIMER

If multiple timers need to be created for equipment, then an array of timers shall be created, resulting:

- FD230_AP10_M10 shall be FD230_AP10_M10_TIMER[***xx***]

Where ***xx*** is an integer number that further represents the number of timers needed for the equipment.

5.6.2 Intermediate Variables

Sometimes, intermediate variables need to be created for calculation, alarm, or any other use or function. The AKS system prefix is always preferred for tag filtering simplification in RSLogix5000.

5.6.2.1 Linked to equipment

Where tags are used in the controller as internal tags related to an existent input/output or an existing dynamic object, it is beneficial to end the tag name with an extension, which conveys easily the specific function of the tag.

The intermediate variable shall have the same format with an adjustment of the variable extension code function part using extension that shall always be preceded with an underscore (_) if used with atomic base tags.

Extension can also be used to identify a device or instrument related to equipment like a main disconnect switch for a conveyor's motor

- FD230_AN10_M10_DISC.

Extensions will be related to the attributes of the instance or elements of the object structure. For structured tags the dot (.) is used instead of an underscore (_).

- FD230_AN10_M10_TIMER[1].ACC

5.6.2.2 Unlinked

The variable shall be entered following instrument's format except that the loop number part shall take a sequential number. In these circumstances, the tag description shall be very descriptive. This kind of tag should only be used when no link exists with an existing instrument. This can be used for internal usage.

- _FD230_SEQ001_TIMER[xx]

5.6.3 External Variables

A tag used for an external variable, received, or sent, shall be entered with an underscore (_) at the beginning. These tags are for example produced, consumed and messages. All other part of the tag name shall follow the tag name of the primary controlled equipment.

Extension	Funtion
Alarm	Alarm
Alm	Alarm
Acq	Acquire
Ack	Acknowledge
Byp	Bypass
Cmd	Command
Com	Communication
Chk	Check
Disable	Disable
Disc	Disconnected
Enable	Enable
Fault	Fault
Fbk	Feedback
Hand	Hand control
Intlk	Interlock
Inhibit	Inhibit
IOFault	Communication module status
JogF	Jog Forward
JogR	Jog Revers
Lock	Lock
NB	Non bypassable
Ok	Ok
Ovrd	Override
Ovrl	Overload
Perm	Permissive
Pause	Pause
PTC	PTC fault
Rel	Release
Run (Running)	Run / Running status

Reset	Reset
Start	Start
Stop	Stop
Starting	Starting
Stopping	Stopping
Sim	Simulation
Trip	Trip
Timer	Timer
Unlock	Unlock

Table 13 – Variable extension codes

Motor disconnected signal from another PLC used in the logic for interlock purpose.

- _FD230_AN10_M10_DISC

5.6.4 Variable Extensions Codes

Variable extension codes for PLC tag naming where required as additional information or for tag separation.

5.7 RSNetWorx File Naming Standards

5.7.1 RSNetWorx for Ethernet

The file name shall be in relation with the PLC and the network area as:

“GRTAC_AKxxx_DTCmmm_IDnnn.ent”

GRT:	Norðurál Grundartangi (GRT is always present).
AC:	Refers to the area code where the application is located (see **Error! Reference source not found.**).
AK:	Norðurál AKS coding system for equipment naming convention, view reference document **Error! Reference source not found.** for further details.
xxx:	Refers to the system number. View reference document listed in **Error! Reference source not found.**, for further details.
DTC:	Refers to the device type code according to Table 10 – Device Type Codes.**Error! Reference source not found.**.
mmm:	Refers to a sequential number to ensure that no duplicate names are created within an area.
ID:	Short description of the application purpose, max. 21 characters.
nnn:	A sequential number to ensure that no duplicate names are created inside an area, see **Error! Reference source not found.**.

Thus, the naming format for this Ethernet/IP configuration file example is:

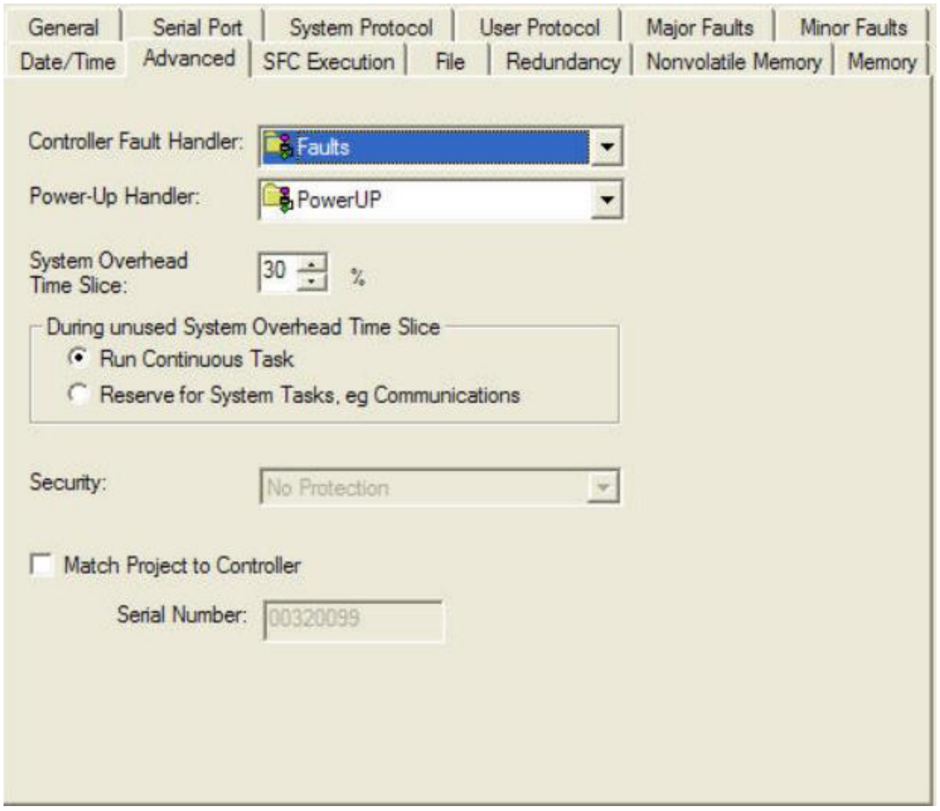
“GRT80_FD200_CLX001_EN001.ent”

6 Processor and Module Configuration Standard

The following processor and module configuration settings shall be used as standard.

6.1 Processor Configuration

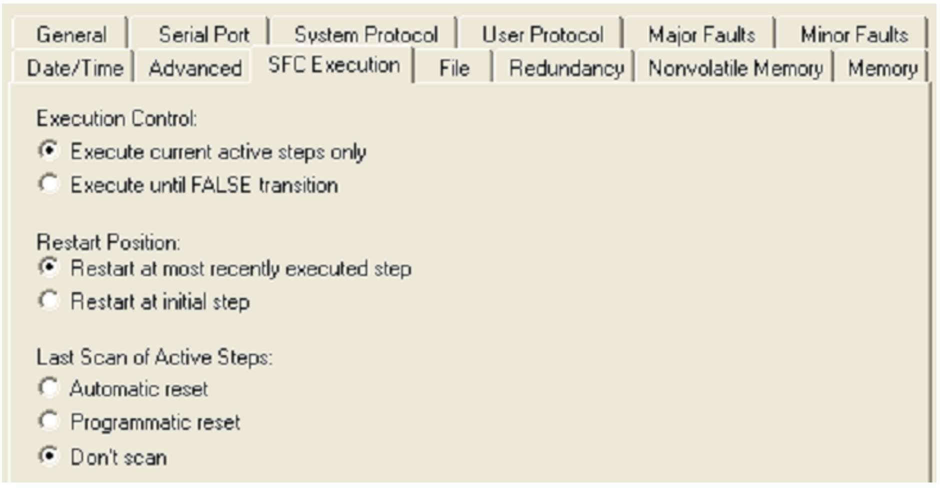
The controller shall be configured with Power up and Fault routines as standard. A system overhead time slice of 30% (default is 20%) shall be configured. This is done to gain communication time resource.



Controller Fault Handler: Faults
Power-Up Handler: PowerUP
System Overhead Time Slice: 30 %
During unused System Overhead Time Slice
[✓ Run Continuous Task] Reserve for System Tasks, eg Communications
Security: No Protection
Match Project to Controller
Serial Number: 00320099

Figure 11 – System Overhead Time Slice

The controller SFC shall also be configured to execute current active steps and restart at the most recently executed step. The configuration of the controller shall not scan the active step at last. The Default configuration shall be used se **Error! Reference source not found.**.



Execution Control:
[✓ Execute current active steps only] Execute until FALSE transition
Restart Position:
[✓ Restart at most recently executed step] Restart at initial step
Last Scan of Active Steps:
Automatic reset Programmatic reset [✓ Don't scan]

Figure 12 – SFC Execution Controller Configuration

6.1.1 Date and Time Setup

A GPS clock will be used to synchronize all servers and PLC’s at the Norðurál Site.

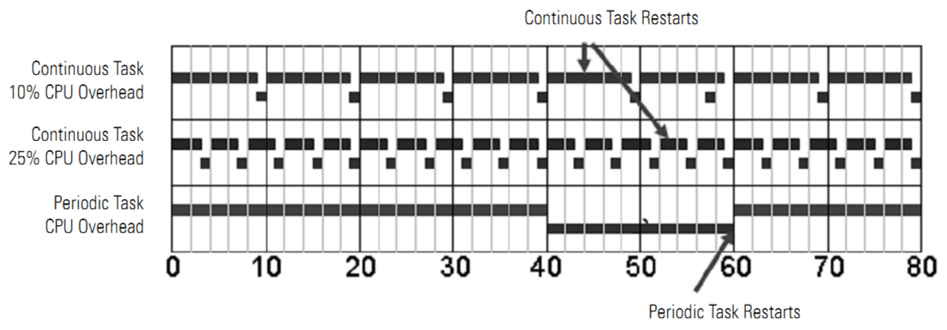
6.1.2 System Overhead Time Slice

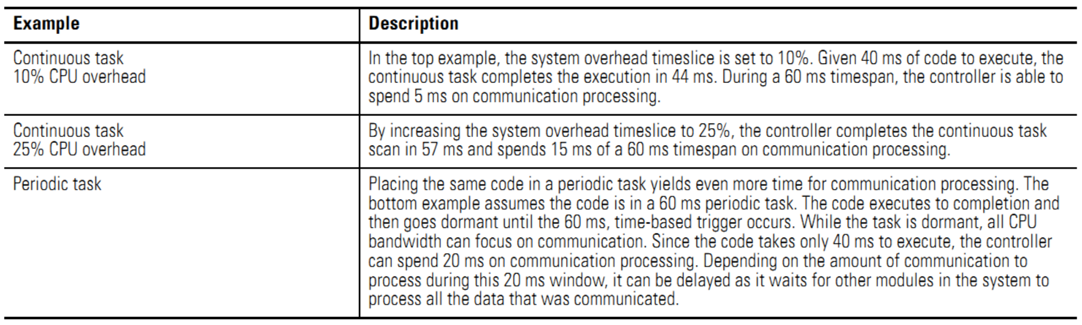
The system overhead time slice specifies the percentage of continuous task execution time that is devoted to communication and background redundancy functions. System overhead functions include the following:

- Communicating with programming and HMI devices (such as RSLogix 5000 software and FTView)
- Responding and sending messages
- Serial port message and instruction processing
- Alarm instruction processing

The controller performs system overhead functions for up to 1 ms at a time. If the controller completes the overhead functions in less than 1 ms, it resumes the continuous task. The advantage in having only periodic tasks are that the CPU will finish scanning all tasks and then the CPU goes idle and handles all other operations such as messaging and communication. When using a continuous task, the task will be interrupted regularly depending on the system overhead time slice setting.

The **Error! Reference source not found.** compares a continuous and periodic task;





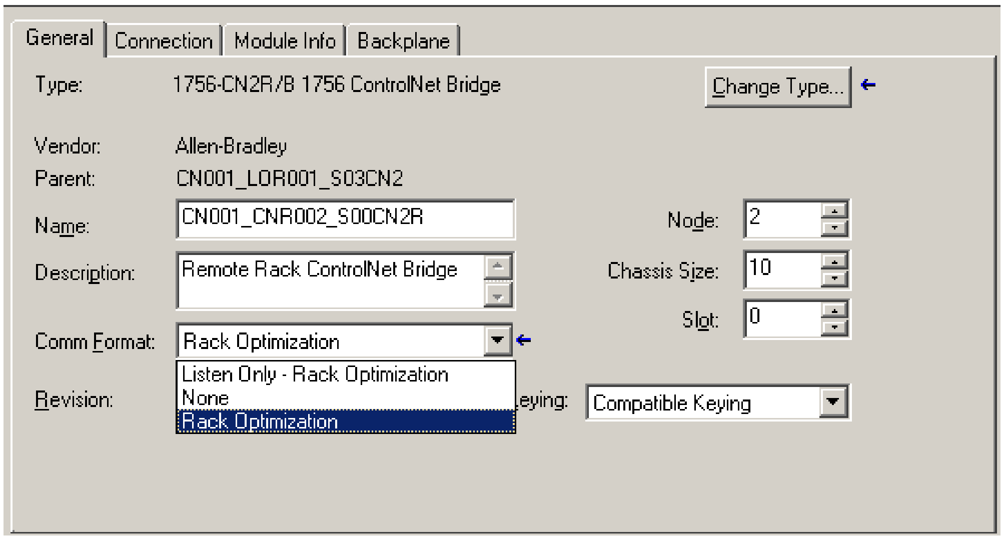
Example	Description
Continuous task 10% CPU overhead	In the top example, the system overhead timeslice is set to 10%. Given 40 ms of code to execute, the continuous task completes the execution in 44 ms. During a 60 ms timespan, the controller is able to spend 5 ms on communication processing.
Continuous task 25% CPU overhead	By increasing the system overhead timeslice to 25%, the controller completes the continuous task scan in 57 ms and spends 15 ms of a 60 ms timespan on communication processing.
Periodic task	Placing the same code in a periodic task yields even more time for communication processing. The bottom example assumes the code is in a 60 ms periodic task. The code executes to completion and then goes dormant until the 60 ms, time-based trigger occurs. While the task is dormant, all CPU bandwidth can focus on communication. Since the code takes only 40 ms to execute, the controller can spend 20 ms on communication processing. Depending on the amount of communication to process during this 20 ms window, it can be delayed as it waits for other modules in the system to process all the data that was communicated.

Figure 13 – Periodic and continuous tasks

The Logix5000 CPU time slices between the continuous task and system overhead. Each task switch between user task and system overhead takes additional CPU time to load and restore task information.

6.2 Communication Modules Setup

There are multiple ways to set up a communication module and they impact the control system in different ways as described below. The recommended setup for the Norðurál Site is Rack optimized for all Remote IO since it will limit the amount of connections used by the network.



General | Connection | Module Info | Backplane

Type: 1756-CN2R/B 1756 ControlNet Bridge
Change Type...

Vendor: Allen-Bradley
Parent: CN001_LOR001_S03CN2
Name: CN001_CNR002_S00CN2R
Node: 2
Description: Remote Rack ControlNet Bridge
Chassis Size: 10
Slot: 0
Comm Format: Rack Optimization
Listen Only - Rack Optimization
None
Rack Optimization
Revision:
Keying: Compatible Keying

Figure 14 – Communication Format settings

Rack Optimized Communication Format

- A rack optimized connection economizes connection usage between the owner and digital I/O modules in the remote chassis. Only one connection is made from the controller to the network adapter for the rack image.
- Choosing a rack connection is only available to digital I/O modules, although direct connections to digital I/O modules are also allowed.
- Analog I/O modules cannot participate in the rack connection, so all connections to Analog modules are direct connections.

“None” Communication Format

- Choosing “None” as the communication format will increase the number of connections required for a remote rack.
- Free calibration of the RPI is possible with “None” as the communication format
- The diagnostic function of digital modules will be available when using diagnostic modules.

Listen Only / Rack Optimized Communication Format

- An I/O connection where another controller owns/provides the configuration data for the module. A controller using a listen only connection does not write configuration. It can only establish a connection to the module when the owner controller is actively controlling.

6.2.1 Electronic Keying

The standard setup for electronic keying for the Norðurál Site for all modules shall be set to compatible keying.

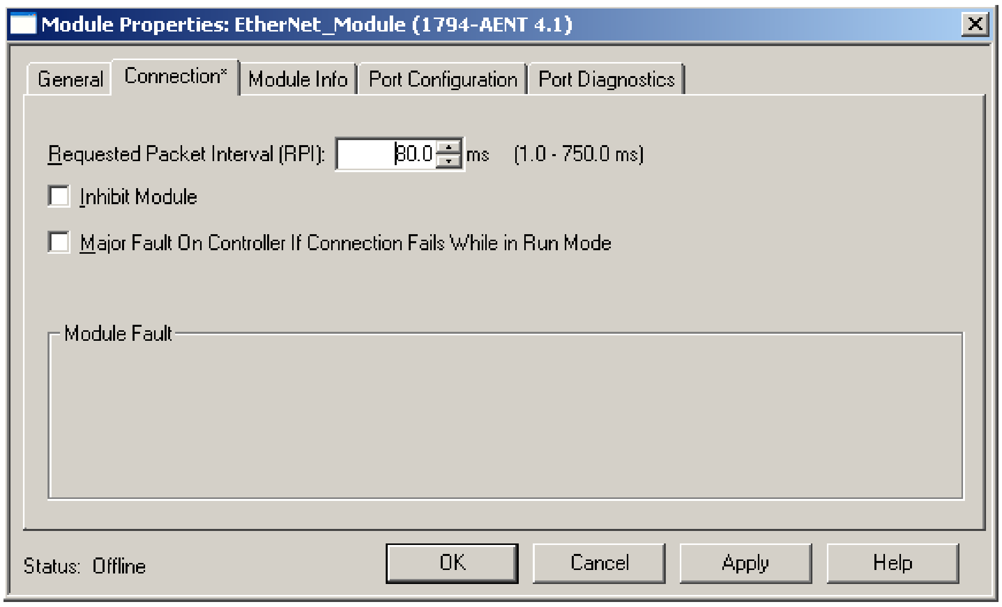
The following criteria must be met, or else the inserted module will reject the connection:

- The Module Types, Catalog Number, and Major Revision must match
- The Minor Revision of the physical module must be equal to or greater than the one specified in the software

6.2.2 RPI Settings

The RPI setting determines how fast the network scans the IO network or individual modules. The setting for the RPI must correspond to the PLC scan time. If the RPI setting is too slow compared to the scan time, then the PLC can miss updates in the IO for one or more scans. Setting the RPI faster (specifying a smaller number) than what your application needs wastes network resources, such as network processing time, and CPU processing time. Choosing an extremely fast RPI setting will result in too much time being used for I/O communication rather than being able to use the time for L2 communication

A general rule of thumbs is that the RPI should be half the scan time of the routine. For example, if the scan time of the PLC is 80ms, set the RPI at 40ms. The data is asynchronous to the controller scan, so you sample data twice as often (but no faster) than you need it to make sure you have the most current data.


Module Properties: EtherNet_Module (1794-AENT 4.1)

Requested Packet Interval (RPI): 80.0 ms (1.0 - 750.0 ms)

Inhibit Module

Major Fault On Controller If Connection Fails While in Run Mode

Module Fault

Status: Offline

Figure 15 – RPI Settings

The type of controller determines the data transmission rate.

- ControlLogix controllers transmit data at the RPI you configure for the module.
- CompactLogix controllers transmit data at powers of 2 ms (such as 2, 4, 8, 16, 64, or 128). For example, if you specify an RPI of 100 ms, the data transfers at 64 ms.

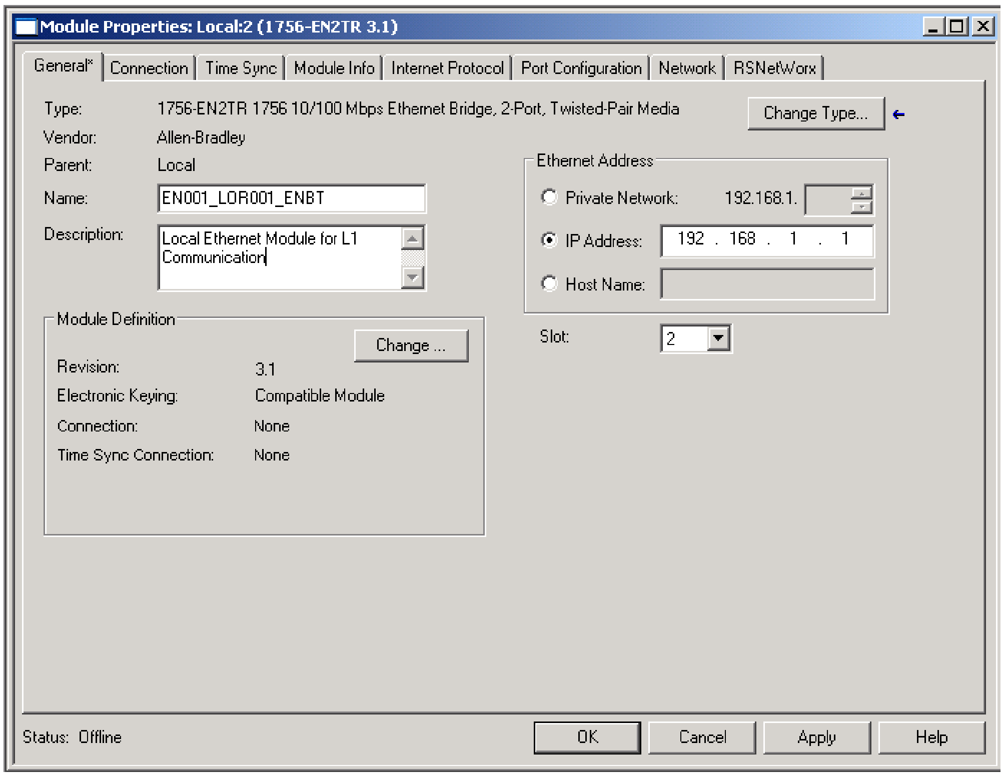
6.3 Ethernet Modules Configurations

Ethernet modules shall be configured in the following manner as standard.

For all L0 communication on Ethernet the default IP range shall be used which is 192.168.1.xxx where the L0 Ethernet module in the local rack will always have the IP address 192.168.1.1.

- PLC and IO 192.168.1.1-49
- Directly connected devices 192.168.1.50-149
- HMI devices: 192.168.1.150-199
- EWS 192.168.1.200-249

For the L0 devices the Subnet Mask shall be 255.255.255.0



Module Properties: Local:2 (1756-EN2TR 3.1)

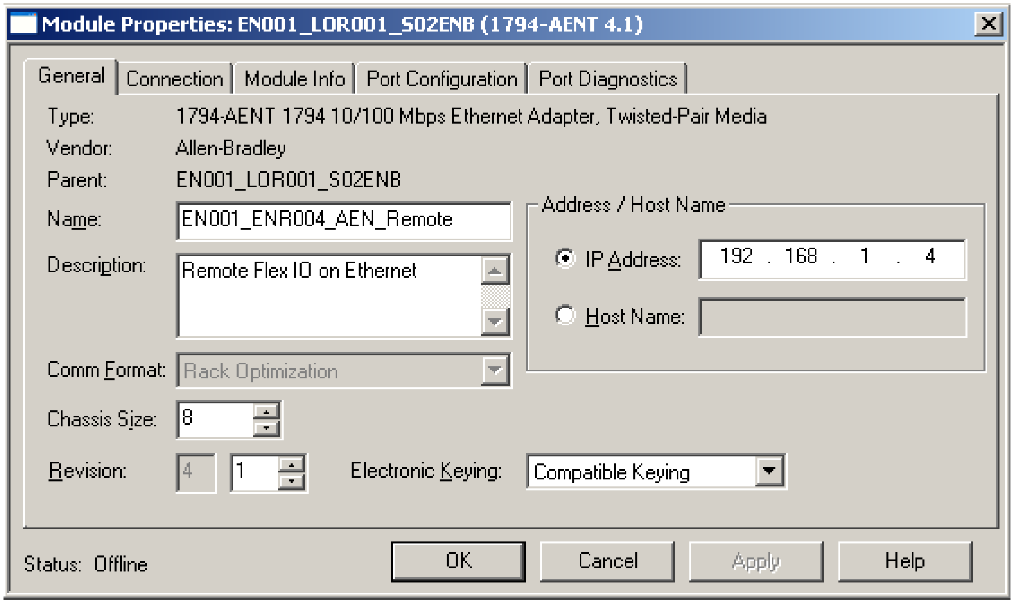
Type: 1756-EN2TR 1756 10/100 Mbps Ethernet Bridge, 2-Port, Twisted-Pair Media
Vendor: Allen-Bradley
Parent: Local
Name: EN001_LOR001_ENBT
Description: Local Ethernet Module for L1 Communication

Module Definition
Revision: 3.1
Electronic Keying: Compatible Module
Connection: None
Time Sync Connection: None

Ethernet Address
Private Network: 192.168.1.
IP Address: 192 . 168 . 1 . 1 [✓]
Host Name:
Slot: 2

Status: Offline

Figure 16 – Local Rack L0 Ethernet Modules Configuration



Module Properties: EN001_LOR001_S02ENB (1794-AENT 4.1)

Type: 1794-AENT 1794 10/100 Mbps Ethernet Adapter, Twisted-Pair Media
Vendor: Allen-Bradley
Parent: EN001_LOR001_S02ENB
Name: EN001_ENR004_AEN_Remote
Description: Remote Flex IO on Ethernet
Comm Format: Rack Optimization
Chassis Size: 8
Revision: 4 1
Electronic Keying: Compatible Keying

Address / Host Name
IP Address: 192 . 168 . 1 . 4 [✓]
Host Name:

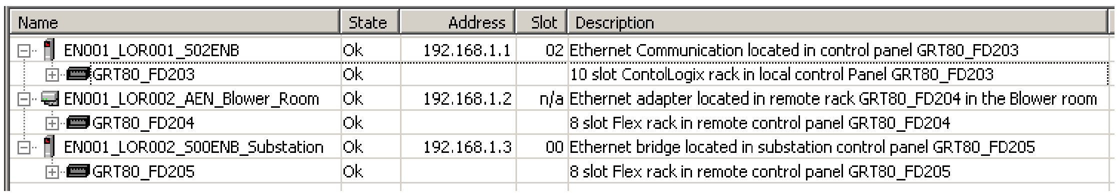
Status: Offline

Figure 17 – Remote Rack L0 Ethernet Modules Configuration

For allowed communication modules for the Norðurál Site refer to Norðurál recommended hardware document.

6.3.1 RSNetWorx for Ethernet

The tag names for equipment shall match in the RSLogix 5000 software and the RSNetWorx. Racks shall get the name of the control panel the modules are located in see **Error! Reference source not found.** and **Error! Reference source not found.**.



Name	State	Address	Slot	Description
EN001_LOR001_S02ENB	Ok	192.168.1.1	02	Ethernet Communication located in control panel GRT80_FD203
GRT80_FD203	Ok			10 slot ContolLogix rack in local control Panel GRT80_FD203
EN001_LOR002_AEN_Blower_Room	Ok	192.168.1.2	n/a	Ethernet adapter located in remote rack GRT80_FD204 in the Blower room
GRT80_FD204	Ok			8 slot Flex rack in remote control panel GRT80_FD204
EN001_LOR002_S00ENB_Substation	Ok	192.168.1.3	00	Ethernet bridge located in substation control panel GRT80_FD205
GRT80_FD205	Ok			8 slot Flex rack in remote control panel GRT80_FD205

Figure 18 – RSNetWorx for Ethernet Configuration

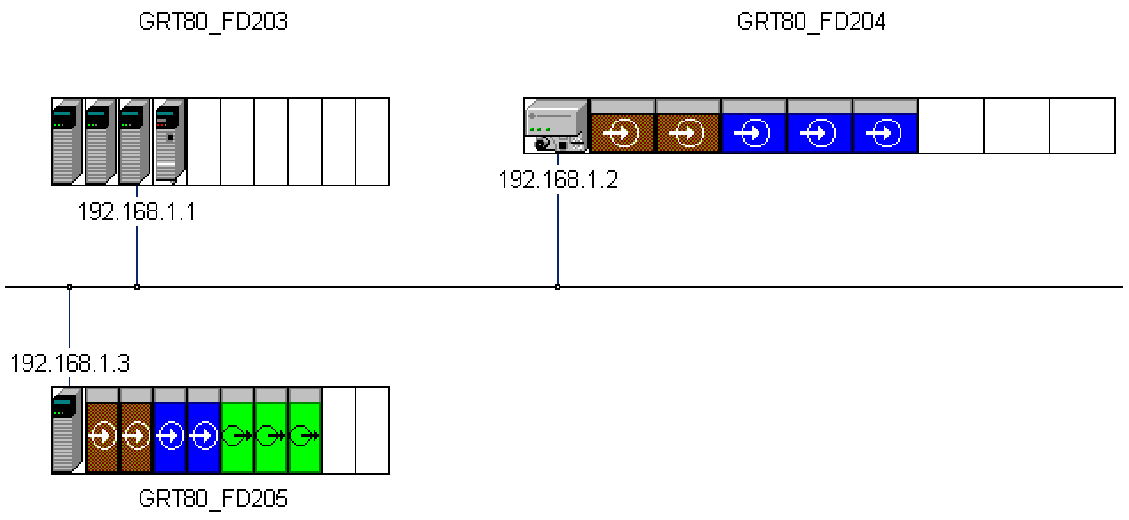


Figure 19 – Ethernet configurations

6.3.2 Level 1 and Level 2 Ethernet configuration

For Level 1 and Level 2 Ethernet communication the IP addresses shall be issued to vendors by Norðurál.

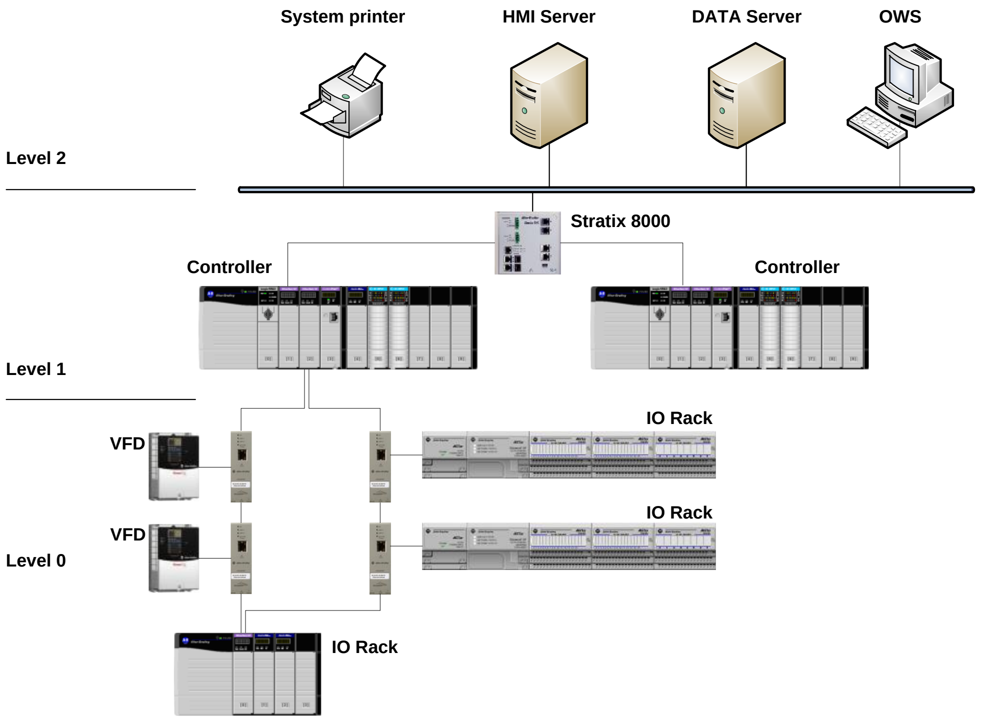


Figure 20 – Network structure

6.3.3 Monitoring of Ethernet IP network

Monitoring of the Control system Ethernet Network shall be done in the PLC and information relayed to the SCADA system, the IP addressed shall be read from the PLC available in the SCADA system on a network overview screen.

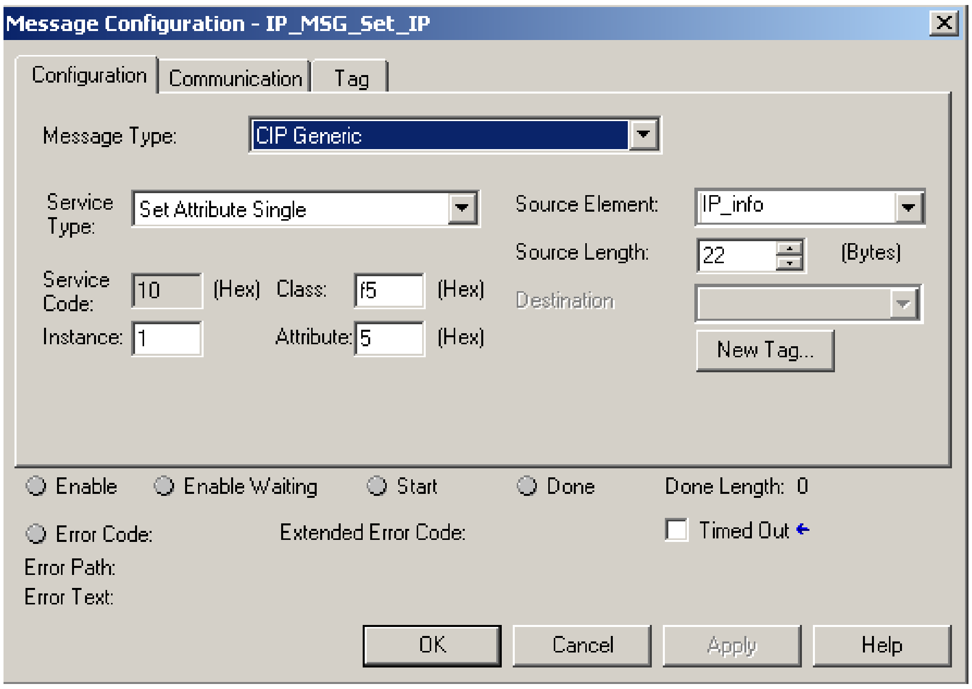
A message instruction must be used to retrieve the IP address from Ethernet devices in order to display them on a SCADA system for easy troubleshooting of the Network.

The tag name should be the same as the tag name for the module followed by _IP

Example:

> **“EN001_LOR001_S01ENB_IP”**
>
> **“EN001_ENR004_AEN_IP”**

NOTE: the descriptive text in the remote rack tag name shall be skipped in this case.


Message Configuration - IP_MSG_Set_IP

Configuration | Communication | Tag

Message Type: CIP Generic

Service Type: Set Attribute Single

Service Code: 10 (Hex) Class: f5 (Hex)

Instance: 1 Attribute: 5 (Hex)

Source Element: IP_info

Source Length: 22 (Bytes)

Destination:

New Tag...

Enable | Enable Waiting | Start | Done | Done Length: 0

Error Code: | Extended Error Code: | Timed Out

Error Path:

Error Text:

OK | Cancel | Apply | Help

Figure 21 – Message instructions to retrieve IP addresses from Modules

Further details on how to monitor IP addresses can be found in the Rockwell Automation documentation.

6.3.4 Stratix 8000 Switches

The Stratix 8000 switches will be used as standard for the Norðurál Site for level 1 communication. The approved switches and expansion modules are listed in Norðuráls recommended hardware documentation.

The Stratix 8000 Ethernet Managed Switches provide a rugged, easy-to-use, secure switching infrastructure for harsh environments. These switches can connect to network devices such as servers, routers, and other switches. In industrial environments, any Ethernet-enabled industrial communication devices can be connected including programmable logic controllers (PLCs), human-machine interfaces (HMIs), drives, sensors, and I/O.

You can mount the switches on a DIN rail in an industrial enclosure, on a wall, or panel.

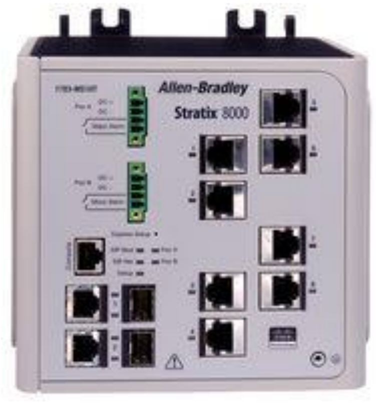


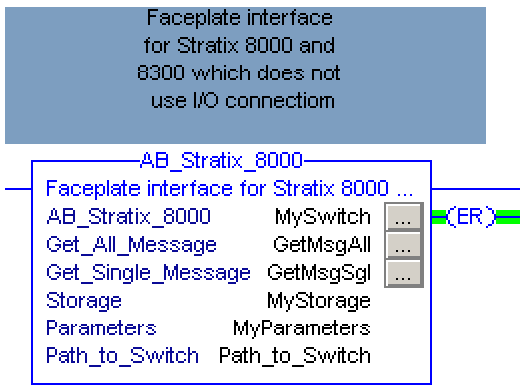
Figure 22 – Stratix 8000 switch

The selected switch shall be chosen with the requirements of the port usage. A 20% of spare ports shall be available. If more ports are needed, it is possible to add a copper or a fiber expansion module with 8 ports.

A power supply 24VDC class 2 shall be used to feed the switch. For more information about installation and configuration of Stratix 8000 see Rockwell Automation documentation.

In the case where the Stratix 8000 is used in a critical redundant system, or where it is required for higher availability and reliability, the second power input could be used. For most of the application it is not mandatory to use the second power input.

The Stratix 8000 has an AOI and a SCADA faceplate that shall be used for monitoring and maintaining the switch.



Faceplate interface
for Stratix 8000 and
8300 which does not
use I/O connectiom

```
AB_Stratix_8000
Faceplate interface for Stratix 8000 ...
AB_Stratix_8000          MySwitch  ...   (ER)
Get_All_Message          GetMsgAll ...
Get_Single_Message       GetMsgSgl ...
Storage                  MyStorage
Parameters               MyParameters
Path_to_Switch           Path_to_Switch
```

Figure 23 – Stratix 8000 AOI

6.3.5 Device Level Ring (DLR)

A DLR is an Ethernet ring network and shall be used in Ethernet applications at the Norðurál Site. The DLR protocol supports a simple, single-ring topology, it has no concept of multiple or overlapping rings. A network installation may however use more than one DLR-based ring provided each of the rings is isolated. DLR protocol messages from one ring must not be present on another ring.

A DLR network includes at least one node configured to be a ring supervisor, and any number of normal ring nodes. It is assumed that all the ring nodes have at least two Ethernet ports and incorporate embedded switch technology. Non-DLR multi-port devices “switches or end-devices” may be placed in the ring subject to certain implementation constraints. Non-DLR devices will also affect the worst-case ring recovery time.

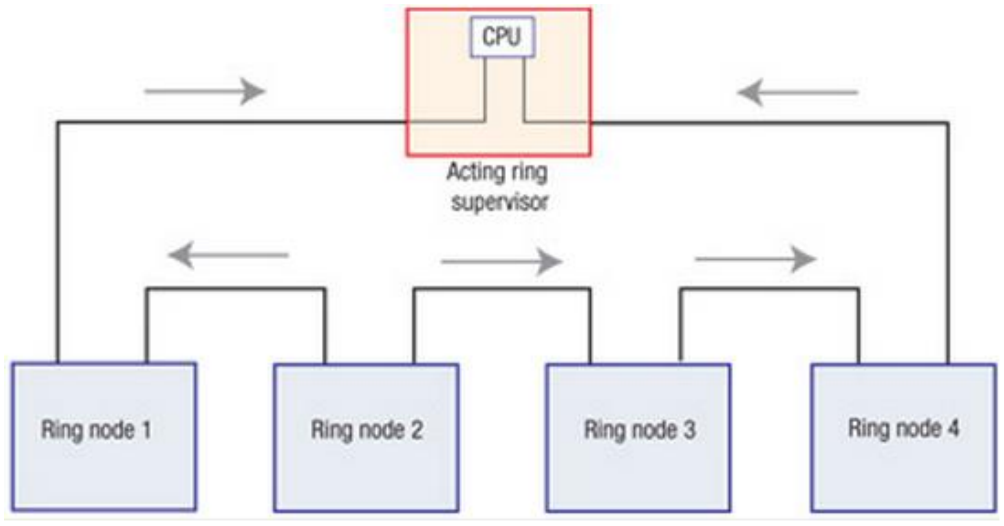


Figure 24 – DLR normal operation

Error! Reference source not found. illustrates the normal operation of a DLR network. As it is shown there, each node has two Ethernet ports, and is assumed to have implemented an embedded switch. When a ring node receives a packet on one of its Ethernet ports, it determines whether the packet needs to be received by the ring node itself (e.g., the packet has the node's MAC address) or whether the packet should be sent out on the node's other Ethernet port.

The active ring supervisor blocks traffic on one of its ports apart from few special frames and does not forward traffic from one port to other. Because of this configuration, a network loop is avoided and only one path exists between any two ring nodes during normal operation. The active ring supervisor transmits a Beacon frame through both of its Ethernet ports once per beacon interval (400µs by default). The active ring supervisor also sends Announce frames once per second.

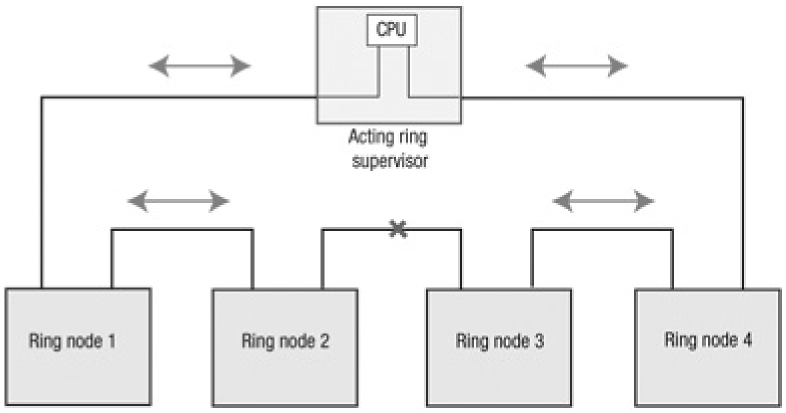
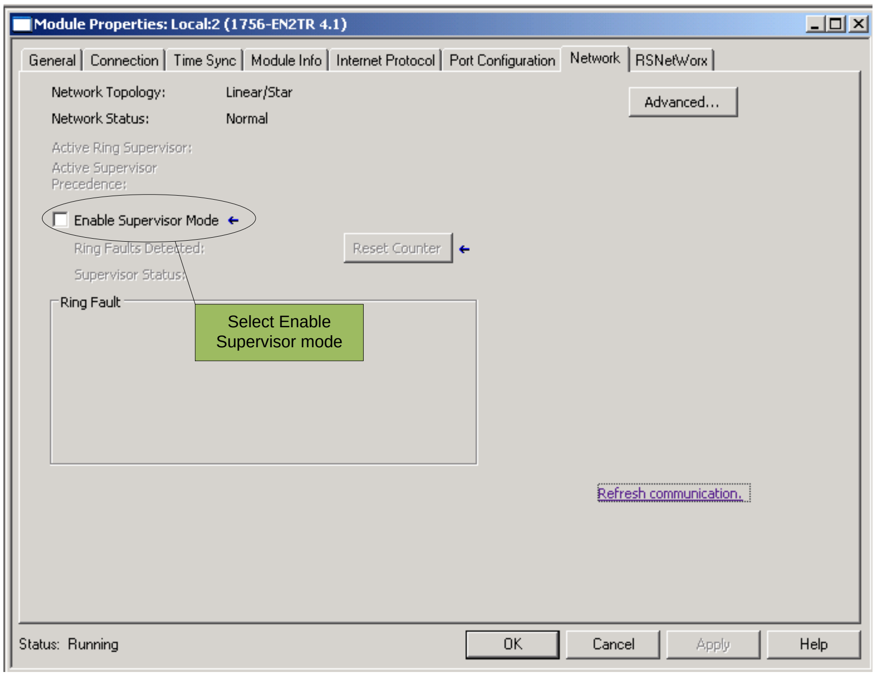


Figure 25 – DLR with a failed link

The DLR protocol is suitable for EtherNet/IP networks. A DLR network is tolerant to all single-point failures providing high network availability in a single-ring topology. The worst-case fault recovery time in a 50-node DLR network is less than 3ms.

6.3.5.1 Device Level Ring (DLR) Configuration

In a DLR configuration, an EN2TR communication card in the CPU CLX Rack is the preferred choice for the acting DLR supervisor. Configuring the Module properties of the communication card is done in the Network tab of the EN2TR configuration pane.


Module Properties: Local:2 (1756-EN2TR 4.1)

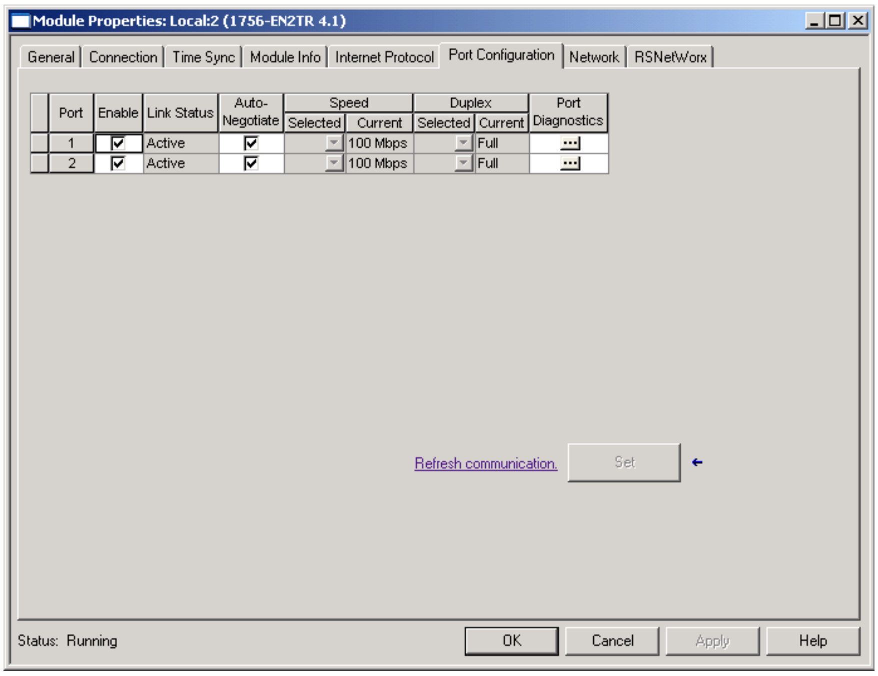
General | Connection | Time Sync | Module Info | Internet Protocol | Port Configuration | Network | RSNetWorx

Network Topology: Linear/Star
Network Status: Normal
Active Ring Supervisor:
Active Supervisor Precedence:
☐ Enable Supervisor Mode
Ring Faults Detected:
Supervisor Status:
Ring Fault

Status: Running

Figure 26 – Enable DLR in Network tab for DLR configurations

Once one supervisor mode has been selected, the network will establish the DLR ring and issue faults according the status. Both ports of the EN2T must be connected and active for the DLR to work.


Module Properties: Local:2 (1756-EN2TR 4.1)

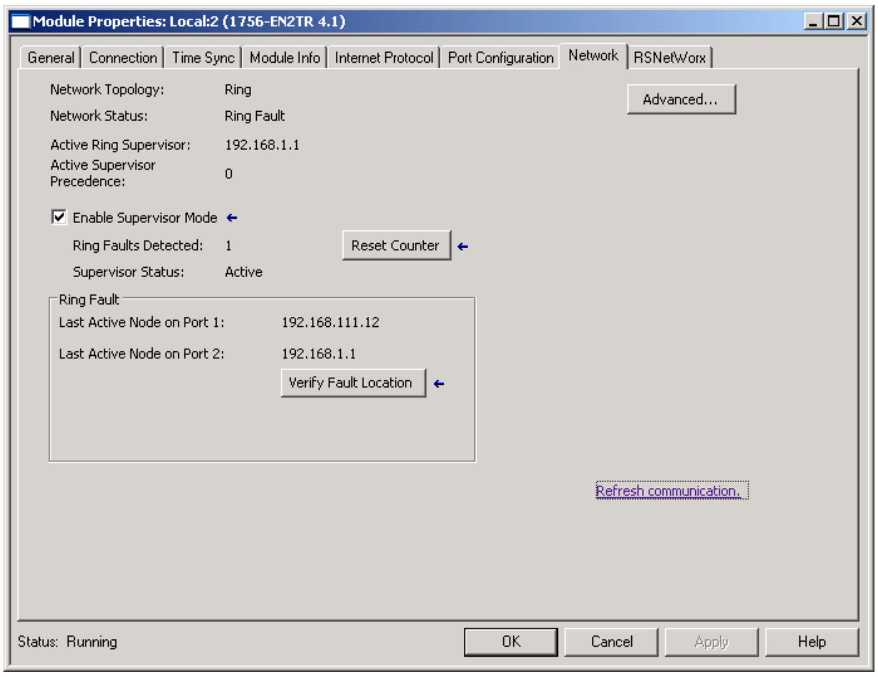
General | Connection | Time Sync | Module Info | Internet Protocol | Port Configuration | Network | RSNetWorx

Port	Enable	Link Status	Auto-Negotiate	Speed		Duplex		Port Diagnostics
				Selected	Current	Selected	Current	
1	✓	Active	✓		100 Mbps		Full	...
2	✓	Active	✓		100 Mbps		Full	...

Status: Running

Figure 27 – Both ports active for DLR operation

Once the supervisor node has been configured the remaining nodes can be configured and all faults cleared.


Module Properties: Local:2 (1756-EN2TR 4.1)

Network Topology: Ring
Network Status: Ring Fault
Active Ring Supervisor: 192.168.1.1
Active Supervisor Precedence: 0
☑ Enable Supervisor Mode
Ring Faults Detected: 1
Supervisor Status: Active

Ring Fault
Last Active Node on Port 1: 192.168.111.12
Last Active Node on Port 2: 192.168.1.1

Status: Running

Figure 28 – DLR has been selected and a fault issued.

6.3.5.2 ETAP Ethernet configuration

The IP addresses and naming for ETAP's follows the same standard as the other Ethernet modules. For configuration of the ETAP's refer to Rockwell Automation literature.

- Support scheduled traffic
- Save and distribute the network parameters and scheduling information

6.4 Analog module configuration

The analog modules shall be scaled via the PlantPAx scaling function in the SCADA system since that will result in easy troubleshooting and analyzing of the scaling for modules. Some modules offer/require internal scaling and shall be set as generic as practical and perform scaling in the P_Ain blocks.

7 Program Structure

A PLC application code is split up into tasks, programs, and routines. When designing a PLC application attention needs to be given to the number of tasks and programs within an application code. Since the way these items are used can affect the scan time of the PLC.

A PLC always reads code from left to right and from top to bottom therefore the structure of a PLC application is important for the code to execute in the right order. Avoid checking the same conditions many times. Each instruction adds scan time to the controller.

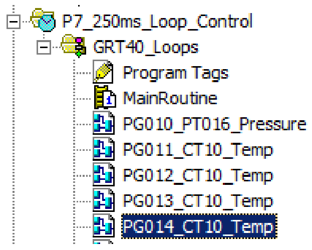
A task provides scheduling and priority information for a set of one or more programs. Tasks can be configured as either continuous, periodic, or event. A task contains programs, each with its own routines and program-scoped tags. Once a task is triggered (activated), all the programs assigned to the task execute in the order in which they are listed in the Controller Organizer.

Programs are useful to segregate areas or application zones and organize groups of routines that need to share a common data area

Routines contain the executable code. Each program has a main routine that is the first routine to execute within a program. Use logic, such as the Jump to Subroutine (JSR) instruction, to call other routines. Optional program fault routine can also be specified. An unlimited amount of routines can be used within a program.

- Tasks are used to configure controller execution
- Programs to group data and logic
- Routines to encapsulate executable code written in a single programming language

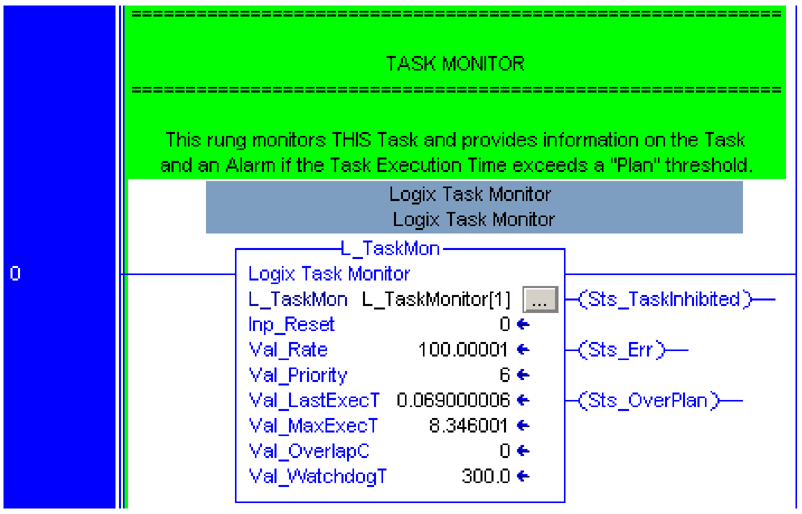
Periodic tasks shall be used for application specific tasks such as PID loop. These tasks shall state in their name according to the associated function, as shown in the picture below:



```
P7_250ms_Loop_Control
  GRT40_Loops
    Program Tags
    MainRoutine
    PG010_PT016_Pressure
    PG011_CT10_Temp
    PG012_CT10_Temp
    PG013_CT10_Temp
    PG014_CT10_Temp
```

Figure 29 – Loop Control

Periodic tasks shall have a task monitor AOI to monitor the operation of the task and display it in the SCADA system.

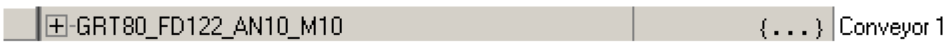


```
==============================================================
TASK MONITOR
==============================================================
This rung monitors THIS Task and provides information on the Task
and an Alarm if the Task Execution Time exceeds a "Plan" threshold.
Logix Task Monitor
Logix Task Monitor

0
L_TaskMon
Logix Task Monitor
L_TaskMon    L_TaskMonitor[1]  ...    (Sts_TaskInhibited)
Inp_Reset                  0
Val_Rate           100.00001          (Sts_Err)
Val_Priority               6
Val_LastExecT    0.069000006          (Sts_OverPlan)
Val_MaxExecT        8.346001
Val_OverlapC               0
Val_WatchdogT          300.0
```

Figure 30 – Task Monitor AOI

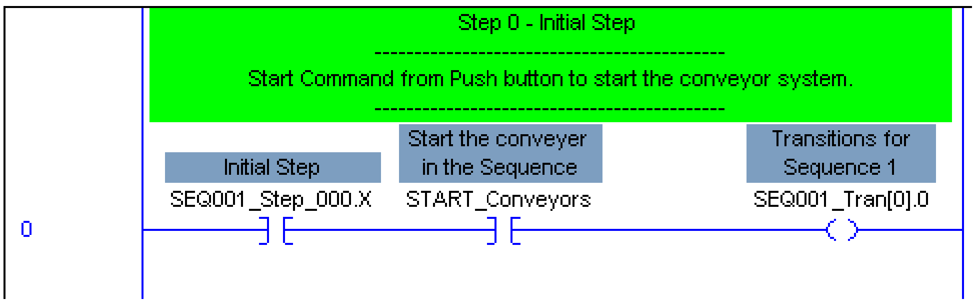
7.1 Comments and Descriptions

The use of descriptions for all items related to a PLC and a network application is mandatory. The owner reserves the right to reject any application or network configuration file that does not fulfill the owner's requirements on comments and descriptions. See below and example of descriptions that is useful to maintenance personnel.



```
+-GRT80_FD122_AN10_M10                {...} Conveyor 1
```

Figure 31 – Tag Comments



```
Step 0 - Initial Step
--------------------------------------------
Start Command from Push button to start the conveyor system.
--------------------------------------------
```

Figure 32 – Rung comments

7.2 Task Priorities

Each task in the controller has a priority level. A higher priority task (such as 1) interrupts any lower priority task (such as 15). The continuous task has the lowest priority and is always interrupted by a periodic or event task.

If a periodic or event task is executing when another is triggered, and both tasks are at the same priority level, the tasks will share execution time and the first task will run for 1ms and then the other will run for the same time. This interruption between the two tasks will continue until one of the tasks completes execution.

At the end of a task, the controller performs output processing for the output modules in your system. This processing depends on the number of output connections configured in the I/O tree.

If there are too many tasks, then;

- Tasks may experience overlaps. If a task is interrupted too frequently or too long, it may not complete its execution before it is triggered again.
- If a periodic task is set to run every 10 ms but the task takes 11ms to scan it will interrupt itself after 10 ms and will not scan the remainder of the Code
- Controller communication might be slower.
- If the application is designed for data collection, multiple tasks should be avoided. Switching between multiple tasks limits communication bandwidth.

7.3 Programming languages

In a Logix controller, there are four types of programming languages and their optimal use differs from one to the other.

- Ladder logic (LD)
    - Continuous or parallel execution of multiple operations (not sequenced)
    - Boolean or bit-based operations
    - Complex logical operations
    - Message and communication processing

  - Machine interlocking
  - Operations that service or maintenance personnel may have to interpret to troubleshoot the machine or process.
  - Servo motion control
- Function block diagram (FBD)
  - Loop control
- Sequential function chart (SFC)
  - High-level management of multiple operations
  - Repetitive sequences of operations
  - Batch process
- Motion control sequencing (via sequential function chart with embedded structure text)
  - State machine operations
- Structured text (ST)
  - Complex mathematical operations
  - ASCII string handling or protocol processing

For the Norðurál Site three language types may be used Ladder Logic, Function Block Diagrams, and Sequential function charts, but they can only be used for their recommended application usage as shown in the list above. In special cases, Structured Text can be used but only with the approval of owner.

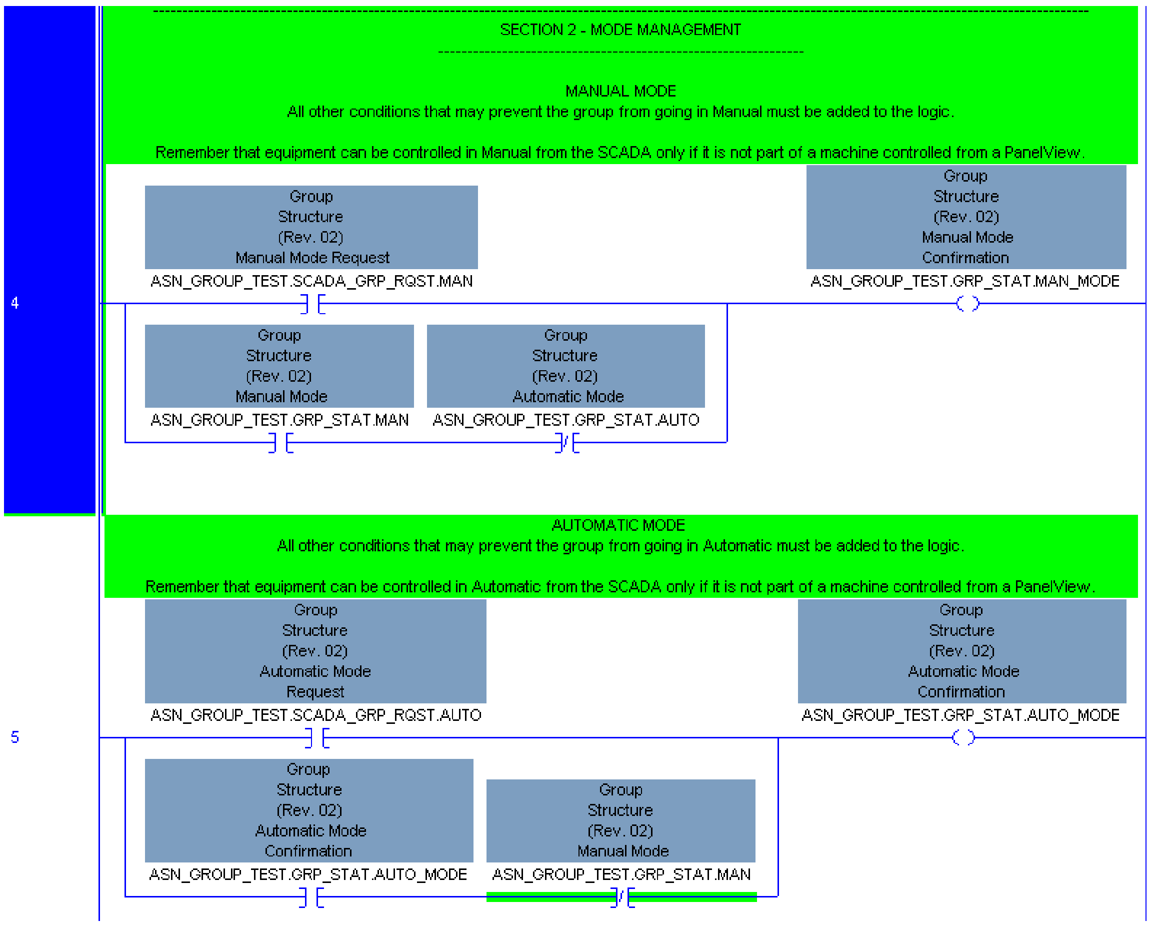
7.3.1 Ladder Logic

Ladder logic shall be the primary language used on the Norðurál Site since it is the simplest form of programming and easily read by maintenance personnel.

Description for all parameters and tags as well as rung section description is mandatory. Comments will be included for each section and routine header, as general function, and operation as standard. Each tag used shall also be commented as standard.

Device based programming is mandatory i.e. all equipment shall be in separate routines such as motors, valves, analog instruments etc. refer to **Error! Reference source not found.** for an example of the Ladder routine setup.

IMPORTANT: The use of the Latch and Unlatch instructions should be kept to a minimum since they will retain their state after power failure and are generally harder to debug.



SECTION 2 - MODE MANAGEMENT

MANUAL MODE
All other conditions that may prevent the group from going in Manual must be added to the logic.

Remember that equipment can be controlled in Manual from the SCADA only if it is not part of a machine controlled from a PanelView.

AUTOMATIC MODE
All other conditions that may prevent the group from going in Automatic must be added to the logic.

Remember that equipment can be controlled in Automatic from the SCADA only if it is not part of a machine controlled from a PanelView.

4
Group Structure (Rev. 02) Manual Mode Request
ASN_GROUP_TEST.SCADA_GRP_RQST.MAN
Group Structure (Rev. 02) Manual Mode Confirmation
ASN_GROUP_TEST.GRP_STAT.MAN_MODE
Group Structure (Rev. 02) Manual Mode
ASN_GROUP_TEST.GRP_STAT.MAN
Group Structure (Rev. 02) Automatic Mode
ASN_GROUP_TEST.GRP_STAT.AUTO
5
Group Structure (Rev. 02) Automatic Mode Request
ASN_GROUP_TEST.SCADA_GRP_RQST.AUTO
Group Structure (Rev. 02) Automatic Mode Confirmation
ASN_GROUP_TEST.GRP_STAT.AUTO_MODE
Group Structure (Rev. 02) Automatic Mode Confirmation
ASN_GROUP_TEST.GRP_STAT.AUTO_MODE
Group Structure (Rev. 02) Manual Mode
ASN_GROUP_TEST.GRP_STAT.MAN

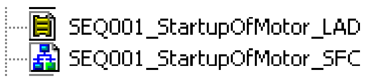
Figure 33 – Ladder Logic

7.3.2 Sequential Function Charts

SFC'S are the preferred method of programming for all kind of sequential functions and state machine management. The SFC's shall however always be used in combination with Ladder logic. All logical evaluations shall be externally handled by a Ladder program.

7.3.2.1 SFC Naming

Error! Reference source not found. shows an SFC routine and the associated ladder routine. The SFC shall have a descriptive name following a running sequence number. The Name shall be appended with a "_SFC". The associated ladder routine shall bear the same name, with an "_LAD" instead of the "_SFC"



SEQ001_StartupOfMotor_LAD
SEQ001_StartupOfMotor_SFC

Figure 34 – Routines to Control a Sequence

7.3.2.2 SFC Basics

SFC programming offers a graphical method of organizing the program. The four main components of an SFC are steps, actions, transitions, and the transition result. Steps are merely chunks of logic, i.e., a unit of programming logic that accomplishes a particular control task. Actions represent commands to external equipment associated with particular actuator or logic action. Transitions are the mechanisms used to move from one step to another. Control logic for each Step, Action and Transition is programmed in Ladder Diagram.

As a graphical language, SFC programming offers you several choices for executing a program, each depicted in a visually distinct way. In a sequential configuration, the processor simply executes the actions in step 1 repeatedly (command to open valve), until the transition logic becomes true (valve is open). The processor then proceeds to step 2.

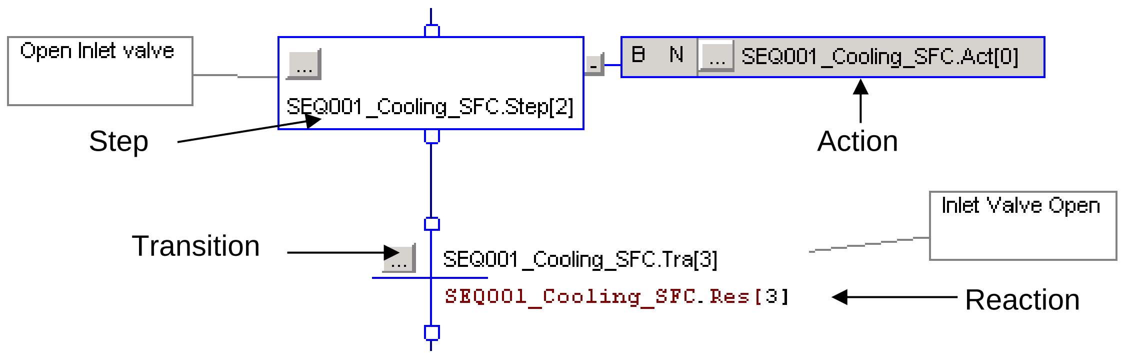

Open Inlet valve
SEQ001_Cooling_SFC.Step[2]
Step
B N SEQ001_Cooling_SFC.Act[0]
Action
Inlet Valve Open
Transition
SEQ001_Cooling_SFC.Tra[3]
SEQ001_Cooling_SFC.Res[3]
Reaction

Figure 35 – SFC elements Initial Step

All SFC begin in their initial steps after restart. The tag in the initial step shall always be _SFC.Step[0].

Care must be taken when doing online modifications to SFC routines. When changes are activated, the SFC is reset to the Initial step.

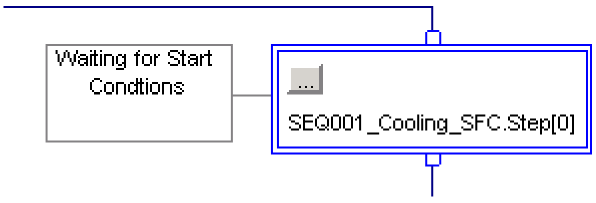


Figure 36 – Initial Step

7.3.2.3 Selective Branch

In a selective branch, only one branch is executed depending on which transition is active. In **Error! Reference source not found.**, a selection branch is shown. In every scan transitions T4, T5 and T6 are checked. The path belonging to the first transition to be active is chosen. If T5 is active first, then S5 is the next active step. If more than one transition is active within the same scan, the default priority shall be used, where the priority decreases from left to right and the leftmost sequence is executed.

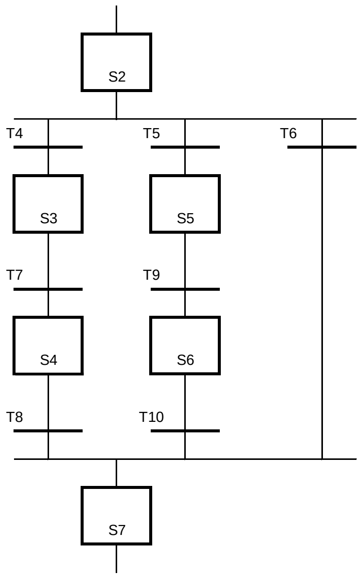


Figure 37 – Selective branch

7.3.2.4 Simultaneous Branch

In a simultaneous branch, all branches are executed until the transition becomes active. From **Error! Reference source not found.**, Steps S3, S5 and S7 are executed after S2/T6 is done. T9 is not tested until all paths are done; S4, S6 and S7 must be active.

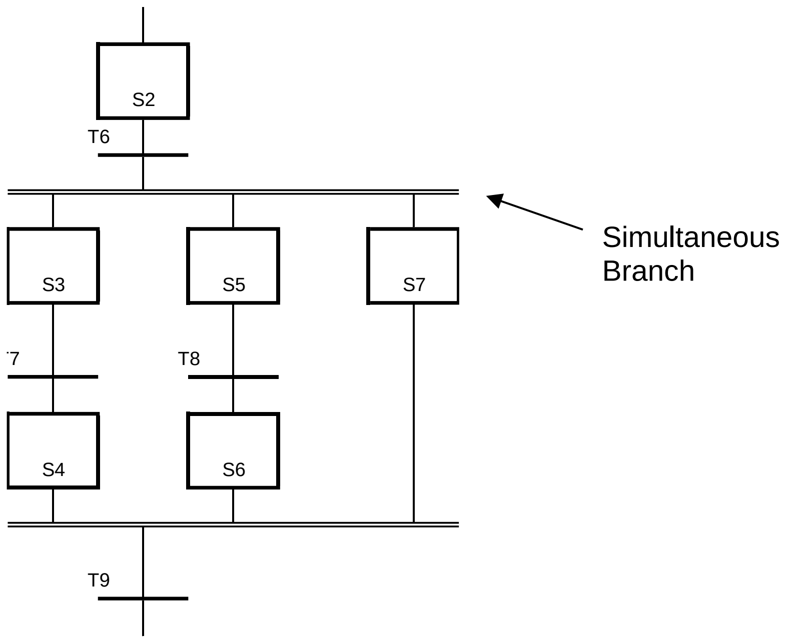


Figure 38 – Simultaneous branch

7.3.3 Function Block Diagram

Like SFC, FBD is a graphical language that allows programming in other languages (ladder, instruction list, or structured text) to be nested within the FBD. In FBD, program elements appear as blocks which are "wired" together in a manner resembling a circuit diagram. FBD is mostly useful in those applications involving a high degree of information/data flow between control components, such as process control.

IMPORTANT: On the Norðurál Site FBD shall only be used for loop control and the standard rules apply to tag names and descriptions.

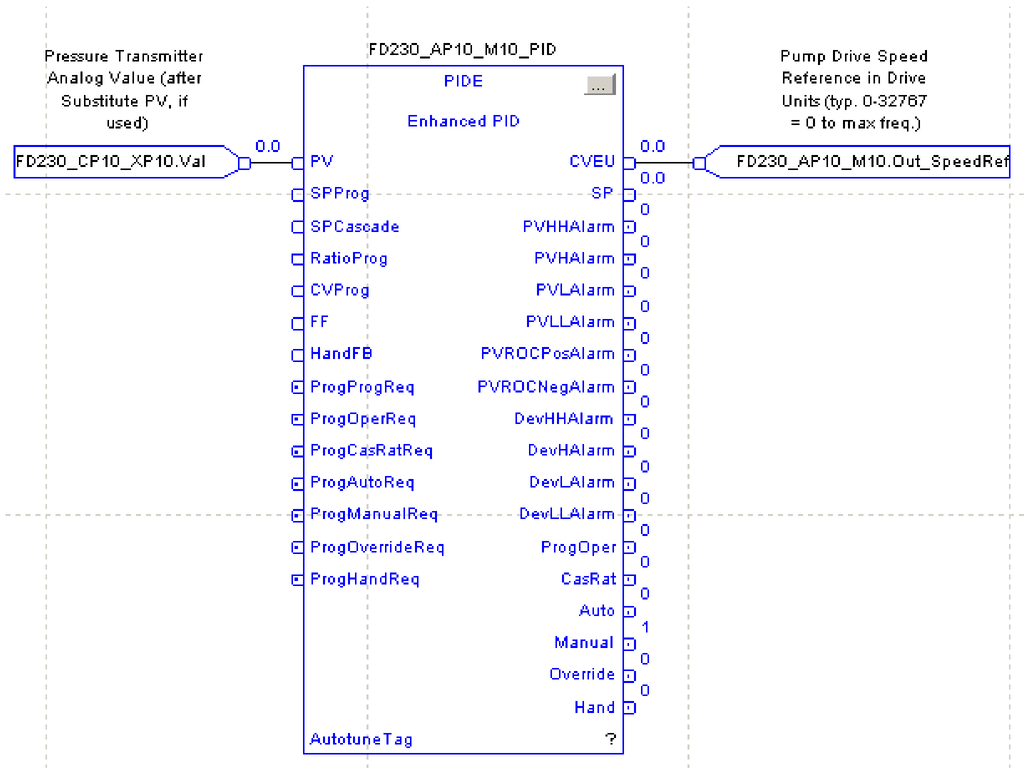


Figure 39 – Function Block Diagram

7.3.4 Start-up of PLC application code.

On transition to Run mode, the controller pre-scans logic to initialize instructions. The controller resets all state-based instructions, such as outputs (OTE) and timers (TON). Some instructions also perform operations during pre-scan. For example, the ONSR instruction turns off the storage bit. Note that latched bits (OTL) retains the state after power failure / Pre-scan/First Scan.

First scan differs from pre-scan in that the controller does not execute logic during prescan. The controller executes logic during first scan, the controller sets the S:FS bit for one scan:

- During the first scan that follows pre-scan.
- During the first scan of a program when it has been uninhibited.
- Each time a step is first scanned (when S:FS is set). You can view the S:FS bit being set only from the logic contained in actions that execute during the first scan of their parent step.

The S:FS bit should be used to ensure that when the PLC is put into run mode the equipment starts from a safe state.

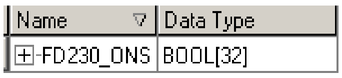
7.3.5 Arrays and User Defined data types (UDT)

User Defined data types shall be used as much as possible if a UDT is not applicable then arrays should be used this will reduce controller scan time and memory.

When a tag is created, the controller always sets aside at least 4 bytes (32 bits) of memory. The controller does this even if the tag needs only 1 bit.

When you create an array or a user-defined data type, the controller packs smaller data types into 4-byte (32-bit) words. This means the controller has less data to manipulate.

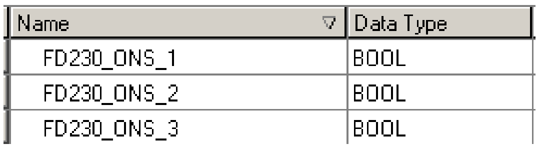
This array of 32 BOOLs takes only 4-bytes;



Name	Data Type
⊞-FD230_ONS	BOOL[32]

Figure 40 – Boolean array

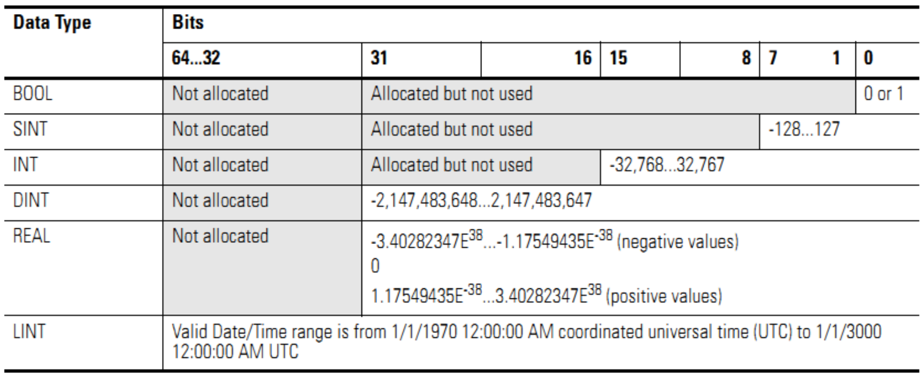
These 3 BOOL tags take 12 bytes total (3 tags x 4 bytes/tag 12 bytes);



Name	Data Type
FD230_ONS_1	BOOL
FD230_ONS_2	BOOL
FD230_ONS_3	BOOL

Figure 41 – Boolean tags

In addition to memory allocation for data, a tag uses additional memory in the controller. To manipulate SINT or INT data, the controller converts the values to DINT values, performs the programmed manipulation, and then returns the result to a SINT or INT value. This requires additional memory and execution time when compared to using DINT values for the same operation.



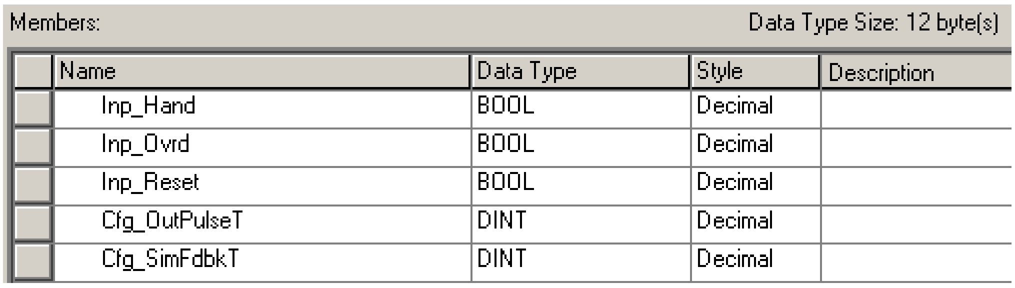
Data Type	Bits							
	64...32	31	16	15	8	7	1	0
BOOL	Not allocated	Allocated but not used						0 or 1
SINT	Not allocated	Allocated but not used				-128...127		
INT	Not allocated	Allocated but not used		-32,768...32,767				
DINT	Not allocated	-2,147,483,648...2,147,483,647						
REAL	Not allocated	$-3.40282347E^{38}$...$-1.17549435E^{-38}$ (negative values) 0 $1.17549435E^{-38}$...$3.40282347E^{38}$ (positive values)						
LINT	Valid Date/Time range is from 1/1/1970 12:00:00 AM coordinated universal time (UTC) to 1/1/3000 12:00:00 AM UTC							

Figure 42 – Data type memory allocation

UDT should be kept as compact as possible and like data types should be put together to save memory. Align all the BOOLs together.

- Align all the SINTs together.
- Put all the INTs together.
- Put all the REALs together.

Error! Reference source not found. shows how categorizing of data types saves memory. In this example, the data type takes up 12 bytes, since the BOOL's are put together;

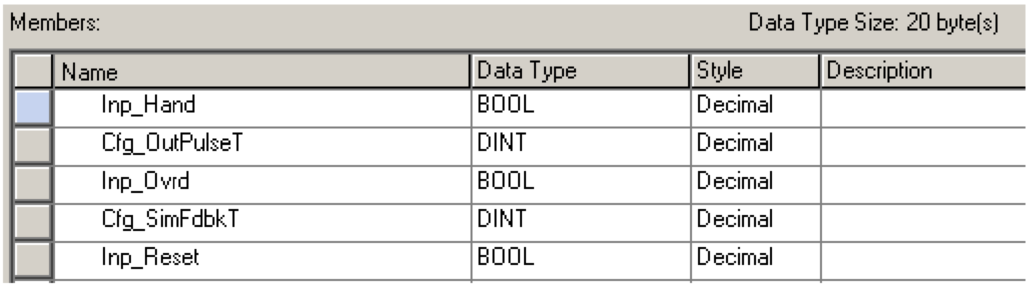


Members: Data Type Size: 12 byte(s)

Name	Data Type	Style	Description
Inp_Hand	BOOL	Decimal	
Inp_Ovrd	BOOL	Decimal	
Inp_Reset	BOOL	Decimal	
Cfg_OutPulseT	DINT	Decimal	
Cfg_SimFdbkT	DINT	Decimal	

Figure 43 – Recommended UDT Setup

Error! Reference source not found. shows the same data as before, but here the BOOL's are distributed. In this example, the data type takes up 20 bytes;

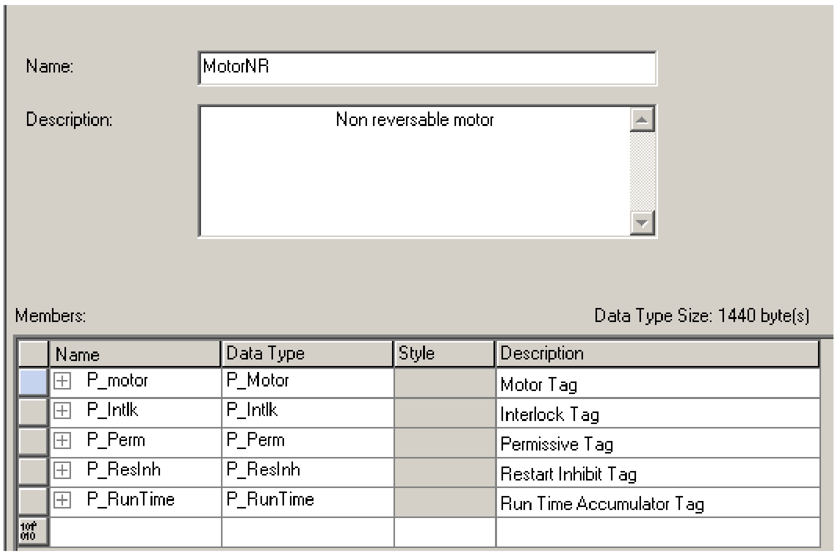


Members: Data Type Size: 20 byte(s)

Name	Data Type	Style	Description
Inp_Hand	BOOL	Decimal	
Cfg_OutPulseT	DINT	Decimal	
Inp_Ovrd	BOOL	Decimal	
Cfg_SimFdbkT	DINT	Decimal	
Inp_Reset	BOOL	Decimal	

Figure 44 – Not Recommended UDT Setup

UDT's can be reused as often as required. UDT can also contain AOI defined data types to group multiple AOI types together into a single tag. The data type shown in **Error! Reference source not found.** is for a non-reversible motor that has multiple "PlantPAx" AOI associated to it like the motor control, interlock, permissive etc.



Name: MotorNR

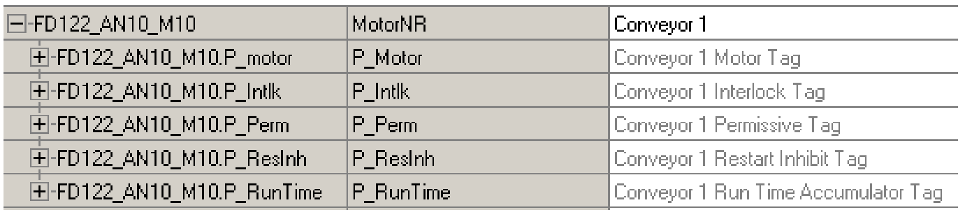
Description: Non reversable motor

Members: Data Type Size: 1440 byte(s)

Name	Data Type	Style	Description
P_motor	P_Motor		Motor Tag
P_Intlk	P_Intlk		Interlock Tag
P_Perm	P_Perm		Permissive Tag
P_ResInh	P_ResInh		Restart Inhibit Tag
P_RunTime	P_RunTime		Run Time Accumulator Tag

Figure 45 – Non-Reversible Motor UDT

This is done to enable a single tag name for all functions of the motor. The motor tag name could be as shown in **Error! Reference source not found.**.



FD122_AN10_M10	MotorNR	Conveyor 1
FD122_AN10_M10.P_motor	P_Motor	Conveyor 1 Motor Tag
FD122_AN10_M10.P_Intlk	P_Intlk	Conveyor 1 Interlock Tag
FD122_AN10_M10.P_Perm	P_Perm	Conveyor 1 Permissive Tag
FD122_AN10_M10.P_ResInh	P_ResInh	Conveyor 1 Restart Inhibit Tag
FD122_AN10_M10.P_RunTime	P_RunTime	Conveyor 1 Run Time Accumulator Tag

Figure 46 – Non-Reversible Motor Tag Structure

7.4 Add-On Instruction (AOI)

The PlantPAx library used for the Norðurál Site is built on AOI's and shall be used always when applicable. The use of user developed AOI is encouraged where the PlantPAx library does not apply to certain scenarios. The User developed AOI should be built in the same way as the PlantPAx AOI Using the standard UDT and AOI for alarms and other functions. This will ensure that the application code will be uniform.

When an AOI is created, it must follow the style and layout of the PlantPAx library. All developed AOI must be verified by owner, prior to use.

PlantPAx prefixes should be used to categorize parameters and tags.

PlantPAx Prefix	Description
Ack_	Alarm acknowledge (used with P_Alarm)
Alm_	Alarm (used with P_Alarm)
Cfg_	Configuration parameters
Err_	Error in configuration
Inp_	Input
MCmd	Maintenance Command
OCmd_	Operator command (set while in operator mode, or man)
Out_	Output
PCmd_	PLC command (set while in PLC mode, or auto)
PSet_	Program owner request
Rdy_	Ready
Sts_	Status
Val_	Values
Wrk_	Intermediate / Internal variable

Table 14 – PlantPAx prefixes

Only Ladder logic can be used inside an AOI.

PlantPAx provides various AOI for different application. Some of these AOI can work together for single equipment while others should be used individually. For example, does a digital input only need to be defined as P_Din AOI while a single speed motor could have interlock (P_Intl), permissive (P_Perm) runtime (P_RunTime) and restart inhibited (P_ResInh) associated.

The Norðurál library PLC example program provides many common UDT groups for easy grouping of functions.

The code within the AOI takes care of interaction between the Program, the SCADA/HMI, and the equipment.

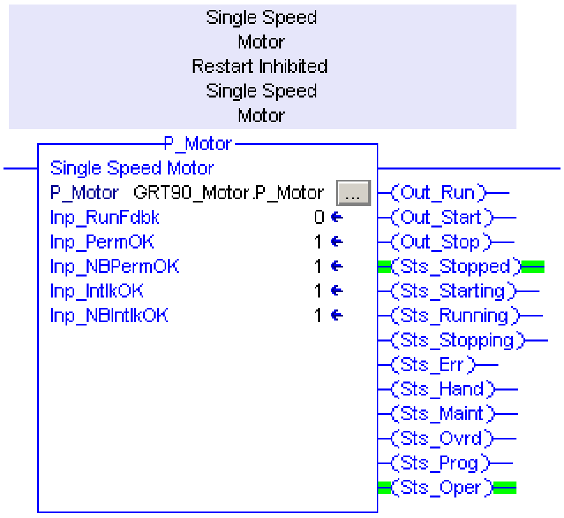
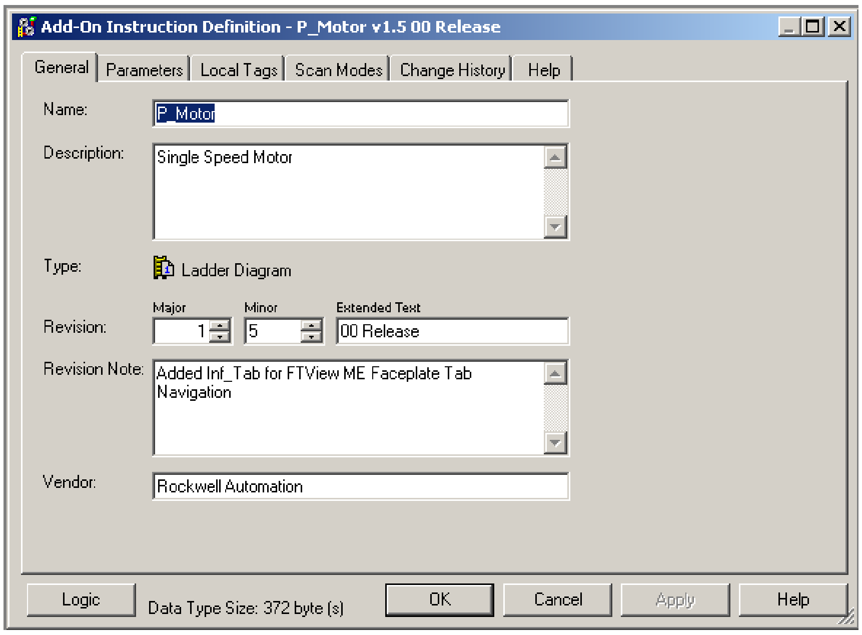


Figure 47 – Single Speed motor AOI

The Add-on Instruction is a self-standing code module, located in the Add-On Instruction folder of the RSLogix 5000 configuration tree. It is like a sub-routine, which is called via a line of code in a routine using the appropriate AOI instruction.

If AOI instruction included in the PLC Program Library does not match the application needed, custom AOI and routine can be developed following the standard. Custom AOI must be well documented and issued to the owner.

7.4.1 General tab



Name: P_Motor
Description: Single Speed Motor
Type: Ladder Diagram
Revision: Major 1 Minor 5 Extended Text 00 Release
Revision Note: Added Inf_Tab for FTView ME Faceplate Tab Navigation
Vendor: Rockwell Automation
Data Type Size: 372 byte (s)

Figure 48 – AOI definition

All parameters must be filled out properly.

Name: Descriptive name of the control element.

All user defined AOI created for the Norðurál Site shall start with the letters GRT_ followed by a short description.

Description: Description of the function and use of the AOI.

Type: The type must be Ladder Logic.

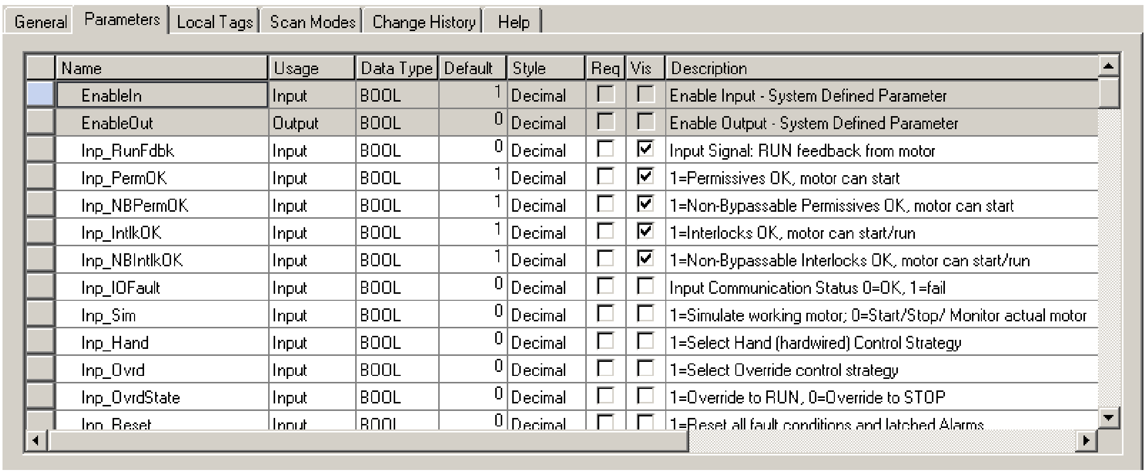
Revision: Revision numbers are essential to keep track of changes to the AOI

Revision note: Description of the changes made in in the current revision

Vendor: The Vendor name.

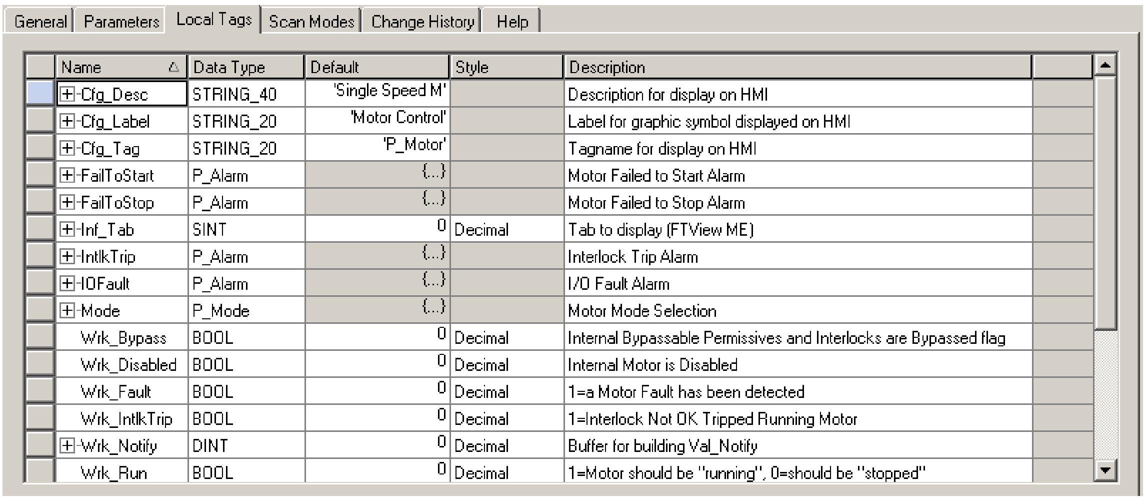
7.4.2 Parameters and tags in an AOI

An AOI can contain both Parameters and local tags. Parameters are used as outputs and inputs from SCADA and IO while local tags are used locally within the AOI.



Name	Usage	Data Type	Default	Style	Req	Vis	Description
EnableIn	Input	BOOL	1	Decimal	☐	☐	Enable Input - System Defined Parameter
EnableOut	Output	BOOL	0	Decimal	☐	☐	Enable Output - System Defined Parameter
Inp_RunFdbk	Input	BOOL	0	Decimal	☐	☑	Input Signal: RUN feedback from motor
Inp_PermOK	Input	BOOL	1	Decimal	☐	☑	1=Permissives OK, motor can start
Inp_NBPermOK	Input	BOOL	1	Decimal	☐	☑	1=Non-Bypassable Permissives OK, motor can start
Inp_IntlkOK	Input	BOOL	1	Decimal	☐	☑	1=Interlocks OK, motor can start/run
Inp_NBIntlkOK	Input	BOOL	1	Decimal	☐	☑	1=Non-Bypassable Interlocks OK, motor can start/run
Inp_IOFault	Input	BOOL	0	Decimal	☐	☐	Input Communication Status 0=OK, 1=fail
Inp_Sim	Input	BOOL	0	Decimal	☐	☐	1=Simulate working motor; 0=Start/Stop/ Monitor actual motor
Inp_Hand	Input	BOOL	0	Decimal	☐	☐	1=Select Hand (hardwired) Control Strategy
Inp_Ovrd	Input	BOOL	0	Decimal	☐	☐	1=Select Override control strategy
Inp_OvrdState	Input	BOOL	0	Decimal	☐	☐	1=Override to RUN, 0=Override to STOP
Inp_Reset	Input	BOOL	0	Decimal	☐	☐	1=Reset all fault conditions and latched Alarms

Figure 49 – AOI Parameters



Name	Data Type	Default	Style	Description	
⊞-Cfg_Desc	STRING_40	'Single Speed M'		Description for display on HMI	
⊞-Cfg_Label	STRING_20	'Motor Control'		Label for graphic symbol displayed on HMI	
⊞-Cfg_Tag	STRING_20	'P_Motor'		Tagname for display on HMI	
⊞-FailToStart	P_Alarm	{...}		Motor Failed to Start Alarm	
⊞-FailToStop	P_Alarm	{...}		Motor Failed to Stop Alarm	
⊞-Inf_Tab	SINT	0	Decimal	Tab to display (FTView ME)	
⊞-IntlkTrip	P_Alarm	{...}		Interlock Trip Alarm	
⊞-IOFault	P_Alarm	{...}		I/O Fault Alarm	
⊞-Mode	P_Mode	{...}		Motor Mode Selection	
Wrk_Bypass	BOOL	0	Decimal	Internal Bypassable Permissives and Interlocks are Bypassed flag	
Wrk_Disabled	BOOL	0	Decimal	Internal Motor is Disabled	
Wrk_Fault	BOOL	0	Decimal	1=a Motor Fault has been detected	
Wrk_IntlkTrip	BOOL	0	Decimal	1=Interlock Not OK Tripped Running Motor	
⊞-Wrk_Notify	DINT	0	Decimal	Buffer for building Val_Notify	
Wrk_Run	BOOL	0	Decimal	1=Motor should be "running", 0=should be "stopped"	

Figure 50 – AOI Local Tags

The tag names shall be in the same format as the PlantPAx tag naming structure starting with a short acronym followed by the tags function:

Description for all parameters and tags as well as rung description is mandatory.

Example:

Tag name	Usage	Data Type	Visibility	Description	Function
Inp_RunFdbk	Input	BOOL	Yes	Input Signal: RUN feedback from motor	Input from IO
Cfg_HasRunFdbk	Input	BOOL	No	1=Motor provides a run feedback signal	Configuration of the equipment function
PCmd_Start	Input	BOOL	No	Program Command to Start Motor	Commands for the equipment from a sequence or condition
OCmd_Start	Input	BOOL	No	Operator Command to Start Motor	Commands from SCADA / HMI
Out_Run	Output	BOOL	Yes	1=Run Motor, 0=Stop Motor	Output to IO
Val_Notify	Output	SINT	NO	Current Alarm Level and Acknowledgement (enumeration)	Numeric representation of the current alarm severity required for the ASN buttons
Sts_Stopped	Output	BOOL	Yes	1=Motor requested to stop and is confirmed stopped	Status of the equipment
Alm_FailToStart	Output	BOOL	No	1=Motor Fail to Start Alarm	Alarm representation to SCADA / HMI
Ack_FailToStart	Output	BOOL	No	1=Fail to Start Alarm has been acknowledged	Handshake to PLC that alarm has been acknowledged in SCADA / HMI
Rdy_Start	Output	BOOL	No	1=Ready to receive OCmd_Start (enables HMI button)	Equipment readiness

Table 15 – Parameter Tags

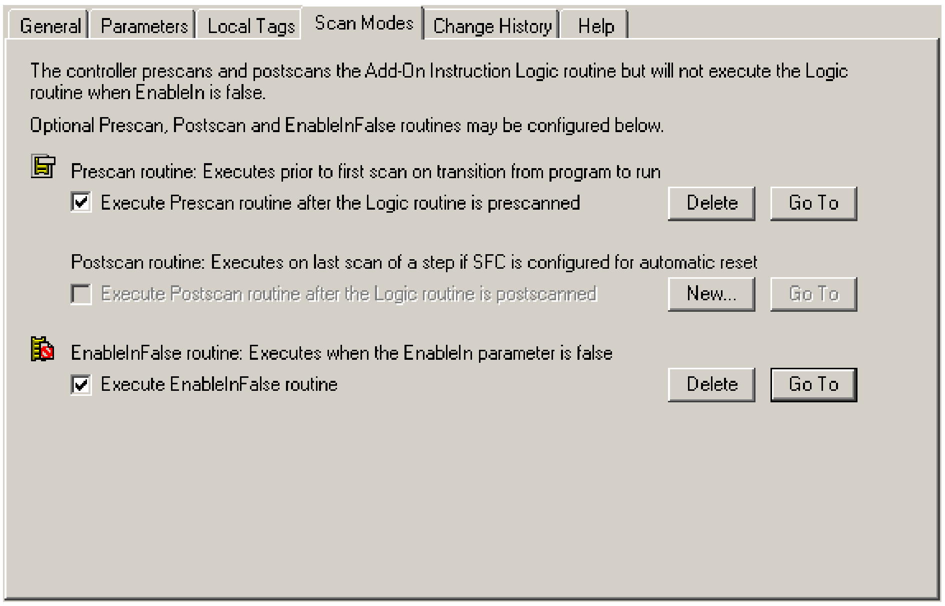
Local tags are created in the AOI for internal control of the AOI. These tags cannot be used in the PLC application code in the same way as the parameter tags. These local tags can tough be accessed from the SCADA system and must be accessed for retrieving the tag name and description. Then the tag name can be selected from the tag browser but the extension must be manually entered for example

- tagname.Cfg_Desc
- tagname.Cfg_Tag.

IMPORTANT: Tags of same data type should always be grouped together to save memory.

7.4.3 Scan modes

A Prescan in AOI should be used to reset command and status bits, intermediate variables etc. An Enable in False routine should also be created to handle cases when the code is not being executed. An example of these functions is shown in the PlantPAx library.


General | Parameters | Local Tags | Scan Modes | Change History | Help

The controller prescans and postscans the Add-On Instruction Logic routine but will not execute the Logic routine when EnableIn is false.

Optional Prescan, Postscan and EnableInFalse routines may be configured below.

Prescan routine: Executes prior to first scan on transition from program to run
[✓] Execute Prescan routine after the Logic routine is prescanned — Delete — Go To

Postscan routine: Executes on last scan of a step if SFC is configured for automatic reset
[ ] Execute Postscan routine after the Logic routine is postscanned — New... — Go To

EnableInFalse routine: Executes when the EnableIn parameter is false
[✓] Execute EnableInFalse routine — Delete — Go To

Figure 51 – AOI Scan Modes

The EnableInFalse routine will turn all outputs off and show the SCADA faceplate as disabled.

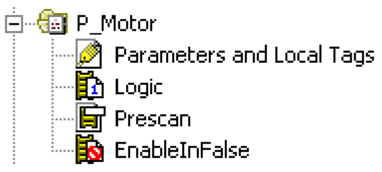
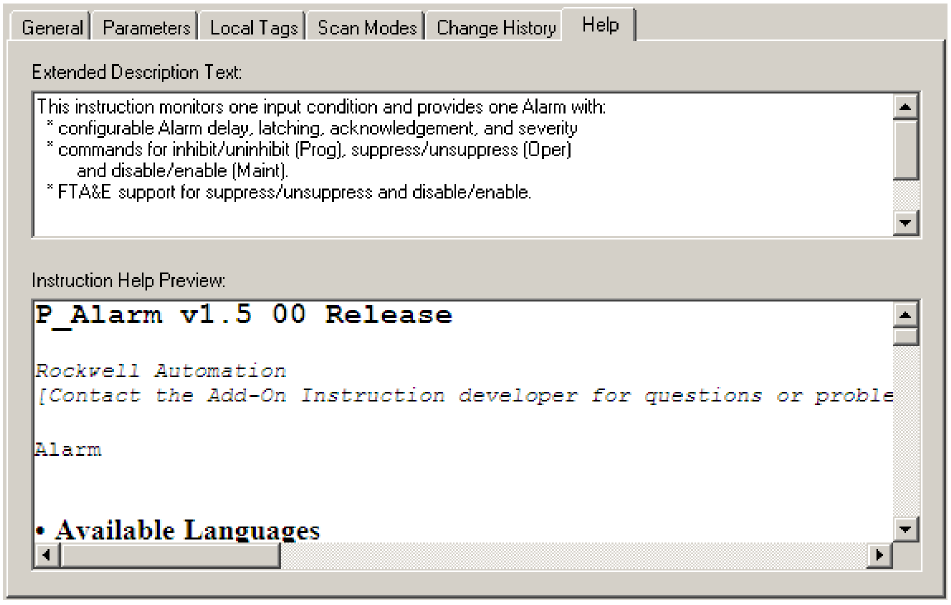


Figure 52 – PlantPAx Motor AOI

7.4.4 Help

The AOI help shall be filled out in a way that instructs a user or maintenance person on its use and function. The Help function will automatically create a help file for all tags and parameters in the AOI based on the description given. But an extended functional description of the AOI shall be provided by the vendor.


General | Parameters | Local Tags | Scan Modes | Change History | Help

Extended Description Text:

This instruction monitors one input condition and provides one Alarm with:
* configurable Alarm delay, latching, acknowledgement, and severity
* commands for inhibit/uninhibit (Prog), suppress/unsuppress (Oper)
  and disable/enable (Maint).
* FTA&E support for suppress/unsuppress and disable/enable.

Instruction Help Preview:

P_Alarm v1.5 00 Release

Rockwell Automation
[Contact the Add-On Instruction developer for questions or proble

Alarm

• **Available Languages**

Figure 53 – AOI Help Tab

7.5 Routine Structure

All routines shall be split up into multiple sections separated by a section description. The sections can for example be split up into input mapping, alarm, mode management, commands, output mapping etc. Section 1 should always contain status information in a rung with a NOP output for easy trouble shooting of the equipment. Input mapping shall always be done before the AOI for the equipment are called. The following sections shall contain the running conditions such as interlocks (P_Intlk), permissive (P_Perm) etc. with the applicable AOI. The equipment control AOI such as P_motor, P_valve etc. shall always be at the bottom of the routine followed by the output energizing.

Within a project the aim should be taken to set up all routines for the same equipment type in the same way with as many sections as required. This will increase readability of the PLC application and ease maintenance.

See the pilot project for more details and examples of section segregation.

7.6 The Norðurál programming library

The available pilot project contains ready to use routines for various purposes. The use of these routines and the associated UDT's is highly encouraged.

The use of these routines ensures similarity of programs throughout new projects and thus simplifies programming and debugging.

The following table lists the available examples at this point. As more projects are completed, additional items will be added.

UDT	Routine	Description
A_Out	PG011_AA30_M10_Spennir	Analog output usage routine with intlk.
Faults		General purpose UDT for Faults
Group	GRT60_StemGrind_Sys	Group control setup
Module_Valid	IO_Validation	Module validation routines
Motor_NonReversable	JF010_AE01_6M1_Grinder	Single direction motor control
Motor_R	JF010_AE03_6M3_Lift	Bi-directional motor control
Motor_VSD	PG010_AP11_DAELA_A	VFD Drive control
Msg_data	Communication	UDT for MSG communication
Prod_Cons	Communication	UDT for produced/consumed
SFC_32	SEQ001_PINBLASTER_SEQ SEQ001_PINBLASTER_LAD	UDT for SFC programming, 32 steps
SFC_64		UDT for SFC programming, 64 steps
Valve_MO		Motor operated valve
Valve_SO	JF010_AE01_6M1_Grinder	Solenoid operated valve
	PG010_PT016_PID PG010_PT016_PIDE	PIDE example
P_AIn	PG010_CP30	Analog input example

Table 16 – Pilot program routines.

7.7 Monitoring of Modules and Nodes

Monitoring of all modules in the application is vital for easy troubleshooting of the system in case of a fault in a module. This applies to all modules on all the three network types. A monitoring routine shall be created for all modules to give an alarm in the SCADA system of a faulty IO. All module alarms shall be of severity 4 so they will take priority of other alarms in the alarm summary since a module fault will cause multiple alarms relative to the module. The module alarms should be segregated to avoid unwanted alarms

When an IO module fails only an alarm for that module should appear and if a communication module fails the alarms for all modules connected to that module should be disabled but the safe state of the IO should always be guaranteed.

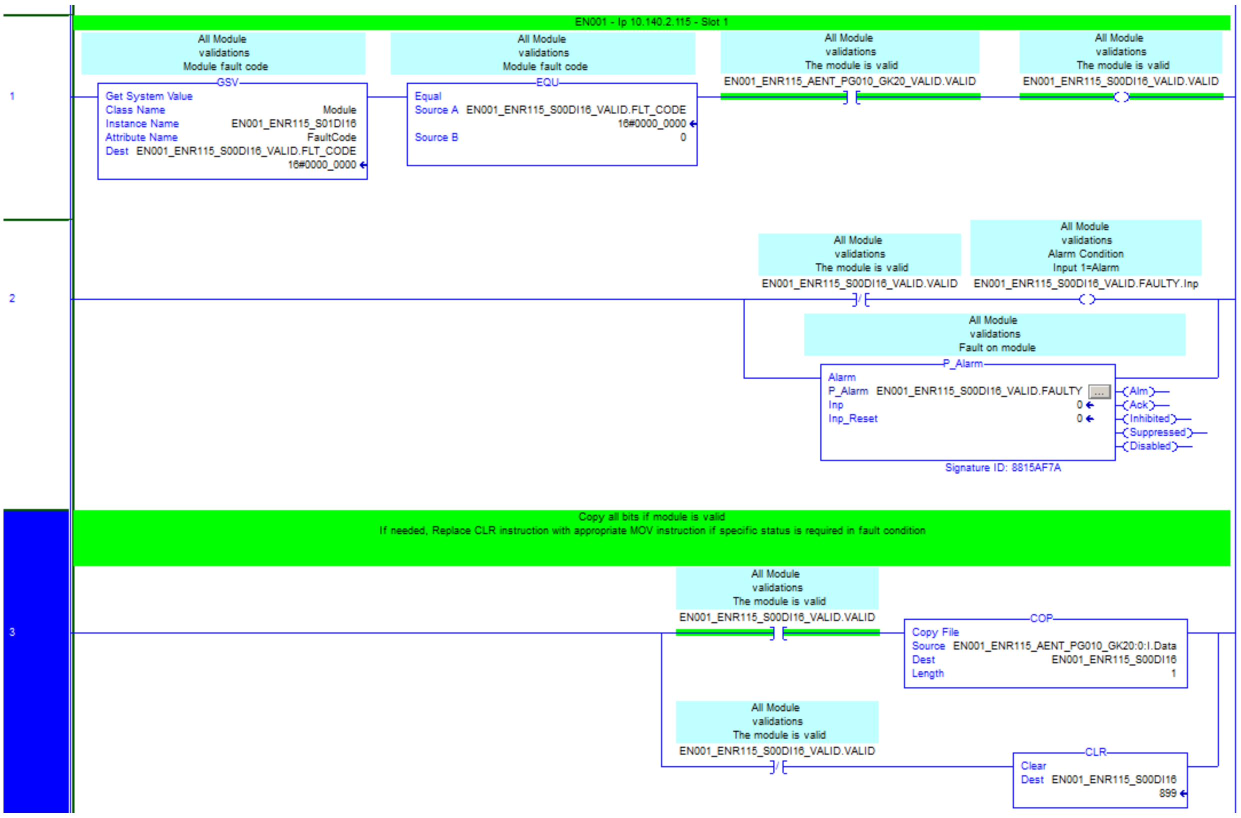
7.7.1 IO Validation

Because most of the input/output control signals are communicating over Ethernet network modules located inside the Local chassis, each I/O signal shall be validated in the PLC program when used. Here, "validated" means that the state of the I/O is confirmed. The methods used to validate transmitted information vary based on their origin. All modules and networks shall be supervised, and an alarm sent to the SCADA system in case of failure in the hardware

Any malfunction in the control system shall generate actions to safely protecting the system without affecting plant safety and integrity.

These principles shall be applied in accordance with the process and the groupings made during network design.

Error! Reference source not found. demonstrates the setup of the IO validation routine. Examples of the IO validation routines are in the pilot project.



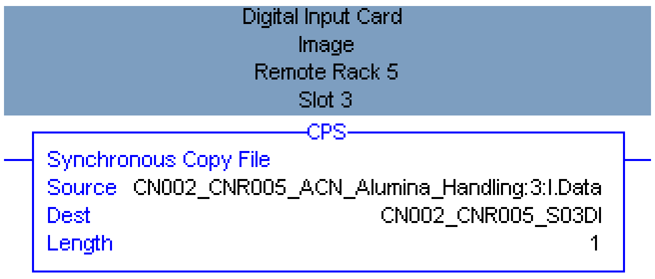
```
EN001 - Ip 10.140.2.115 - Slot 1

1   All Module validations Module fault code
    GSV
    Get System Value
    Class Name                                        Module
    Instance Name                        EN001_ENR115_S01DI16
    Attribute Name                                 FaultCode
    Dest  EN001_ENR115_S00DI16_VALID.FLT_CODE
                                             16#0000_0000 ←

    All Module validations Module fault code
    EQU
    Equal
    Source A  EN001_ENR115_S00DI16_VALID.FLT_CODE
                                             16#0000_0000 ←
    Source B                                             0

    All Module validations The module is valid
    EN001_ENR115_AENT_PG010_GK20_VALID.VALID   ] [

    All Module validations The module is valid
    EN001_ENR115_S00DI16_VALID.VALID   ( )

2   All Module validations The module is valid
    EN001_ENR115_S00DI16_VALID.VALID   ]/[

    All Module validations Alarm Condition Input 1=Alarm
    EN001_ENR115_S00DI16_VALID.FAULTY.Inp   ( )

    All Module validations Fault on module
    P_Alarm
    Alarm
    P_Alarm  EN001_ENR115_S00DI16_VALID.FAULTY  ...   (Alm)
    Inp                                       0 ←    (Ack)
    Inp_Reset                                 0 ←    (Inhibited)
                                                     (Suppressed)
                                                     (Disabled)
    Signature ID: 8815AF7A

    Copy all bits if module is valid
    If needed, Replace CLR instruction with appropriate MOV instruction if specific status is required in fault condition

3   All Module validations The module is valid
    EN001_ENR115_S00DI16_VALID.VALID   ] [
    COP
    Copy File
    Source  EN001_ENR115_AENT_PG010_GK20:0:I.Data
    Dest                    EN001_ENR115_S00DI16
    Length                                     1

    All Module validations The module is valid
    EN001_ENR115_S00DI16_VALID.VALID   ]/[
    CLR
    Clear
    Dest  EN001_ENR115_S00DI16
                         899 ←
```

Figure 54 – IO Validation and Input Image

7.7.2 Input Handling

As all inputs are read asynchronously by the controller it becomes important to map (copy) all physical inputs to internal tags, which are then used with a stable value inside the ladder code scan execution. This is done in the Input handling routine and the Synchronous Copy File (CPS) instruction used, see **Error! Reference source not found.**.

Input handling is mostly relevant for digital inputs as most analog inputs are mapped in a specific Analog input routine. See the pilot project for an example.


Digital Input Card
Image
Remote Rack 5
Slot 3

CPS
Synchronous Copy File
Source CN002_CNR005_ACN_Alumina_Handling:3:I.Data
Dest CN002_CNR005_S03DI
Length 1

Figure 55 – CPS Instruction

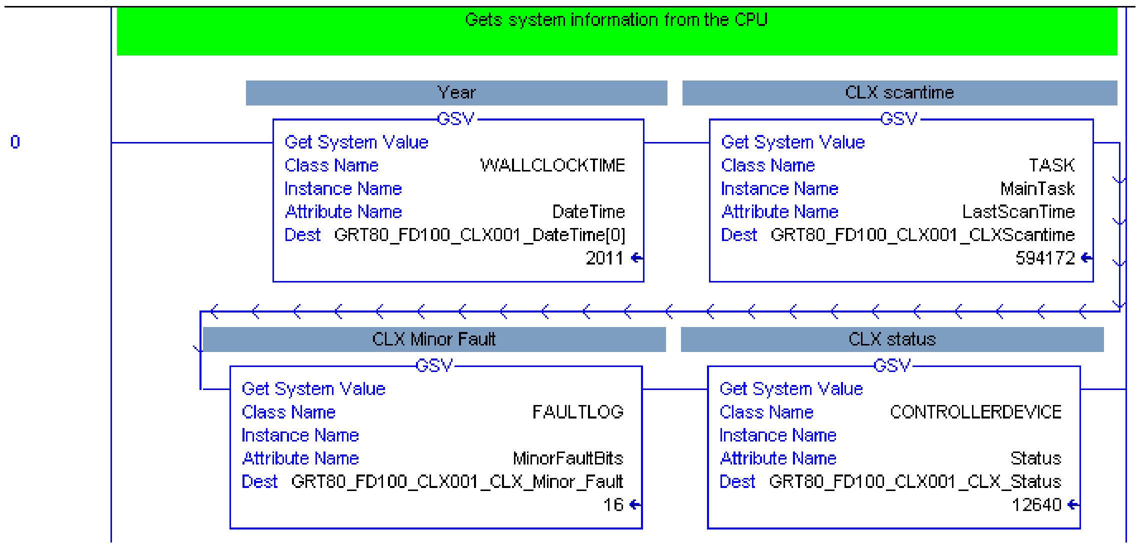
7.7.3 Outputs

The output modules are validated in the IO validation routine in the same way as the input modules. But the outputs do not have an output handling routine since all outputs are directly referenced in the equipment routines.

7.7.4 System Status / Utilities

A system status program is required for information on the controller and general system status. Alarms for all items that can result in a failure in the control system shall be created and displayed in the alarm summary in the SCADA system.

- Controller Status
- Battery status
- Clock
- IP addresses
- System status / Scada watchdog


Gets system information from the CPU

0

Year
GSV
Get System Value
Class Name WALLCLOCKTIME
Instance Name
Attribute Name DateTime
Dest GRT80_FD100_CLX001_DateTime[0]
2011

CLX scantime
GSV
Get System Value
Class Name TASK
Instance Name MainTask
Attribute Name LastScanTime
Dest GRT80_FD100_CLX001_CLXScantime
594172

CLX Minor Fault
GSV
Get System Value
Class Name FAULTLOG
Instance Name
Attribute Name MinorFaultBits
Dest GRT80_FD100_CLX001_CLX_Minor_Fault
16

CLX status
GSV
Get System Value
Class Name CONTROLLERDEVICE
Instance Name
Attribute Name Status
Dest GRT80_FD100_CLX001_CLX_Status
12640

Figure 56 – GSV from the Controller

In this routine, the CLX AOI should be created to give information on the CLX status such as faults, scan time, modes etc.

- L_CPU

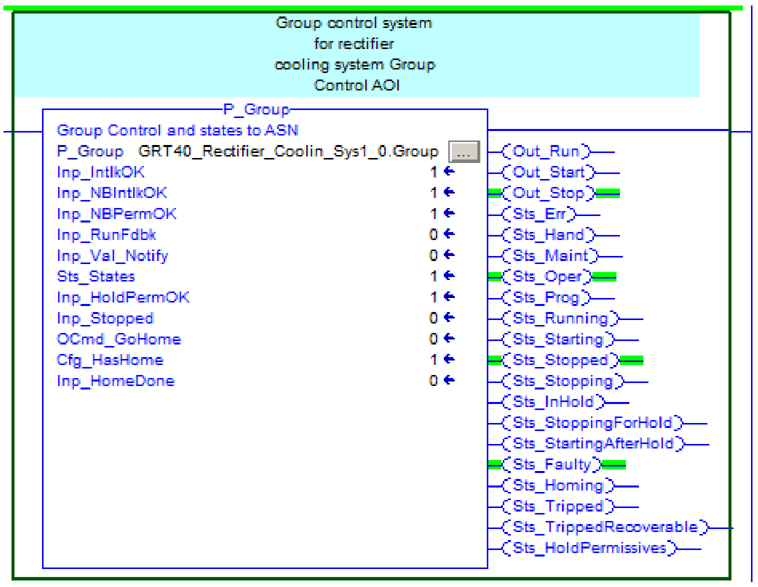
Information on fault codes can be found in the RSLogix5000 manuals and help files.

7.8 SCADA / HMI interfacing

Each AOI has a SCADA faceplate associated to it. This means that any communication from PLC to SCADA for a certain equipment is done through an AOI defined data type that makes connecting a faceplate in the SCADA system very user friendly.

7.8.1 Group control

A group routine shall always be created to group together items in a process line or a group. This group control is used for the ASN function of the SCADA system where all alarms and states are shown for a process. For this handling of information an AOI and a Group function ladder routine has been created. The AOI P_Group handles group functionality, see **Error! Reference source not found.**. The group control function shall enable a group of equipment to start, stop, hold, restart, or go to home position depending on the application requirements.



```
Group control system
for rectifier
cooling system Group
Control AOI

P_Group
Group Control and states to ASN
P_Group   GRT40_Rectifier_Coolin_Sys1_0.Group   ...
Inp_IntlkOK          1 ←     -( Out_Run )-
Inp_NBIntlkOK        1 ←     -( Out_Start )-
Inp_NBPermOK         1 ←     -( Out_Stop )-
Inp_RunFdbk          0 ←     -( Sts_Err )-
Inp_Val_Notify       0 ←     -( Sts_Hand )-
Sts_States           1 ←     -( Sts_Maint )-
Inp_HoldPermOK       1 ←     -( Sts_Oper )-
Inp_Stopped          0 ←     -( Sts_Prog )-
OCmd_GoHome          0 ←     -( Sts_Running )-
Cfg_HasHome          1 ←     -( Sts_Starting )-
Inp_HomeDone         0 ←     -( Sts_Stopped )-
                             -( Sts_Stopping )-
                             -( Sts_InHold )-
                             -( Sts_StoppingForHold )-
                             -( Sts_StartingAfterHold )-
                             -( Sts_Faulty )-
                             -( Sts_Homing )-
                             -( Sts_Tripped )-
                             -( Sts_TrippedRecoverable )-
                             -( Sts_HoldPermissives )-
```

Figure 57 – ASN Group AOI

The function of the group control can be split into multiple functions.

- Control of a process line or sequence is done through the group control function of the SCADA or ME system. From the group control popup, the process section can be operated, started, stopped, held, paused, or restarted.
- Statuses of the process group are programmed in the group control to indicate in which state the process group is in.
- Displaying alarms in the ASN section of the SCADA display. For this function, all alarms belonging to a process section are grouped together in the PLC as they would appear on a SCADA process screen containing the equipment. By doing this a field below the selection button for a process page will change color in accordance with the alarm severity.
- Interlocks and Permissive for the process group running criteria are shown.

Equipment that does not belong to any process group should be included into a group that has relevance to the equipment as seen from the SCADA display screen. Ensuring that when an alarm for that equipment is active it will appear on the ASN button bar for the page the equipment is located on. It shall however not have any interlock or control connection to the group.

IMPORTANT: Any equipment that is shown on a SCADA display shall belong to a group for the display it is located on, so the alarm functionality is guaranteed.

The name of the group control program shall always start with a short description followed by Sys and number. The number can be used to differentiate between similar named systems in a controller or multiple systems groups.

- Material_Handling_Sys1_0
- Airlift_Sys1_0
- PG011_Sys1_0
- PG011_Sys1_1

8 PLC Programming Standards

For the Norðurál Site the following standards for PLC programming shall apply. The use of AOI and UDT shall be used as much as possible by doing so a high standard of grouping is achieved. Comments will be included for each section and routine header, as general function, and operation as standard. Each tag used shall also be commented as standard.

NOTE: The usage of UDT with all instruments and/or equipment for a specific process or system is not indorsed by Norðurál. As this will complicate search for tags for that instrument or equipment.

8.1 Coding Standards

Coding shall be according to this document and the AKS coding system. All equipment tag names shall be as they are defined in the P&ID or drawings.

8.1.1 Attribute

An attribute is the lowest form of a data structure element within an object structure device, to represent the parameters of this device, such as VFD parameter, Sensor state or data, motor protector parameters.

Each attribute shall be defined by using standard extension code key word from the table as shown under Variable Extensions Codes section of this document **Error! Reference source not found.**. Also, see rules establish in the same section regarding tag naming convention.

Example:

- FD122_AN20_M10.**Inp_PermOK**(Where Inp_PermOK is the attribute for the permissive state of the motor).
- FD122_AN20_M10.**PCmd_Start** (Where PCmd_Start is the attribute for the program-controlled start).

8.1.2 Instance

An instance is a sub-structure UDT or AOI defined embedded in an object structure. A good practice is to create different instances (as needed) in which we regroup logically the attributes (parameters).

For example, I/O relative to a device (an object) like a Motor will have multiple AOI used to control the motor like interlock, permissive, motor control etc.

Example:

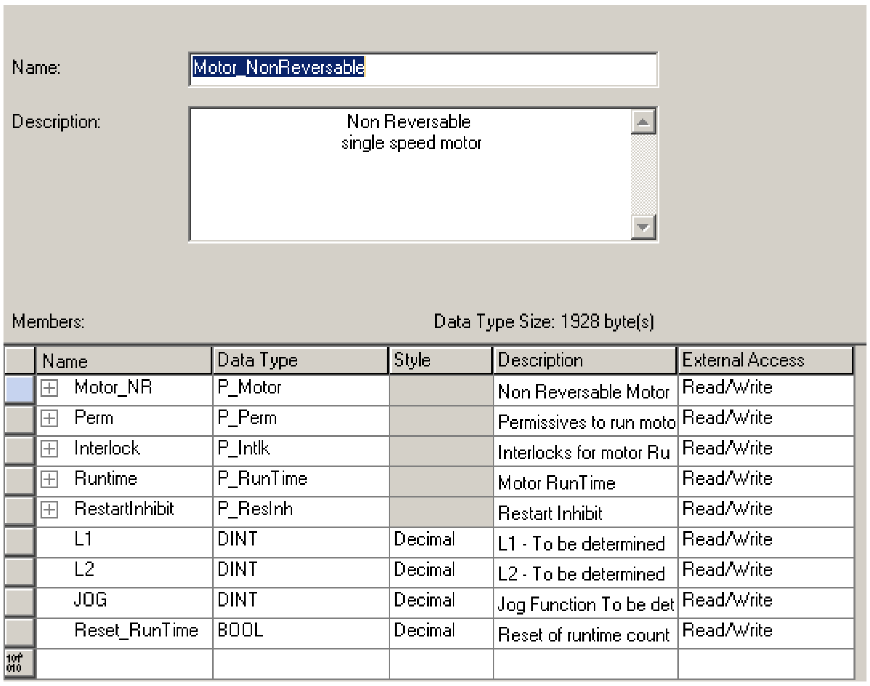
- FD122_AN20_M10.**P_Perm** (will be the instance UDT for the AOI that controls the permissive state of the motor)
- FD122_AN20_M10.**P_Intlk** (will be the instance UDT for the AOI that controls the interlock state of the motor)

8.1.3 Class

A class is a family (a category) of instances. Concept of Class is used to identify and facilitate the explanation/comprehension of these programming standards.

A device such as a motor has a class where multiple AOI have been grouped together to for a single tag.

Example:



Name: Motor_NonReversable

Description: Non Reversable single speed motor

Members: Data Type Size: 1928 byte(s)

Name	Data Type	Style	Description	External Access
⊞ Motor_NR	P_Motor		Non Reversable Motor	Read/Write
⊞ Perm	P_Perm		Permissives to run moto	Read/Write
⊞ Interlock	P_Intlk		Interlocks for motor Ru	Read/Write
⊞ Runtime	P_RunTime		Motor RunTime	Read/Write
⊞ RestartInhibit	P_ResInh		Restart Inhibit	Read/Write
L1	DINT	Decimal	L1 - To be determined	Read/Write
L2	DINT	Decimal	L2 - To be determined	Read/Write
JOG	DINT	Decimal	Jog Function To be det	Read/Write
Reset_RunTime	BOOL	Decimal	Reset of runtime count	Read/Write

Figure 58 – Motor non-reversible UDT

Examples of the class, instance and attribute structure can be seen bellow. Each class can have multiple instances and each instance can have multiple attributes.

Class	Instance	Attribute	Description
MotorNR	P_Motor	PCmd_Start	Start command issued from the program
		OCmd_Start	Start command issued from an operator
	P_Intlk	Sts_IntlkOK	Status if any interlock is active then motor stops
		PCmd_Reset	Reset command issued from the program
	P_Perm	Sts_Perm	INT value for individual permissive status
		Sts_BypActive	Permissive criteria are bypassed
	P_ResInh	Sts_Ready	Permissive for unit to start
		Inp_Stopped	Equipment is confirmed Stopped
	P_RunTime	Inp_Running	Motor is Running feedback from motor
		PCmd_ClearTotHrs	Program Command to Clear Total Running Time

Table 17 – Non-Reversible Motor Class

The tag name for an on reversible motor would for example be.

- FD122_AN10_M10.P_RunTime.Val_CurRunHrs
  - FD122_AN10_M10 is the class
  - P_RunTime is the Instance
  - Val_CurRunHrs is the attribute

8.2 Programming Standards

The following section describes and shows how RSLogix 5000 and ControlLogix shall be used with the previously described structure of object-oriented coding.

RSLogix 5000 shall be used to control the processor programming and control architecture in the following manner. For the Norðurál Site, a sample of device level structures UDT, Add-On instructions (AOI) and specific device control Routines was created in coordination to support typical Device type as VFD, Motor Starter and Transmitter. These items are included in the PLC

Program pilot project and shall be used as guidelines for the Vendors to develop their PLC program.

8.2.1 Device Control Module - Routine Template

Device Control Routines shall be used in the Norðurál Site and shall always be developed in ladder for typical devices, such as a VFD or Motor Soft Starter, Valve, and Transmitter. Each Device Control Routine shall include all the ladder code to take care of at least the following functions:

- Device Input mapping to UDT attributes
- HMI Message, Fault & Alarm warning associated with the device
- Interlocking of the device
- Status and Mode of operation control
- Starting and stopping of the device
- Activating real Device Output from UDT attributes

8.2.2 Power Up Handler Program

The power-up handler is an optional program that executes when the controller powers up in run or remote run mode. Use the power-up handler when you want to accomplish the following after restart of the PLC.

- When power is restored, take specific actions, and then resume normal operation:
- Clear the major fault and enter the logic for the actions
- Set (latch) or unset (unlatch) bit and bool typed tags.

8.2.3 Language and Comments

In the task, program and routine, everything shall be fully documented in such a way that any reader can understand it throughout the lifetime of the installation.

Every sequence should carry reference number of particular SEQ module or sub-module.

- Each line group forming a function shall be documented with rung comments
- Each instruction shall have a comment title
- Each sequence step, action and transition shall be documented
- Sequence step numbering shall be from top to bottom and left to right.
- Each sequence step shall have a Boolean Action and an indicator tag to clarify the step conditions.

All descriptions and tag names in the PLC application shall be in English.

8.3 Modes of Operation

An AOI has been created to control a Line or a group of equipment's called Group Control. This function will be used to control a process line or an equipment section starting, stopping, restarting, holding etc.

The modes of operation shall allow a system to:

- Reach the right operating conditions before production
- If possible, stay in production during abnormal conditions

- Be stopped then re-started
- Be stopped in clean and secure manner in all situations
- Automatically reset itself after an abnormal condition or failure and during PLC start-up.

8.3.1 Control Modes

The equipment Add-On instructions use the following standard Modes implemented using an embedded P_Mode Add-On instruction.

Control Mode	Description
Operator	The operator starts and stops the motor using the HMI / SCADA Faceplate.
Program	Logic outside the AOI starts and stops the equipment using Program Commands (PCmd_Start, PCmd_Stop).
Override	Instruction or strategy is driven to an override state based on exceptional process conditions or other logic above and beyond normal operation.
Maintenance	Instruction or strategy is removed from normal production and is reserved for commanding by Maintenance personnel.
Hand	Instruction or strategy is controlled by hardwired logic and cannot be manipulated by or through the controller.

Table 18 - Control Modes

The modes have the following priority:

- Hand (highest)
- Maintenance
- Override
- Operator and Program (lowest)

8.3.2 Control Program Architecture

Line, unit, and equipment is a group of machines, equipment’s or control device within a common area that produce a common product or process outcome. These line, unit and equipment are also named group.

An example of this structure is shown below where line 1 can contain multiple Units and each unit can contain many equipment units.

- Line 1
  - Unit 1
    - Equipment 1.1
    - Equipment 1.n+1
  - Unit 2
    - Equipment 2.1
    - Equipment 2.n+1
- Line 2

It shall always be done in a way that any equipment can be started before a line or a unit and the equipment shall resume operation when the line is started.

8.4 SFC Sequence

Normally an SFC sequence controls the sequence of events within a machine or operation. The SFC is the routine that controls and issues command to the equipment of a Group according to the mode of operations. Depending on the process involved, it can be unique or separated into many SFCs as shown below:

- Start-up and preparation (Initialization) SFC sequence
    - Start a group of equipment in a proper order, for example series of conveyors.
        - Example: SEQ001_Cooling_Startup_SFC
- Normal operation SFC sequence (can be multiple SFC's)
        - Example: SEQ001_Cooling_SFC
- Shutdown and closure (Stop) SFC sequence due to breakdown or aborting SFC sequence
    - Stop a group of equipment in a proper order, for example series of conveyors.
        - Example: SEQ001_Cooling_Stopping_SFC
- Homing Sequence
    - Ensures an automatic safe return of a machine to its initial position.
        - Example: SEQ001_Cooling_Homing_SFC

8.4.1 SFC Management in the _LAD routine

Logically, an SFC is controlled by the Group System Control. The group controls the state (e.g. running, reset, and pause) of the SFC routine. Commands from HMI's, other program parts, actuator status and sensors act as inputs to the sequence controlled by the SFC. SFC issue commands for actuators, other program parts and give statuses to HMI according to the state, see the I/O chart on **Error! Reference source not found.**.

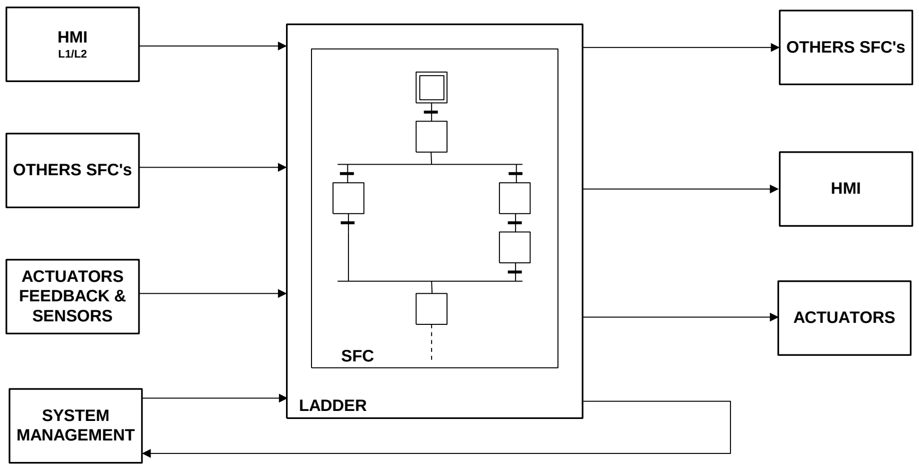


Figure 59 – SFC Input / Output Chart

Group system management provides three commands that control the running state of the SFC routine.

- Execute
    - Allows execution of SFC and stepping through transitions

- Pause
  - When group enters “Hold” State the SFC is “Paused”
  - In pause, transitions are blocked, until the sequence is restarted.
- Reset
  - When Group System control issues reset, the SFC is reset to the initial state
  - Reset to any other step than initial is not allowed.

The SFC routine is called inside the _LAD routine that is described below, just after the operation modes of the SFC issued in the same order as shown on **Error! Reference source not found.**.

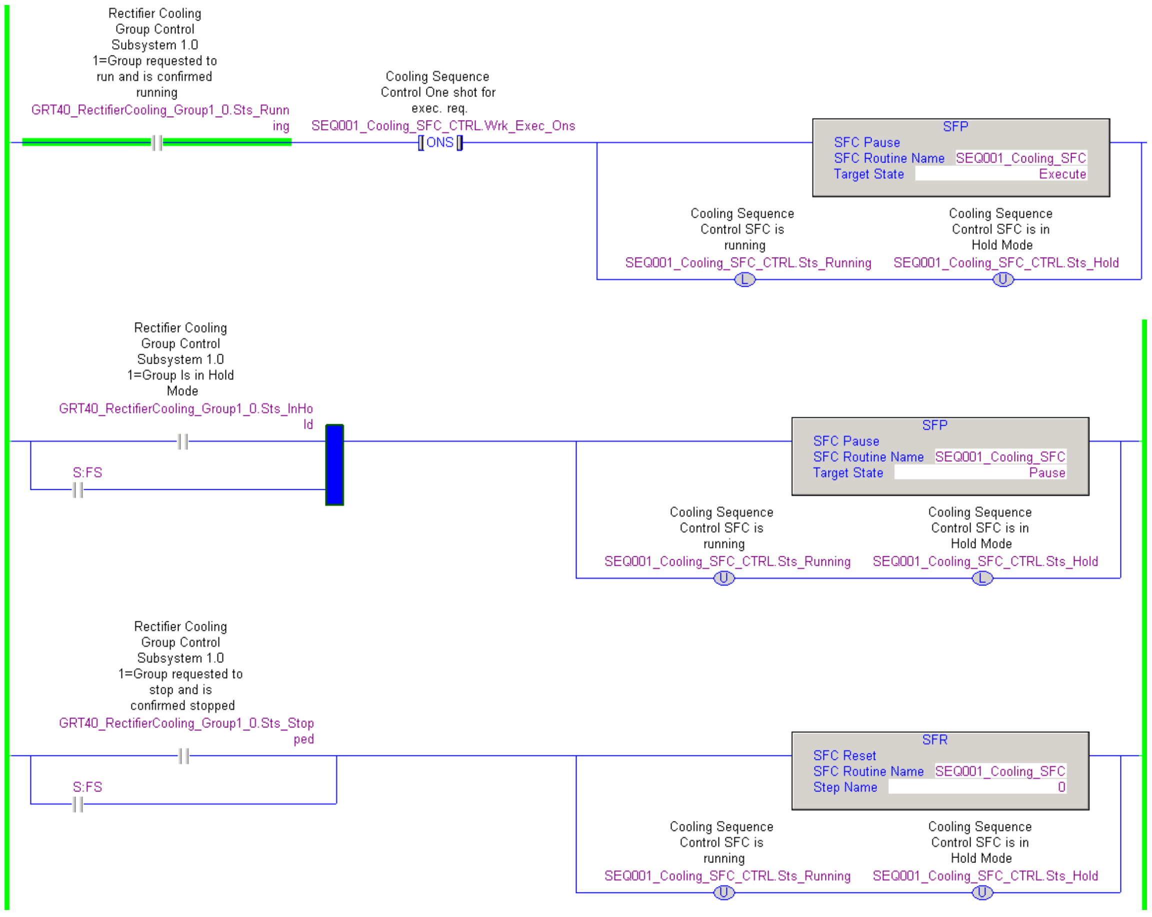
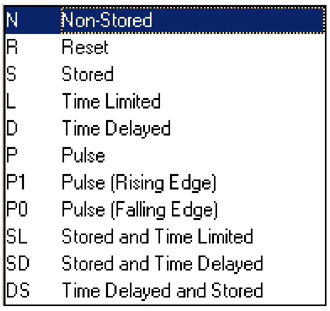

Rectifier Cooling Group Control Subsystem 1.0 1=Group requested to run and is confirmed running
GRT40_RectifierCooling_Group1_0.Sts_Running
Cooling Sequence Control One shot for exec. req.
SEQ001_Cooling_SFC_CTRL.Wrk_Exec_Ons
ONS
SFP
SFC Pause
SFC Routine Name SEQ001_Cooling_SFC
Target State Execute
Cooling Sequence Control SFC is running
SEQ001_Cooling_SFC_CTRL.Sts_Running
L
Cooling Sequence Control SFC is in Hold Mode
SEQ001_Cooling_SFC_CTRL.Sts_Hold
U
Rectifier Cooling Group Control Subsystem 1.0 1=Group Is in Hold Mode
GRT40_RectifierCooling_Group1_0.Sts_InHold
S:FS
SFP
SFC Pause
SFC Routine Name SEQ001_Cooling_SFC
Target State Pause
Cooling Sequence Control SFC is running
SEQ001_Cooling_SFC_CTRL.Sts_Running
U
Cooling Sequence Control SFC is in Hold Mode
SEQ001_Cooling_SFC_CTRL.Sts_Hold
L
Rectifier Cooling Group Control Subsystem 1.0 1=Group requested to stop and is confirmed stopped
GRT40_RectifierCooling_Group1_0.Sts_Stopped
S:FS
SFR
SFC Reset
SFC Routine Name SEQ001_Cooling_SFC
Step Name 0
Cooling Sequence Control SFC is running
SEQ001_Cooling_SFC_CTRL.Sts_Running
U
Cooling Sequence Control SFC is in Hold Mode
SEQ001_Cooling_SFC_CTRL.Sts_Hold
U

Figure 60 – SFC management in SEQ001_Cooling_LAD

The first rung shows the Group control issuing an execute command when entering “Running” state. The second rung shows the group control issuing a Pause command when entering “Hold” State. The third rung resets the SFC when the system has stopped for any reason, for example a non-recoverable fault or operator stop command.

8.4.2 Actions

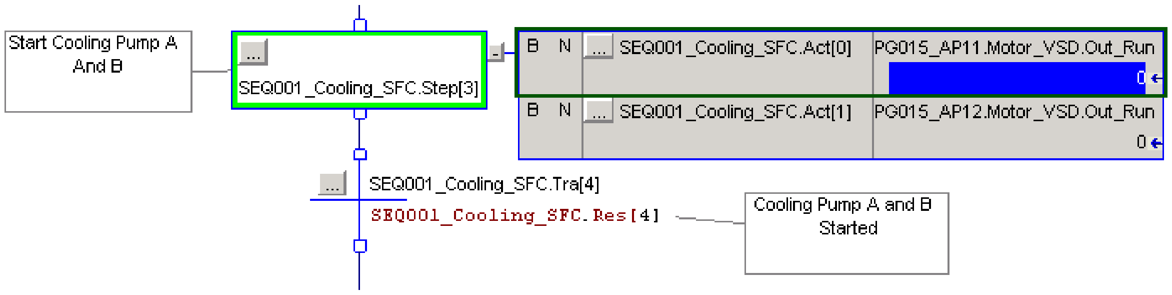
The operation of individual actions within a step can be varied with the use of action qualifiers. On the Norðurál Site all actions shall be of type N – Non-Stored and other logic if needed shall be performed in the associated ladder and not in the SFC.



N	Non-Stored
R	Reset
S	Stored
L	Time Limited
D	Time Delayed
P	Pulse
P1	Pulse (Rising Edge)
P0	Pulse (Falling Edge)
SL	Stored and Time Limited
SD	Stored and Time Delayed
DS	Time Delayed and Stored

Figure 61 – Action Qualifier

For each step the actions that make the associated transition true shall be visible as non-stored Boolean. If a step starts a motor the motor running indication shall be visible next to the step. This will enhance readability of the SFC code, see **Error! Reference source not found.**.



Start Cooling Pump A And B

SEQ001_Cooling_SFC.Step[3]

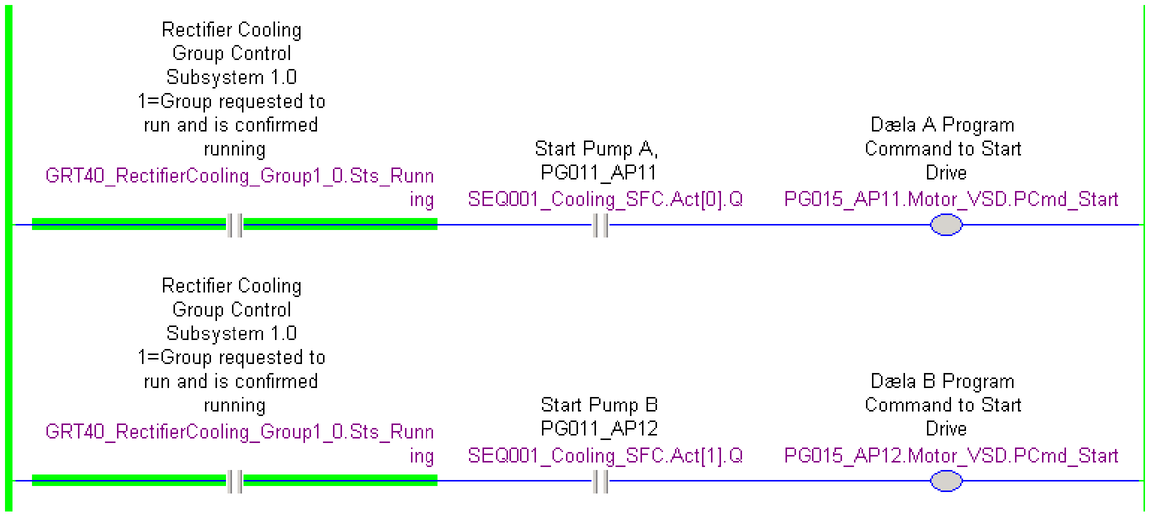
B	N	SEQ001_Cooling_SFC.Act[0]	PG015_AP11.Motor_VSD.Out_Run 0
B	N	SEQ001_Cooling_SFC.Act[1]	PG015_AP12.Motor_VSD.Out_Run 0

SEQ001_Cooling_SFC.Tra[4]
SEQ001_Cooling_SFC.Res[4]

Cooling Pump A and B Started

Figure 62 – Actions and reaction indication in an SFC

The Actions shall always have a corresponding rung in the _LAD routine as shown in **Error! Reference source not found.** and **Error! Reference source not found.**.



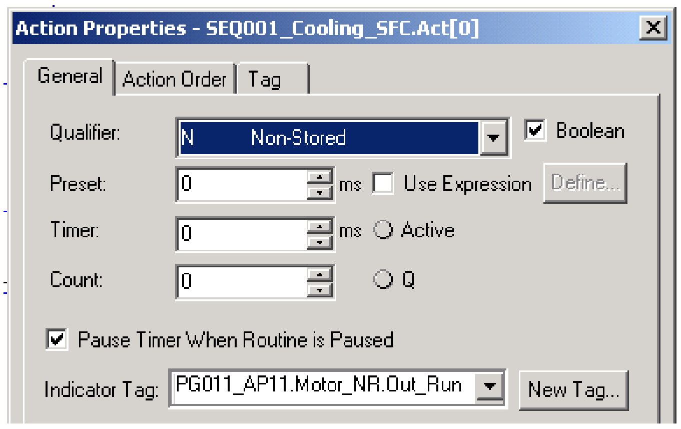
Rectifier Cooling Group Control Subsystem 1.0 1=Group requested to run and is confirmed running
GRT40_RectifierCooling_Group1_0.Sts_Running
Start Pump A, PG011_AP11
SEQ001_Cooling_SFC.Act[0].Q
Dæla A Program Command to Start Drive
PG015_AP11.Motor_VSD.PCmd_Start

Rectifier Cooling Group Control Subsystem 1.0 1=Group requested to run and is confirmed running
GRT40_RectifierCooling_Group1_0.Sts_Running
Start Pump B PG011_AP12
SEQ001_Cooling_SFC.Act[1].Q
Dæla B Program Command to Start Drive
PG015_AP12.Motor_VSD.PCmd_Start

Figure 63 – Corresponding action in _LAD Routine

The settings for each action are configured as shown on **Error! Reference source not found.**, where the indicator tag has been added. Furthermore, the Non-Stored Qualifier has been selected.

<u>IMPORTANT:</u> Ticking the “Pause Timer When Routine is Paused” is highly recommended.



Action Properties - SEQ001_Cooling_SFC.Act[0]

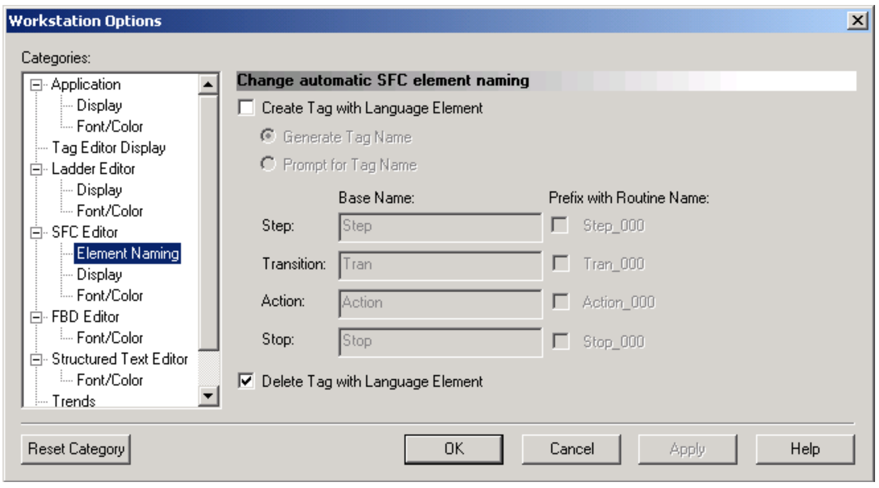
General | Action Order | Tag

Qualifier: N Non-Stored ☑ Boolean
Preset: 0 ms ☐ Use Expression Define...
Timer: 0 ms ○ Active
Count: 0 ○ Q
☑ Pause Timer When Routine is Paused
Indicator Tag: PG011_AP11.Motor_NR.Out_Run New Tag...

Figure 64 – Action Properties

8.4.3 SFC editor and SFC Step data types

The standard uses an SFC "UDT" for all tags used within an SFC routine and therefore "Auto-Naming" must be turned off, else it will create new tags every time a new step is added.



Workstation Options

Categories:
- Application
  - Display
  - Font/Color
- Tag Editor Display
- Ladder Editor
  - Display
  - Font/Color
- SFC Editor
  - Element Naming
  - Display
  - Font/Color
- FBD Editor
  - Font/Color
- Structured Text Editor
  - Font/Color
- Trends

Change automatic SFC element naming

☐ Create Tag with Language Element
● Generate Tag Name
○ Prompt for Tag Name

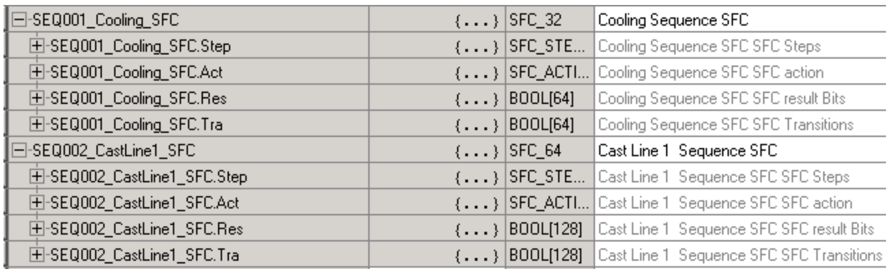
	Base Name:	Prefix with Routine Name:
Step:	Step	☐ Step_000
Transition:	Tran	☐ Tran_000
Action:	Action	☐ Action_000
Stop:	Stop	☐ Stop_000

☑ Delete Tag with Language Element

Reset Category | OK | Cancel | Apply | Help

Figure 65 – Automatic SFC element naming shall be turned off

For step data, a special SFC data structure called SFC_32 for sequences with fewer than 32 steps and SFC_64 for sequences fewer than 64 steps shall be used. The data structure is based on a simple UDT that can be copied and modified if more than 64 steps are needed. The tag instance shall take the same name as the SFC to be programmed, as shown in **Error! Reference source not found.**.

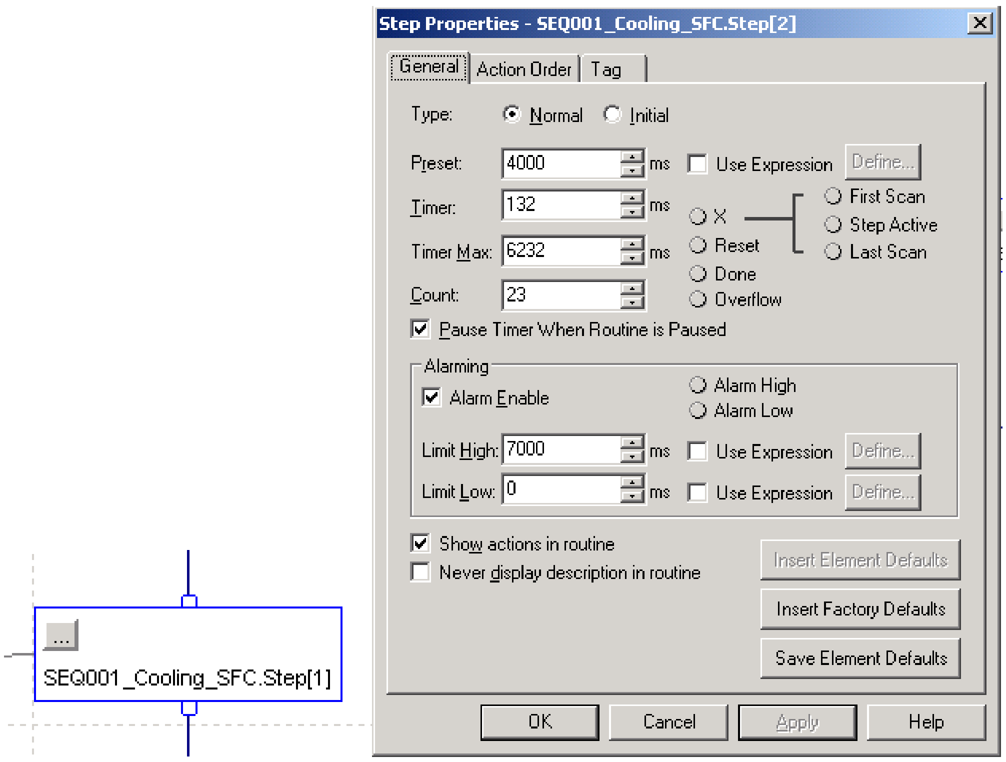


⊟-SEQ001_Cooling_SFC	{...}	SFC_32	Cooling Sequence SFC
⊞-SEQ001_Cooling_SFC.Step	{...}	SFC_STE...	Cooling Sequence SFC SFC Steps
⊞-SEQ001_Cooling_SFC.Act	{...}	SFC_ACTI...	Cooling Sequence SFC SFC action
⊞-SEQ001_Cooling_SFC.Res	{...}	BOOL[64]	Cooling Sequence SFC SFC result Bits
⊞-SEQ001_Cooling_SFC.Tra	{...}	BOOL[64]	Cooling Sequence SFC SFC Transitions
⊟-SEQ002_CastLine1_SFC	{...}	SFC_64	Cast Line 1 Sequence SFC
⊞-SEQ002_CastLine1_SFC.Step	{...}	SFC_STE...	Cast Line 1 Sequence SFC SFC Steps
⊞-SEQ002_CastLine1_SFC.Act	{...}	SFC_ACTI...	Cast Line 1 Sequence SFC SFC action
⊞-SEQ002_CastLine1_SFC.Res	{...}	BOOL[128]	Cast Line 1 Sequence SFC SFC result Bits
⊞-SEQ002_CastLine1_SFC.Tra	{...}	BOOL[128]	Cast Line 1 Sequence SFC SFC Transitions

Figure 66 – The SFC UDT structure and naming convention.

A typical step configuration is shown in **Error! Reference source not found.**.

NOTE: The use of the embedded timers is encouraged and if used, checking the "Pause timer when Routine is pause" is required. This will lower the number of established timers in the overall system.



Step Properties - SEQ001_Cooling_SFC.Step[2]

General | Action Order | Tag

Type: [✓ Normal] Initial

Preset: 4000 ms ☐ Use Expression Define...

Timer: 132 ms

Timer Max: 6232 ms

Count: 23

○ X — ○ First Scan, ○ Step Active, ○ Last Scan
○ Reset
○ Done
○ Overflow

☑ Pause Timer When Routine is Paused

Alarming
☑ Alarm Enable
○ Alarm High
○ Alarm Low
Limit High: 7000 ms ☐ Use Expression Define...
Limit Low: 0 ms ☐ Use Expression Define...

☑ Show actions in routine
☐ Never display description in routine

Insert Element Defaults | Insert Factory Defaults | Save Element Defaults

OK | Cancel | Apply | Help

Figure 67 – SFC step data and configuration

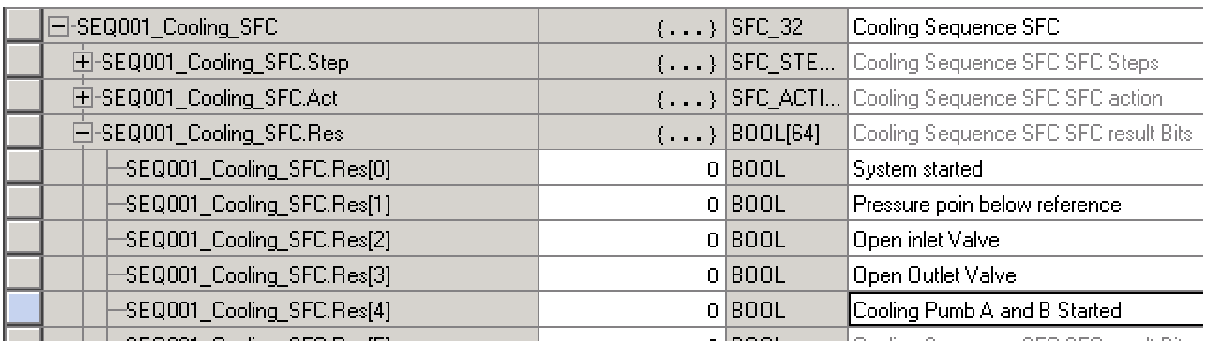
IMPORTANT: Ticking the “Pause Timer When Routine is Paused” is recommended to avoid timer related problems during “Hold”.

IMPORTANT: Description for all parameters and tags as well as rung description is mandatory. All actions, transitions and results shall include an appropriate description in the SFC UDT.

8.4.4 Transition Triggers

Transitions trigger shall be placed in the _LAD routine after the Action section. All logic condition to set the trigger bit of each transition in a sequence is programmed in the ladder.

Using only the Result bit from the SFC UDT makes the code more flexible as restarts of SFC are avoided during online changes. Furthermore, Ladder is the preferred choice for logical evaluations. Status is monitored by cross-referencing the .Res bit and viewing the ladder code. An example is shown in **Error! Reference source not found.** to **Error! Reference source not found.**.



⊟-SEQ001_Cooling_SFC	{...}	SFC_32	Cooling Sequence SFC
⊞-SEQ001_Cooling_SFC.Step	{...}	SFC_STE...	Cooling Sequence SFC SFC Steps
⊞-SEQ001_Cooling_SFC.Act	{...}	SFC_ACTI...	Cooling Sequence SFC SFC action
⊟-SEQ001_Cooling_SFC.Res	{...}	BOOL[64]	Cooling Sequence SFC SFC result Bits
SEQ001_Cooling_SFC.Res[0]	0	BOOL	System started
SEQ001_Cooling_SFC.Res[1]	0	BOOL	Pressure poin below reference
SEQ001_Cooling_SFC.Res[2]	0	BOOL	Open inlet Valve
SEQ001_Cooling_SFC.Res[3]	0	BOOL	Open Outlet Valve
SEQ001_Cooling_SFC.Res[4]	0	BOOL	Cooling Pumb A and B Started

Figure 68 – Result Bits in SFC UDT

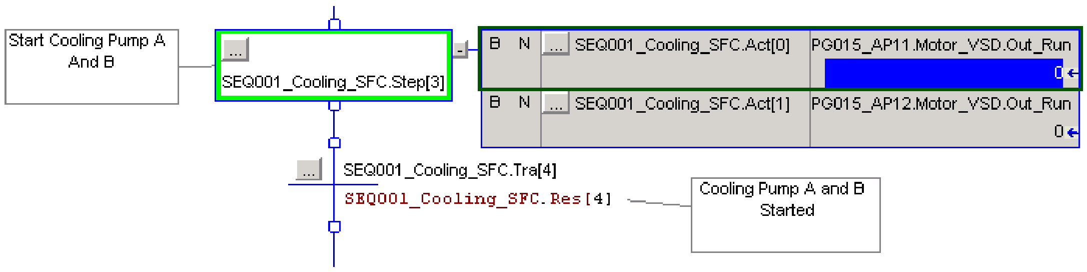

Start Cooling Pump A And B
SEQ001_Cooling_SFC.Step[3]
B N SEQ001_Cooling_SFC.Act[0] PG015_AP11.Motor_VSD.Out_Run
B N SEQ001_Cooling_SFC.Act[1] PG015_AP12.Motor_VSD.Out_Run
SEQ001_Cooling_SFC.Tra[4]
SEQ001_Cooling_SFC.Res[4]
Cooling Pump A and B Started

Figure 69 – Result bit used in an SFC chart

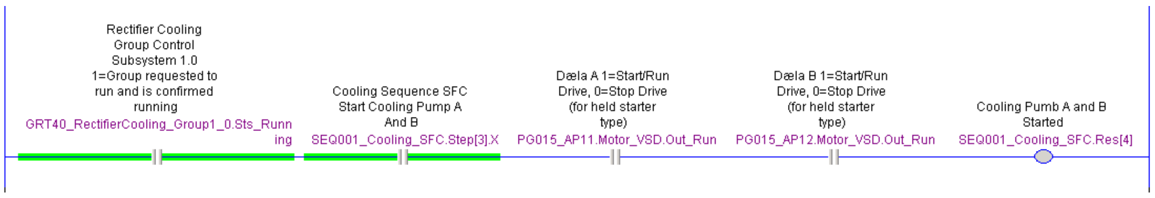

Rectifier Cooling Group Control Subsystem 1.0 1=Group requested to run and is confirmed running
GRT40_RectifierCooling_Group1_0.Sts_Running
Cooling Sequence SFC Start Cooling Pump A And B
SEQ001_Cooling_SFC.Step[3].X
Dæla A 1=Start/Run Drive, 0=Stop Drive (for held starter type)
PG015_AP11.Motor_VSD.Out_Run
Dæla B 1=Start/Run Drive, 0=Stop Drive (for held starter type)
PG015_AP12.Motor_VSD.Out_Run
Cooling Pumb A and B Started
SEQ001_Cooling_SFC.Res[4]

Figure 70 – Result Bit used in the _LAD routine

When the sequence is programmed using a Selective branch, all evaluations must be grouped together within the same rung. If the rung evaluation is very large, then rungs must be in consecutive rungs next to each other in the _LAD routine. This makes the evaluations more readable. An example is shown on **Error! Reference source not found.** to **Error! Reference source not found.**.

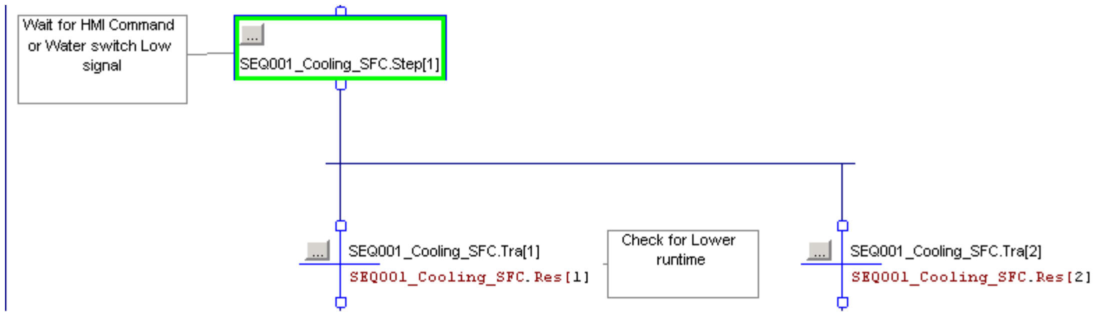

Wait for HMI Command or Water switch Low signal
SEQ001_Cooling_SFC.Step[1]
SEQ001_Cooling_SFC.Tra[1]
SEQ001_Cooling_SFC.Res[1]
Check for Lower runtime
SEQ001_Cooling_SFC.Tra[2]
SEQ001_Cooling_SFC.Res[2]

Figure 71 – Selective branch in an SFC chart

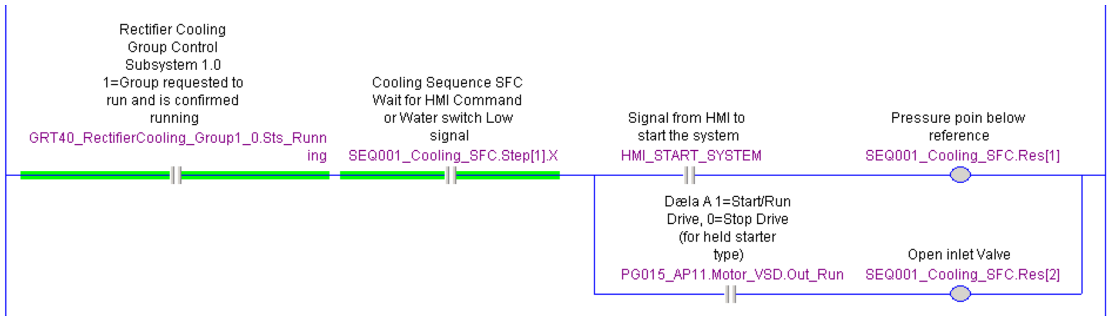

Rectifier Cooling Group Control Subsystem 1.0 1=Group requested to run and is confirmed running
GRT40_RectifierCooling_Group1_0.Sts_Running
Cooling Sequence SFC Wait for HMI Command or Water switch Low signal
SEQ001_Cooling_SFC.Step[1].X
Signal from HMI to start the system
HMI_START_SYSTEM
Pressure poin below reference
SEQ001_Cooling_SFC.Res[1]
Dæla A 1=Start/Run Drive, 0=Stop Drive (for held starter type)
PG015_AP11.Motor_VSD.Out_Run
Open inlet Valve
SEQ001_Cooling_SFC.Res[2]

Figure 72 – Selective Branch LAD routine

8.5 Testing and Simulation of CLX Programming

All software, inclusive of modules interfacing statuses, I/O routine's, control modules, equipment modules, procedural programs and routines shall be tested in a modular method.

The principal method of testing developed code, shall be utilizing RSLogix 5000 Emulator, for testing developed programs within the PC.

Following this, simulation of the code shall be done both within the PC using RSLogix Emulator 5000, or in a simulated lab environment utilizing Allen Bradley CLX hardware, I/O modules and appropriate networks and devices.

Simulation shall be carried out to ensure that all code development work is in line with operational parameters and executes as expected.

Full simulation shall be carried out that replaces the inputs with simulated values, these values shall be injected into the input assemblies as standard in simulation mode.

During FAT of the vendor application the fully simulated PLC code shall be used where the function of the code is tested in a safe environment.

Automated code for simulation purposes only, may be contained within a separate task that is specifically for simulation mode. When not in simulation mode this task shall not be in the task schedule to run.

Final versions of working code placed in the field shall not contain this simulation task; it shall be retained though as a specific version of the RSLogix software program on file issued to Norðurál.

8.5.1 Instructions for debugging application code

The use of instructions such as AFI is prohibited since it is impossible to maintain. If a vendor or a programmer needs to make changes to an application code or to block certain parts of a code temporarily the programmer shall create a DINT and use the XIO and XIC instructions. In the description field for the used elements of the DINT the programmer shall put his name and email address as well as the date and description of the modification. IO forces are also prohibited for use in any application in the Norðurál Site.

IMPORTANT: Unused branches are not allowed in the Norðurál Site.

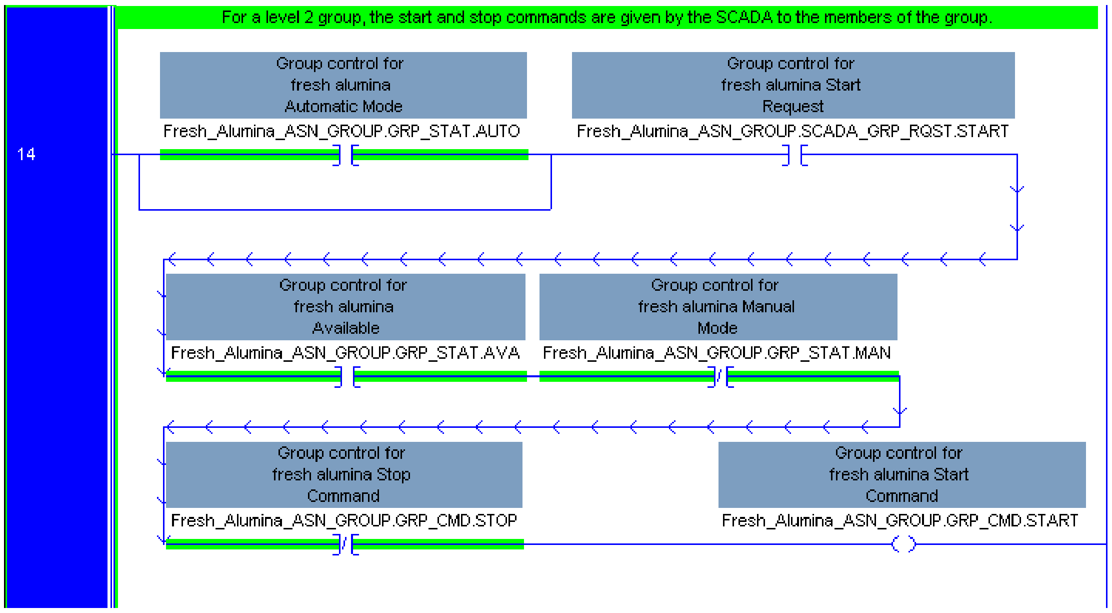

For a level 2 group, the start and stop commands are given by the SCADA to the members of the group.
14
Group control for fresh alumina Automatic Mode
Fresh_Alumina_ASN_GROUP.GRP_STAT.AUTO
Group control for fresh alumina Start Request
Fresh_Alumina_ASN_GROUP.SCADA_GRP_RQST.START
Group control for fresh alumina Available
Fresh_Alumina_ASN_GROUP.GRP_STAT.AVA
Group control for fresh alumina Manual Mode
Fresh_Alumina_ASN_GROUP.GRP_STAT.MAN
Group control for fresh alumina Stop Command
Fresh_Alumina_ASN_GROUP.GRP_CMD.STOP
Group control for fresh alumina Start Command
Fresh_Alumina_ASN_GROUP.GRP_CMD.START

Figure 73 – Unused Branch not allowed

8.6 Quality and Verification of Programming

At certain point during programming, verification of PLC, HMI and/or SCADA Vendor's programs shall be made to insure delivery of these programs to fulfill completely its quality level.

At 25% completion verification is mandatory. If program (at 25% completion) evaluation is not satisfactory, the vendor will be asked to modify its program according to comments and resubmit for approval. If resubmitted program evaluation is satisfactory, the vendor will be asked to submit its program at 50% completion for a second evaluation. If second evaluation is satisfactory, then the vendor will continue until 75% completion to be verified again as normal procedure.

When the programming has reached 100% a FAT shall take place. For the FAT, the vendors shall deliver all required test documents. In the FAT, the PLC application will be tested in a workshop environment to reduce commissioning time and hazards.

8.6.1 Program Evaluation 25%

- Software version verification
- General configuration
  - Controller naming
  - IO configuration shall be finished
  - Module naming
  - System overhead time slice
  - Node addresses for all DNB and CNB modules
  - Ethernet addresses for all ENB modules
- Program evaluation
  - Tasks structure shall be ready and period time for periodic tasks
  - Network setup shall be ready
- Comments and Descriptions
  - Descriptions shall be as per specifications
- Tags
  - Tag naming standards shall be followed
  - UDT and AOI structure as per project standards
- Mapping and Validation
  - Input mapping as per project standards shall be completed
  - I/O validation and alarm function as per project standards shall be completed
- Equipment Program
  - One Equipment Routine shall be ready for each type of equipment.
  - Main routine defined with logic “JSR instruction”
  - Fault routine (if used)
  - Modes routine containing logic (for SCADA indication)

8.6.2 Program Evaluation 50%

- Program evaluation
  - All routines and programs shall be ready
  - All Equipment routines shall be programmed.
- Comments and Descriptions
  - Descriptions shall be as per specifications
- Tags

    - All Tags shall be created

- Equipment Program
    - Equipment program shall be finished with all equipment connected
    - At least one sequence routine shall be completed.

8.6.3 Program Evaluation 75%

- General configuration
    - Available memory as per project standards, a minimum of 25% free space.
- Program evaluation
    - All sequences and Transition routines shall be completed.

8.6.4 Program Evaluation 100%

- FAT
    - All functions of the PLC system shall be simulated.

9 Communication and Interfaces

Network communication means transferring and validating information between devices like controller, HMI and SCADA. These mechanisms are listed below for data exchange, fault, alarm, and interlock.

9.1 Communication Link

Data exchange means data sent or received between two PLC's, or a PLC with external devices. There are two communication mechanisms used in the Norðurál Site, which are Produced / Consumed and Message:

Produced / Consumed	Message
Deterministic	Non-Deterministic
Scheduled portion of the bandwidth	Unscheduled portion of the bandwidth
Repetitive time interval (RPI)	No repetitive time interval
Highest priority	Low priority
Does not change medium	Can use different network types
Does not support online edit	Can be modified online

Table 19 – Communication links

NOTE: Time critical interlocks, control bits and other time sensitive data that have a direct impact on the behavior of the control system must be communicated through the method of Produced / Consumed tags with Ethernet communication.

Messages are less sensitive to settings in switches and routers and therefore often more convenient to use messages for general purpose data.

9.2 Produced and Consumed Tags

9.2.1 Communication

Where communication to other PLC's and devices are required, a special program shall be created for controller to controller communication for easy maintenance access.

Routine for communication with other PLCs shall be created under this program. One routine named "Com_Produced" will be used to map the data (interlock or control signals) produced for another PLC(s). Also, a routine named "Com_Consumed" will be used to validate the data (interlock or control signals) from another PLC(s).

One routine named "Message_Read" will be used to validate and map the data read into another PLC or explicit message reading. Also, a routine named "Message_Write" will be used to map the data to be written into another PLC or explicit message writing.

To share produced and/or consumed tags between controllers, controllers shall be attached to the same control network segment. The produced/consumed communication method is the first choice of communication to be used. Produced and consumed tags cannot be used over two distinct networks using a bridge. In that case, Message Instruction will be required.

Controllers can produce and consume tags over these networks:

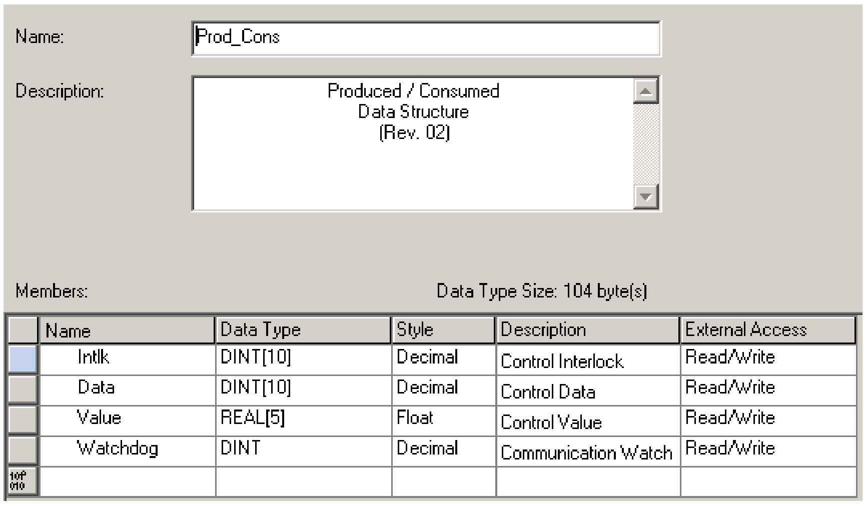
- ControlLogix chassis backplane (local rack)
- Ethernet/IP network

Produced tag: A tag configured as Produced in a controller can become available for another controller(s) (consumer). Multiple controllers can simultaneously consume this tag. When the tag is created, and configured, the produced tag is already sent (by the source controller) to the specified number of consumers.

Consumed tag: It is a tag configured in each controller who wants to receive (consume) the data of a produced tag. When a consumed tag is created, it must be the same data type and array dimension that the produced tag. Also, at the configuration of a consumed tag, you specify at which frequency the tag will be requested at the source controller. This parameter is the RPI (Requested Packet Interval). The RPI shall be at set minimum to network update time (NUT) or shall be a binary multiple of the NUT.

Principally because, produced and consumed tags cannot be configured online then they shall be transferred via UDT structured tag. These tags will be configured on all PLC's that need to receive the data, one as Produced tag in the source PLC and one as Consumed tag in the destination PLC's. This setup will permit exchange of signal(s) from the first PLC to the other PLC's.

The Produced / Consumed data type shall be as shown in **Error! Reference source not found.**;



Name: Prod_Cons

Description: Produced / Consumed
Data Structure
(Rev. 02)

Members: Data Type Size: 104 byte(s)

Name	Data Type	Style	Description	External Access
Intlk	DINT[10]	Decimal	Control Interlock	Read/Write
Data	DINT[10]	Decimal	Control Data	Read/Write
Value	REAL[5]	Float	Control Value	Read/Write
Watchdog	DINT	Decimal	Communication Watch	Read/Write

Figure 74 – Produced / Consumed data type

Also, there should be a timer running all the time from which the accumulator value is copied on the element (Watchdog) of the produced UDT structured tag. The accumulator mapping shall be done at last, after each data has been copied into the structure. In the PLC who consumed UDT structured tag, the timer value shall be recopied in a produced UDT structured tag to the PLC who has the timer to be compared.

These mechanisms will be used to detect if the connections associated to the UDT structured tags are actives or that the PLC is in run mode. These timers shall have duration of 10 seconds for the purposes of communication watchdog.

Each PLC, who consumed UDT structured tag, shall monitor the element (WATCHDOG) of the produced UDT structured tag value (timer accumulator) and detect if the value is changing to ensure communication link is being updated.

Finally, alias tag will be used to access elements of the Consumed UDT structured tag. The name of the alias tag shall match the tag of the element mapped in the Produced UDT structured tag. The communication watchdog bit shall always be used to validate data coming from the Consumed UDT structured tag when used in the logic.

IMPORTANT: When a Watchdog fails, it should be interpreted as an interlock fault. If the watchdog times out there shall be an alarm in the SCADA system.

9.2.2 Produced and Consumed Tags Naming and Contents

Naming rules for Produced and Consumed tag shall be as per the following format:

In the PLC producing a tag, the UDT structured tag name starts with "Produced_" followed by the PLC name.

- Produced_GRT80_FD100_CLX001

In the PLC consuming a tag, the UDT structured tag name starts with "Consumed_" followed by the PLC name producing the tag, to understand from which PLC the data is produced.

- Consumed_GRT80_FD100_CLX001

When a tag is consumed by another controller the tag names must match for both the producing controller and the consuming controller. The tag description of the alias tag in the consumer PLC shall be the same as the producer PLC tag description

The produced tag should be mapped to the appropriate UDT see example below.

GRT80_FD122_AN20_M10.Sts_Stopped: is mapped to the produced tag

- Produced_GRT80_FD100_CLX001.DATA[1].0

Consumed_GRT80_FD100_CLX001.DATA[1].0 is in turn mapped to a tag name that matches the tag name in the producing PLC

- _GRT80_FD122_AN20_M10_Sts_Stopped

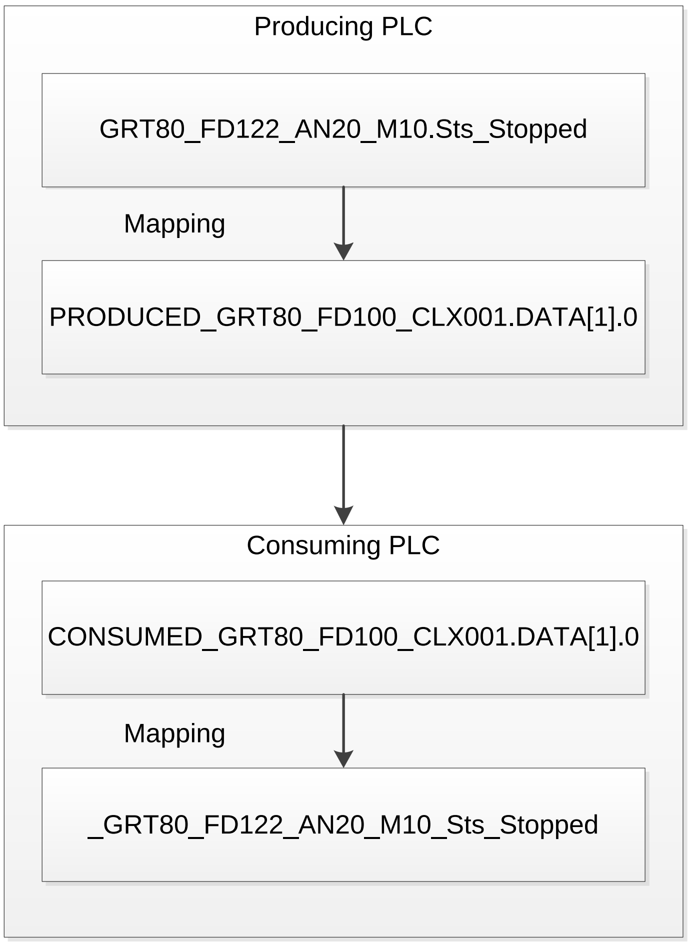

Producing PLC
GRT80_FD122_AN20_M10.Sts_Stopped
Mapping
PRODUCED_GRT80_FD100_CLX001.DATA[1].0
Consuming PLC
CONSUMED_GRT80_FD100_CLX001.DATA[1].0
Mapping
_GRT80_FD122_AN20_M10_Sts_Stopped

Figure 75 – Produced / Consumed tags

9.3 Message Instruction (MSG)

Message communication is any communication that you do not configure through the I/O configuration folder of the project, such as the MSG instructions.

Message communication occurs only when a periodic or event task is not running. If you use multiple tasks, make sure that their scan times and execution intervals leave enough time for message communication. Adjust the update rates of the tasks as needed to get the best trade-off between executing your logic and servicing message communication. The disadvantage is that extra care needs to be given to the scan time to avoid the tasks from overlapping and leaving time for other communication.

NOTE: Writing messages is considered more efficient that reading messages and therefore messages are to be sent using “CIP Data Table Write”.

The MSG instruction asynchronously reads or writes a PLC block of data (tag array or custom UDT that include different type of array like the produced-consumed UDT) to or from another

PLC block of data. The message communication method has a low priority and therefore should be watchdog monitored.

These tag arrays or custom UDT are configured in each controller who wants to read or write a tag. As with the produced/consumed tags, it must be the same data type and array dimension in both PLC's. Also, at the configuration of a read or write message instruction, you specify if a Cache Connection is needed when the frequency is important (under 1 sec.) If more than four continuous messages are to be used, cascade method shall be used instead of Cache Connection. Cascade message will be sent one at the time. This will use less amount of the unscheduled bandwidth.

Logic to ensure that the write message is conditional and time based cyclic is done in the communication routine. It is encouraged to program each message to trigger a write on different time intervals, even with prime numbers, to avoid collisions.

A watchdog and an alarm for communication failure are mandatory for message instructions like the produced and consumed connections.

9.3.1 Message Tags Naming and Contents

Naming rules for tag array sent or received shall be as per the following format:

A transmitting message tag shall begin with the letters "To_" followed by the destination PLC name followed by "_MSG1[xx]" for the first tag array. The [xx] represent the array dimension.

- To_GRT80_FD100_CLX001_MSG1[xx]

A receiving message tag shall begin with the letters "FROM_" followed by the source PLC name followed by "_MSG1[xx]" for the first tag array.

- From_GRT80_FD200_CLX002_MSG1[xx]

The same mapping shall be done for message tags as is used for the produced consumed tags see chapter **Error! Reference source not found.**.

The pilot project provides an UDT similar to produced/consumed for use with messages (Msg_data).

As for the produced/consumed tag, the description of the alias tag in the destination PLC shall be the same as the PLC source tag description. The tag description of the element of the tag array shall also follow the same format as a produced/consumed tag. The tag array shall have enough extra array elements declared for future use.

9.4 Alarm Handling

Alarms shall be created as standard in the PLC for all device objects and control objects, this shall be done in the device control module (routines) and equipment control module (for group).

Process-based alarms shall be created in the PLC at the procedural and equipment level as standard. Network, communication and module status events and alarms shall be created directly by the fault routines.

The PlantPAx library includes an alarm AOI (P_Alarm) and its use is highly encouraged.

Alarms shall be classified with the following table.

State	Severity	Range	Use
Low	1	0-250	Message
Medium	2	251-500	Warning
high	3	501-750	Alarm
Urgent	4	751-1000	Control system faults Network and PLC

Table 20 – Alarm Severity's

All alarms shall have an acknowledge bit associated as standard, for use with SCADA and HMI systems.

When an alarm occurs, depending on the priority, the equipment or group stops. To rearm, after that equipment or group fault or alarm warning has been removed, a re-start shall be done for the equipment or group.

9.5 Alarm List

When the vendor supplies only the PLC program, the vendor must then provide the Alarm List to Norðurál at specific milestones.

This list must include the following information;

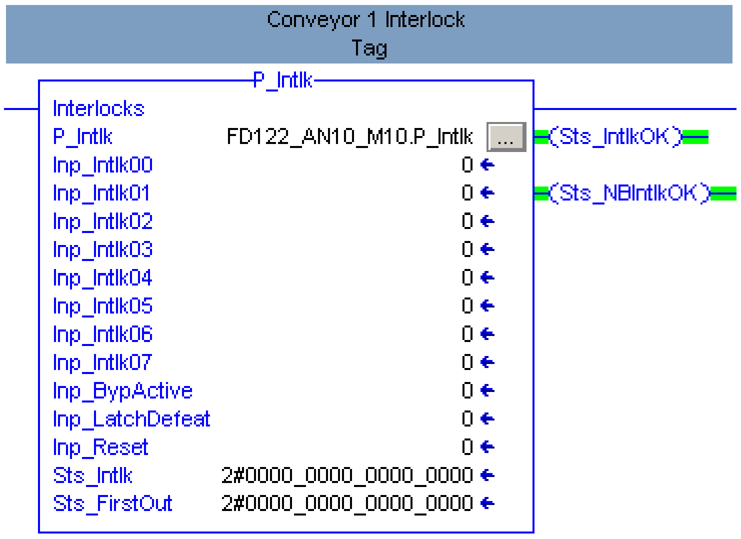
Item	Description
Tag Name	The tag name of the alarm specified in detail
Message Severity	As it is possible to change this in the Runtime SCADA system this should be stated as a default value for the severity, proposed by the vendor.
Message	The message for the user that will be displayed in the SCADA system
Alarm Class	The alarm class for the alarm, this is depending on the process the equipment is a part of. This is used for alarm filtering with in the SCADA system.
Disable Tag	Disable the Alarm
Suppress Tag	Suppress the Alarm
Acknowledge Tag	Acknowledge a single alarm
Global Acknowledge	Acknowledge all alarms

Table 21 – Alarm list

9.6 Interlocks

In the PlantPAx standard which the Norðurál standard is based on the Interlock is categorized as an occurrence in the process that results in the equipment stopping e.g. an overload fault or stall fault or connecting equipment fails while running.

The Interlock AOI can handle up to sixteen (16) interlocks. The Interlock AOI will handle the Interlocks and the result of the interlock validation will be an input to the equipment block.



```
Conveyor 1 Interlock
Tag
P_Intlk
Interlocks
P_Intlk          FD122_AN10_M10.P_Intlk   (Sts_IntlkOK)
Inp_Intlk00                          0    (Sts_NBIntlkOK)
Inp_Intlk01                          0
Inp_Intlk02                          0
Inp_Intlk03                          0
Inp_Intlk04                          0
Inp_Intlk05                          0
Inp_Intlk06                          0
Inp_Intlk07                          0
Inp_BypActive                        0
Inp_LatchDefeat                      0
Inp_Reset                            0
Sts_Intlk       2#0000_0000_0000_0000
Sts_FirstOut    2#0000_0000_0000_0000
```

Figure 76 – Interlock AOI

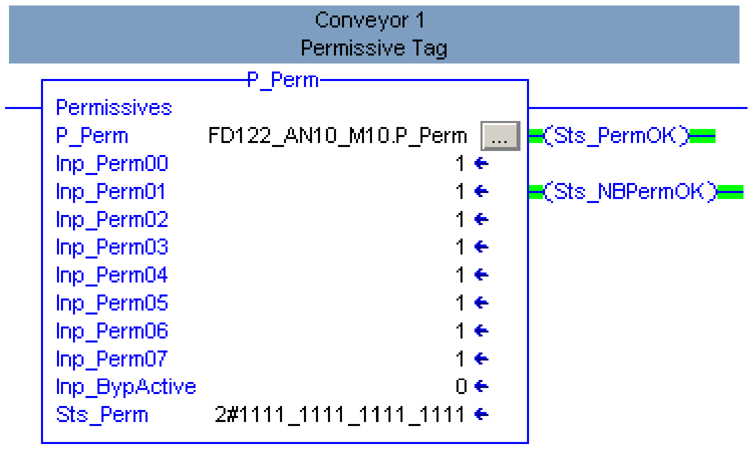
Process interlocks are for equipment operation and some can be modified for a specific process. Interlocks do not require acknowledgement unless they are connected to an alarm. Interlock alarms are visible from the popup for the equipment and from the alarm summary page. Acknowledgement for specific equipment interlock alarms can only be done from the alarm summary.

All interlocks immediately stop equipment in all modes of operation.

NOTE: Interlocks are not to be used for process stops. As a process stop is a part of normal automation function and interlocks will generate alarms.

9.7 Permissive

Permissive is categorized as permission criteria for the equipment to operate. All permissive conditions must be fulfilled for the equipment to start. A permissive state does not require acknowledgment.



```
Conveyor 1
Permissive Tag

P_Perm
Permissives
P_Perm          FD122_AN10_M10.P_Perm  ...   (Sts_PermOK)
Inp_Perm00                          1
Inp_Perm01                          1        (Sts_NBPermOK)
Inp_Perm02                          1
Inp_Perm03                          1
Inp_Perm04                          1
Inp_Perm05                          1
Inp_Perm06                          1
Inp_Perm07                          1
Inp_BypActive                       0
Sts_Perm         2#1111_1111_1111_1111
```

Figure 77 – Permissive AOI

The permissive conditions are configurable in the same way as the interlock conditions. Permissive only take effect when the equipment is in off mode or in stopped state.

10 Appendices

10.1 B – PlantPAx library

Standard Number	Description
SYSLIB-RM001	Basic Analog Input (P_AIn)
SYSLIB-RM002	Standard Alarm Sub-Block (P_Alarm)
SYSLIB-RM003	Discrete Input (P_DIn)
SYSLIB-RM004	Interlocks with First-Out and Bypass (P_Intlk)
SYSLIB-RM005	Standard Modes (P_Mode)
SYSLIB-RM006	Single-Speed Motor (P_Motor)
SYSLIB-RM007	Permissives with Bypass (P_Perm)
SYSLIB-RM008	Shared Reset (P_Reset)
SYSLIB-RM009	Restart Inhibit for Large Motor (P_ResInh)
SYSLIB-RM010	RunTime and Starts (P_RunTime)
SYSLIB-RM011	Analog Output (P_AOut)
SYSLIB-RM012	Two-Speed Motor (P_Motor2Spd)
SYSLIB-RM013	Reversing Motor (P_MotorRev)
SYSLIB-RM014	Motor Operated Valve (P_ValveMO)
SYSLIB-RM015	Solenoid Valve (P_ValveSO)
SYSLIB-RM016	Variable Speed Drive (P_VSD)
SYSLIB-RM018	Advanced Analog Input (P_AInAdv)
SYSLIB-RM019	Dual Analog Input (P_AInDual)
SYSLIB-RM020	Flow meter Dosing (P_DoseFM)
SYSLIB-RM021	Weigh Scale Dosing (P_DoseWS)
SYSLIB-RM022	Hand-Operated Motor Monitor (P_MotorHO)
SYSLIB-RM025	Hand-Operated 2-Position Valve (P_ValveHO)
SYSLIB-RM026	Multible Analog Outputs (P_AinMulti)
SYSLIB-RM027	Boolean Logic with Snapshot (P_Logic)
SYSLIB-RM028	Discrete 2-,3-,or 4-state Device (P_D4SD)
SYSLIB-RM029	Discrete Output (P_DOut)
SYSLIB-RM030	Analog Fanout (P_Fanout)
SYSLIB-RM031	n-Position Device (P_nPos)
SYSLIB-RM032	Pressure/Temp. Compensated Flow (P_PTComp)
SYSLIB-RM033	Tank Strapping Table (P_StrapTbl)
SYSLIB-RM034	Analog Control Valve (P_ValveC)
SYSLIB-RM035	Mix Proof Valve (P_ValveMP)
SYSLIB-RM036	2-State Valve Statistics (P_ValveStats)
	PowerFlex 755 module (P_PF755)
MMS_047415	GuardLogix Project Files
MMS_047416	ME Faceplates for GuardLogix Safety Systems
MMS_055487	Device Level Ring
MMS_057881	Stratix 8000
Logix Diagnostics RM003	PlantPAx library of Logix Diagnostics Objects
(RA-BAS) L_TaskMon-Faceplate	L faceplates for Logix controllers task monitor
(RA-BAS) L_CPU-Faceplate	L faceplates for Logix controllers CPU details

Table 22 – PlantPAx library

10.2 C – Abbreviations

Abbreviations	Definitions
1756	ControlLogix Control System
1794	Flex I/O distributed
1768	CompactLogix Control System
CAN	ControlNet Adapter Module
ACNR	Adapter ControlNet Redundant
ACS	ABB ACS (VFD)
AND	DeviceNet Adapter Module
ADR	Automatic Device Replacement
AC	Area Code
AK	AKS Code
AI	Analogue Input Module
AO	Analogue Output Module
AOI	Add-On Instruction
API	Actual Packet Interval
CIP	Control and Information Protocol, Common Industrial Protocol
CLX	ControlLogix controller
CNB(R)	ControlNet Bridge Module (Redundant)
COS	Change Of State
CPL	CompactLogix CPU
DI	Digital Input Module
DNB	DeviceNet Scanner Bridge Module
DO	Digital Output Module
DTC	Device Type Code
ENBT	Ethernet Bridge Module
EN2T	Ethernet Bridge Module
EQT	Equipment Tag
FLEX	Flex I/O Module of distributed I/O point, networked
FT	Factory Talk
FTD	Factory Talk Directory
GLX	GuardLogix Safety Controller
GPS	Global Position System
HI	Hart Input Module
HMI	Human Machine Interface, PanelView Plus, solid state
HO	Hart Output Module
HSC	High Speed Counter module
I/O	Input Output Device or point
ICE	Inview CE
ID	Net ID
IDI	Isolated Digital Input Module
IDO	Isolated Digital Output Module
MAC	Media Access Control address
MES	Manufacturing Execution Systems – Management
MVI	Serial Communication Module
NUT	Network Update Time
OSI	Open Systems International
P&ID	Process and Instrumentation Diagram
PID	Proportional Integral Derivative control
PM	Power Monitor
PVP	PanelView Plus (HMI)
RIUP	Removal and Insertion Under Power
RPI	Requested Packet Interval
RAN	Rack Name
CNR	ControlNet Remote Rack
ENR	EtherNet Remote Rack
LOR	Local Rack
SCADA	PC-based Supervisory Control And Data Acquisition System
SMAX	Scheduled Maximum (node No)
UDT	User-Defined Data Type

Abbreviations	Definitions
UMAX	Unscheduled Maximum (node no)
VFD	Variable Frequency Drive
IIS	Internet Information Service

Table 23 – Abbreviations