

EcoStruxure Control Expert Guidelines

Original instructions

02/2023

EIO0000004875.00

Legal Information

The Schneider Electric brand and any trademarks of Schneider Electric SE and its subsidiaries referred to in this guide are the property of Schneider Electric SE or its subsidiaries. All other brands may be trademarks of their respective owners.

This guide and its content are protected under applicable copyright laws and furnished for informational use only. No part of this guide may be reproduced or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), for any purpose, without the prior written permission of Schneider Electric.

Schneider Electric does not grant any right or license for commercial use of the guide or its content, except for a non-exclusive and personal license to consult it on an "as is" basis. Schneider Electric products and equipment should be installed, operated, serviced, and maintained only by qualified personnel.

As standards, specifications, and designs change from time to time, information contained in this guide may be subject to change without notice.

To the extent permitted by applicable law, no responsibility or liability is assumed by Schneider Electric and its subsidiaries for any errors or omissions in the informational content of this material or consequences arising out of or resulting from the use of the information contained herein.

As part of a group of responsible, inclusive companies, we are updating our communications that contain non-inclusive terminology. Until we complete this process, however, our content may still contain standardized industry terms that may be deemed inappropriate by our customers.

© 2023 – Schneider Electric. All rights reserved.

Table of Contents

Safety Information.....	5
Before You Begin.....	5
Start-up and Test.....	6
Operation and Adjustments	7
About the Book.....	8
Program Organization	12
Program Organization Concepts.....	12
Data Types and Variables.....	14
Function Blocks.....	16
Programming Languages	20
Programming Language Performance.....	20
Modules and Functional View	21
Physical Station Configuration	22
Project Management	26
Build a Project.....	26
Project Generation.....	27
Download the Project to the Controller	30
Online Modification	30
Improve Online Build Performance.....	31
Improve Offline Build Performance.....	32
Import Functions.....	33
Save and Restore the Latest Variable Values	34
Migration of Control Expert Projects	39
Duplicate a Function Block Instance.....	40
OFS and Control Expert Synchronization FB Instances	41
Modify the IP Address	42
Glossary	43
Index	47

Safety Information

Important Information

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, service, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.



The addition of this symbol to a “Danger” or “Warning” safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.



This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

DANGER

DANGER indicates a hazardous situation which, if not avoided, **will result in** death or serious injury.

WARNING

WARNING indicates a hazardous situation which, if not avoided, **could result in** death or serious injury.

CAUTION

CAUTION indicates a hazardous situation which, if not avoided, **could result in** minor or moderate injury.

NOTICE

NOTICE is used to address practices not related to physical injury.

Please Note

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and its installation, and has received safety training to recognize and avoid the hazards involved.

Before You Begin

Do not use this product on machinery lacking effective point-of-operation guarding. Lack of effective point-of-operation guarding on a machine can result in serious injury to the operator of that machine.

⚠ WARNING

UNGUARDED EQUIPMENT

- Do not use this software and related automation equipment on equipment which does not have point-of-operation protection.
- Do not reach into machinery during operation.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

This automation equipment and related software is used to control a variety of industrial processes. The type or model of automation equipment suitable for each application will vary depending on factors such as the control function required, degree of protection required, production methods, unusual conditions, government regulations, etc. In some applications, more than one processor may be required, as when backup redundancy is needed.

Only you, the user, machine builder or system integrator can be aware of all the conditions and factors present during setup, operation, and maintenance of the machine and, therefore, can determine the automation equipment and the related safeties and interlocks which can be properly used. When selecting automation and control equipment and related software for a particular application, you should refer to the applicable local and national standards and regulations. The National Safety Council's Accident Prevention Manual (nationally recognized in the United States of America) also provides much useful information.

In some applications, such as packaging machinery, additional operator protection such as point-of-operation guarding must be provided. This is necessary if the operator's hands and other parts of the body are free to enter the pinch points or other hazardous areas and serious injury can occur. Software products alone cannot protect an operator from injury. For this reason the software cannot be substituted for or take the place of point-of-operation protection.

Ensure that appropriate safeties and mechanical/electrical interlocks related to point-of-operation protection have been installed and are operational before placing the equipment into service. All interlocks and safeties related to point-of-operation protection must be coordinated with the related automation equipment and software programming.

NOTE: Coordination of safeties and mechanical/electrical interlocks for point-of-operation protection is outside the scope of the Function Block Library, System User Guide, or other implementation referenced in this documentation.

Start-up and Test

Before using electrical control and automation equipment for regular operation after installation, the system should be given a start-up test by qualified personnel to verify correct operation of the equipment. It is important that arrangements for such a check are made and that enough time is allowed to perform complete and satisfactory testing.

⚠ WARNING

EQUIPMENT OPERATION HAZARD

- Verify that all installation and set up procedures have been completed.
- Before operational tests are performed, remove all blocks or other temporary holding means used for shipment from all component devices.
- Remove tools, meters, and debris from equipment.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Follow all start-up tests recommended in the equipment documentation. Store all equipment documentation for future references.

Software testing must be done in both simulated and real environments.

Verify that the completed system is free from all short circuits and temporary grounds that are not installed according to local regulations (according to the National Electrical Code in the U.S.A, for instance). If high-potential voltage testing is necessary, follow recommendations in equipment documentation to prevent accidental equipment damage.

Before energizing equipment:

- Remove tools, meters, and debris from equipment.
- Close the equipment enclosure door.
- Remove all temporary grounds from incoming power lines.
- Perform all start-up tests recommended by the manufacturer.

Operation and Adjustments

The following precautions are from the NEMA Standards Publication ICS 7.1-1995:

(In case of divergence or contradiction between any translation and the English original, the original text in the English language will prevail.)

- Regardless of the care exercised in the design and manufacture of equipment or in the selection and ratings of components, there are hazards that can be encountered if such equipment is improperly operated.
- It is sometimes possible to misadjust the equipment and thus produce unsatisfactory or unsafe operation. Always use the manufacturer's instructions as a guide for functional adjustments. Personnel who have access to these adjustments should be familiar with the equipment manufacturer's instructions and the machinery used with the electrical equipment.
- Only those operational adjustments required by the operator should be accessible to the operator. Access to other controls should be restricted to prevent unauthorized changes in operating characteristics.

About the Book

Document Scope

This document provides you with best practices in the Control Expert environment. With this guide you can:

- Optimize the possibilities offered by the software. Refer to the description of program organization concepts, page 12.
 - file characteristics, page 39
 - library version, page 39
- Optimize the use of memory. Refer to these topics:
 - elementary data type, page 14.
 - addressing mode, page 18
 - application cleanup, page 19
- Optimize the performance of the PC in run time. Refer to these topics:
 - Program Units, page 13
 - DFB Block Size, page 17
 - Language Performance in RUN, page 20
 - Increase Performance, page 26
- Optimize the performance of the controller in build time. Refer to these topics:
 - Local Incremental Build, page 26
 - General Project Settings, page 31
 - General Considerations, page 33

Validity Note

This document is valid for all versions of Control Expert software and its precursor, Unity Pro.

The technical characteristics of the devices described in the present document also appear online. To access the information online, go to the Schneider Electric home page www.se.com/ww/en/download/.

The characteristics that are described in the present document should be the same as those characteristics that appear online. In line with our policy of constant improvement, we may revise content over time to improve clarity and accuracy. If you see a difference between the document and online information, use the online information as your reference.

Related Documents

Title of documentation	Reference number
<i>Modicon M580 Standalone System Planning Guide for Frequently Used Architectures</i>	HRB62666 (English), HRB65318 (French), HRB65319 (German), HRB65320 (Italian), HRB65321 (Spanish), HRB65322 (Chinese)
<i>Modicon M580 System Planning Guide for Complex Topologies</i>	NHA58892 (English), NHA58893 (French), NHA58894 (German), NHA58895 (Italian), NHA58896 (Spanish), NHA58897 (Chinese)
<i>Modicon M580 Hot Standby System Planning Guide for Frequently Used Architectures</i>	NHA58880 (English), NHA58881 (French), NHA58882 (German), NHA58883 (Italian), NHA58884 (Spanish), NHA58885 (Chinese)
Modicon M580, Hardware, Reference Manual	EIO0000001578 (English), EIO0000001579 (French), EIO0000001580 (German), EIO0000001582 (Italian), EIO0000001581 (Spanish), EIO0000001583 (Chinese)
Modicon M580, M340, and X80 I/O Platforms, Standards and Certifications	EIO0000002726 (English), EIO0000002727 (French), EIO0000002728 (German), EIO0000002730 (Italian), EIO0000002729 (Spanish), EIO0000002731 (Chinese)
Modicon M580, Change Configuration on the Fly, User Guide	EIO0000001590 (English), EIO0000001591 (French), EIO0000001592 (German), EIO0000001594 (Italian), EIO0000001593 (Spanish), EIO0000001595 (Chinese)
EcoStruxure™ Control Expert, Program Languages and Structure, Reference Manual	35006144 (English), 35006145 (French), 35006146 (German), 35013361 (Italian), 35006147 (Spanish), 35013362 (Chinese)
EcoStruxure™ Control Expert, Operating Modes	33003101 (English), 33003102 (French), 33003103 (German), 33003104 (Spanish), 33003696 (Italian), 33003697 (Chinese)
EcoStruxure™ Control Expert, Installation Manual	35014792 (English), 35014793 (French), 35014794 (German), 35014795 (Spanish), 35014796 (Italian), 35012191 (Chinese)

NOTE: Refer also to the online help for the maintenance expert tool (see EcoStruxure Automation Device Maintenance, Firmware Upgrade Tool, Online Help).

Product Related Information

⚠ WARNING
LOSS OF CONTROL - Perform a Failure Mode and Effects Analysis (FMEA), or equivalent risk analysis, of your application, and apply preventive and detective controls before implementation. - Provide a fallback state for undesired control events or sequences. - Provide separate or redundant control paths wherever required. - Supply appropriate parameters, particularly for limits. - Review the implications of transmission delays and take actions to mitigate them. - Review the implications of communication link interruptions and take actions to mitigate them. - Provide independent paths for control functions (for example, emergency stop, over-limit conditions, and error conditions) according to your risk assessment, and applicable codes and regulations. - Apply local accident prevention and safety regulations and guidelines.¹ - Test each implementation of a system for proper operation before placing it into service. Failure to follow these instructions can result in death, serious injury, or equipment damage.

¹ For additional information, refer to NEMA ICS 1.1 (latest edition), *Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control* and to NEMA ICS 7.1 (latest edition), *Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems* or their equivalent governing your particular location.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Only use software approved by Schneider Electric for use with this equipment.
- Update your application program every time you change the physical hardware configuration.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Terminology Derived from Standards

The technical terms, terminology, symbols and the corresponding descriptions in this manual, or that appear in or on the products themselves, are generally derived from the terms or definitions of international standards.

In the area of functional safety systems, drives and general automation, this may include, but is not limited to, terms such as *safety*, *safety function*, *safe state*, *fault*, *fault reset*, *malfunction*, *failure*, *error*, *error message*, *dangerous*, etc.

Among others, these standards include:

Standard	Description
IEC 61131-2:2007	Programmable controllers, part 2: Equipment requirements and tests.
ISO 13849-1:2015	Safety of machinery: Safety related parts of control systems. General principles for design.
EN 61496-1:2013	Safety of machinery: Electro-sensitive protective equipment. Part 1: General requirements and tests.
ISO 12100:2010	Safety of machinery - General principles for design - Risk assessment and risk reduction
EN 60204-1:2006	Safety of machinery - Electrical equipment of machines - Part 1: General requirements
ISO 14119:2013	Safety of machinery - Interlocking devices associated with guards - Principles for design and selection
ISO 13850:2015	Safety of machinery - Emergency stop - Principles for design
IEC 62061:2015	Safety of machinery - Functional safety of safety-related electrical, electronic, and electronic programmable control systems
IEC 61508-1:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: General requirements.
IEC 61508-2:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Requirements for electrical/electronic/programmable electronic safety-related systems.
IEC 61508-3:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Software requirements.
IEC 61784-3:2016	Industrial communication networks - Profiles - Part 3: Functional safety fieldbuses - General rules and profile definitions.
2006/42/EC	Machinery Directive
2014/30/EU	Electromagnetic Compatibility Directive
2014/35/EU	Low Voltage Directive

In addition, terms used in the present document may tangentially be used as they are derived from other standards such as:

Standard	Description
IEC 60034 series	Rotating electrical machines
IEC 61800 series	Adjustable speed electrical power drive systems
IEC 61158 series	Digital data communications for measurement and control – Fieldbus for use in industrial control systems

Finally, the term zone of operation may be used in conjunction with the description of specific hazards, and is defined as it is for a hazard zone or danger zone in the Machinery Directive (2006/42/EC) and ISO 12100:2010.

NOTE: The aforementioned standards may or may not apply to the specific products cited in the present documentation. For more information concerning the individual standards applicable to the products described herein, see the characteristics tables for those product references.

Program Organization

Program Organization Concepts

Introduction

Control Expert implements a modular approach by organizing programming code into these types of instructions:

- tasks (multitasking)
- sections
- subroutines (SRs)
- program units

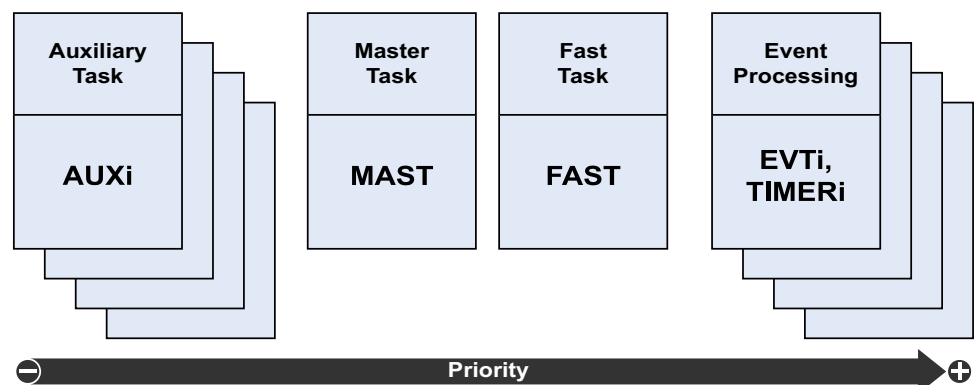
NOTE: Refer to the EcoStruxure™ Control Expert, Program Languages and Structure, Reference Manual.

Multitasking Optimizes Processing

Control Expert uses these tasks to distribute the code:

- *MAST*: The required MAST task contains the main part of the program. MAST tasks are periodic or cyclic.
- *FAST*: The periodic FAST task executes those parts of the application that require priority processing. This task is intended for use with short-cycle processes to avoid lower cycle tasks (MAST, AUX) to achieve the maximum (watchdog) cycle time.
- *Aux*: Auxiliary tasks execute parts of the application that have lower processing priorities. Periodic auxiliary tasks are available only if the MAST task is also periodic.
- *events*: Events execute the highest priority parts of the application when an external event from the I/O modules or programmable timer is triggered.

You can have up to four *Aux* tasks. This graphic represents a multitasking structure and the priority of tasks within the structure:



NOTE:

- The *i* suffix refers to the AUX, EVT, and TIMER task numbers.
- This multitasking system enables the optimization of controller performance and adapts the program to best meet the requirements of the process that is scheduled for automation.

Sections

You can program Control Expert sections in several IEC languages. Refer to the detailed description of those languages elsewhere in this guide, page 20.

Program Units

Program units facilitate the duplication and organization of a program with local and public variables. A program unit has these components:

- parameters
- local variables
- sets of sections

You can use program units to build a very modular application. In this case, online modifications are more effective because changes are localized within a small area of the application. Use programming units as much as possible in your program.

Subroutines

Subroutines (called SR) are used for recurring processing tasks that implement the same global application variables.

You can program a subroutine section in the IEC languages described. Like sections, each subroutine section can belong to only one task. As subroutines use only global variables to pass parameters, use Derived Functions Blocks which provide a clearer interface definition with dedicated parameters.

NOTE: Control Expert continues the subroutine mechanism for a migration from our legacy products.

Data Types and Variables

Introduction

Control Expert facilitates the use of these categories of variables:

- *located*: Located variables are defined by their physical memory locations.
- *unlocated*: Unlocated variables are defined by symbolic names. The use of unlocated variables allows you to capitalize on these advantages:
 - Increased standardization can be realized among controller families (Premium, Quantum, Atrium).
 - Because the system manages the variables, an independence from physical controller memory enhances program consistency and efficiency by reducing the number of overlapping memory spaces.
 - There is a potential for development in the absence of high-level coordination between development teams.
 - Program sections that implement unlocated variables can be reused to reassign variables using their names.
 - Maintenance is reduced and simplified. For example, the ability to access and modify tables (arrays) is enhanced.

Elementary Data Types

Control Expert uses these elementary data types (EDTs):

- Integers (INT, UINT, DINT, UDINT) or floating point real data types are used in arithmetic operations.
- Bit-string data types (WORD, DWORD, BYTE) are used in logical operations (AND, OR, XOR, etc.).
- Boolean data types are used for single bits (BOOL) or with edge management (EBOOL).
- Time management data types (Date, Time, TOD, DT).

In Control Expert you can use extracted bits of WORD (type INT, WORD, and BYTE) instead of boolean types. That allows you to work directly on a word instead of a bit (which you cannot do with BOOL types). It also saves memory; every 16 extracted WORD bits consume 2 memory bytes, while 16 BOOL types consume 16 memory bytes.

Derived Data Types

You can define derived data types (DDTs) as sets of other data types. DDTs generally allow the structure and legibility of an application to be optimized through the creation of DDT variables that correspond to application objects (motors, valves, etc.).

Observe the organization of data types:

- *structure*: A DDT structure contains different types of elements.
- *array*: An array contains elements of the same type.

The modification of a DDT type (like the addition or deletion of elements) is automatically propagated to all corresponding instances within the program, thereby reducing the project build time.

Use DDT structures as much as possible in your program code to assist in limiting the number of variables while optimizing the performance of the variable editor in Control Expert, thereby reducing the build time for the application.

Legible and efficient programs do not use large DDT structures that include multiple objects. More refined programs define a single structure for each object (motor, valve, etc.) to help enhance application legibility.

Function Blocks

Elementary Function Blocks

Elementary function blocks (EFBs) are written in the C programming language. EFBs have these components:

- input variables
- internal variables
- output variables
- processing algorithm (C language)

The optional Control Expert EFB Toolkit is required to implement EFBs.

The comprehensive set of instructions within the C language facilitates the execution of application code that requires highly complex algorithmic processing. To optimize the performance of run-time applications, the code that is generated by the EFB Toolkit is executed by the controllers.

EFBs facilitate the standardization of specific functions in the procedure to be automated or to standardize application objects (Motor, Valve, etc.) by offering this functionality:

- The generic processing of information exchanges is through parameters that can be transferred by reference (address) or directly by value.
- Parameters and internal variables can be latched during processing.

Run-time performance is particularly sensitive if the EFB uses iterative instructions (loops) or digital treatment.

NOTE: If you share EFBs, take measures to protect your intellectual property, including the inclusion of the copyright symbol or notice.

Derived Function Blocks

The user develops derived function blocks (DFBs) using one or more of these IEC languages (according to the number of sections):

- graphical languages (LD, FBD)
- textual languages (ST, IL)

DFBs enable the standardization of specific functions in the procedure to automate or standardize application devices (motors, valves, etc.). In a more general sense, DFBs offer these advantages:

- The design of the program and the entry of data are simplified.
- The legibility of the program is increased.
- Program debugging is facilitated.
- The volume of generated code is reduced.

DFB Performance Guidelines

- Limit the size of DFB sections to optimize performance during online modifications. In this case, the ST language offers better performance than the LD or FBD languages.
- Run-time performance is enhanced if the DFB uses BOOLEAN instructions, linear instructions (without loops), or instructions that act on tables (arrays).
NOTE: Optimize DFB code that is used repeatedly, especially if the code is reused hundreds of times.
- Performance during run time is particularly degraded with the use or overuse of these features:
 - loops (Do while, For Next, etc.)
 - indexed programming (array[index])
 - upstream jumps
 - large movement of data memory
 - several levels of nested DFBs

Some DFB instances are used several times in the same program. This can reduce the amount of consumed data memory, but observe the following rules to avoid conflicts within the application data.

Avoid these situations:

- The DFB contains private variables for which stored values are required to be maintained after the execution of the DFB instance.
- The DFB has Init values that are configured in public or private variables or in the input parameters.
- The DFB contains timers or counters.

There is no data conflict in these cases:

- The DFB contains inputs, outputs, and input/output parameters.
- The DFB contains private variables for which stored values are not necessarily maintained after the execution of the DFB instance.

Addressing Mode

You can also reduce data consumption by the DFB by using addressing mode passed by address.

The addressing mode of a function block element is linked to the element type.

Addressing modes are passed by:

- value (VAL)
- relocation table entry (RTE)
- logical address: RTE+Offset (L-ADR)
- logical address and number of elements (L-ADR-LG)
- I/O channel structure (IOCHS)

For each DFB parameter, use these object families with the associated addressing modes:

Object Families	EDT	STRING	Anony-mous or DDT array	DDT	IODDT	GDT: ANY_ARRAY_x	FB	ANY...
inputs	VAL(1)	L-ADR-LG	L-ADR-LG	L-ADR(2)	no	L-ADR-LG	no	L-ADR-LG
inputs/outputs	L-ADR	L-ADR-LG	L-ADR-LG	L-ADR(3)	IOCHS	L-ADR-LG	no	L-ADR-LG
outputs	VAL	VAL	L-ADR-LG	VAL(4)	no	L-ADR-LG	no	L-ADR-LG
public variables	VAL	VAL	VAL	VAL	no	no	no	no
private variables	VAL	VAL	VAL	VAL	no	no	RTE	no

Considerations:

- When possible, group EDT inputs in a DDT structure to reduce the scan time. The exchange of EDT data is performed by the value VAL(1) while DDT data is exchanged by address L-ADR(2). This reduces the number of input pins that are occupied by exchanges.
- When instances of large structures (DDTs with several hundred elements) are connected to a DFB's output parameter, addressing mode by VAL(4), you can optimize the execution time of the DFB by modifying the output parameter in the input/output parameter, addressing mode by address L-ADR(3).

Elementary Functions

Elementary function (EF) blocks are program functions that perform predefined software functions.

Control Expert provides a set of system EFs, but you can develop your own EFs in the C language using the optional EFB ToolKit, which integrates Microsoft's C compiler.

EFs do not have an internal latch of variables. Use this operating mode to help limit the consumption of memory in the data area.

NOTE: For applications that require more memory capacity in the data zone, use EFs rather than EFBs when the desired function is proposed for both EF and EFB types.

Application Cleanup

During programming, functions or variables that become useless can be removed from programming sections within the application. Even when these items are no longer present, they remain associated with the application. In that case, purge the unused FB, EFB, and DFB functions from the application to clear the corresponding memory by following the appropriate instruction sets below.

Purge Unused FB Instances:

Step	Action
1	Open the Data Editor (Tools > Data Editor).
2	Select the Function Blocks tab.
3	Right-click the row in the table and scroll to Purge Unused FB Instances .

Purge Unused DFB types:

Step	Action
1	Open the Data Editor (Tools > Data Editor).
2	Select the DFB Types tab.
3	Right-click a row in the table and scroll to Purge Unused Types .

Purge Unused Types:

Step	Action
1	Open the Types Library Browser (Tools > Types Library Browser).
2	Select the All Types tab.
3	Right-click a row in the table and scroll to Purge Unused Types .

The purge of unused types achieves these goals:

- Eliminate EF binary code from the program memory area when it is no longer called by sections or DFBs in the application.
- Eliminate EFB binary code from the program memory area when instances have already been deleted from the variables editor.
- Remove EF or EFB types that are no longer used in the Libset's *Application* directory.

Purge Variables:

Step	Action
1	Open the Data Editor (Tools > Data Editor).
2	Select the Variables tab.
3	Right-click a row in the table and scroll to Purge Unused Variables .

You can also clean up the DFB types for useless private variables:

Step	Action
1	Open the Data Editor (Tools > Data Editor).
2	Select the DFB Types tab.
3	Right-click a row in the table and scroll to Purge Unused Private Data Instances.

NOTE: It is good practice to purge these DFB types after you program the DFB to decrease the memory consumption of the corresponding instances.

Programming Languages

Programming Language Performance

Introduction

Control Expert includes these IEC61131-6 languages as standard:

Language	Description	Limit
IL (instruction list)	Because this language is very close to assembly language, you can work in very close conjunction with the controller by using its registers and accumulators.	300 lines
ST (structured text)	This structured language allows you to process algorithmic programs with complex digital processing.	1000 lines
LD (ladder diagram)	The ladder language is aligned with the logic in schematic electrical diagrams. It is particularly well adapted to programs written in combinational logic. However, the set of instructions enables more complex algorithms to be implemented in operation blocks through ST instructions.	1000 lines
FBD (functional block diagram)	This language is particularly suited to programs that include process control.	300 blocks
SFC (sequential functional chart)	This "state machine" type language adapts to sequential processing by steps and transitions.	—
LL984 (Ladder Logic 984)	LL984 networks are grouped in segments.	40 networks per segment

The *Limits* column shows the approximate size limitations that apply to each language. Whenever possible, use small sections to facilitate program maintenance and optimize performance during online modifications. The sizes of sections are limited by the application structure.

Considerations:

- The execution of a section is controlled by a conditional bit that is configured in the section properties. This feature facilitates the optimization of the scan time when the process is not required to execute at least one section in each controller cycle.
- You can protect the section with a password in Read or Write mode to help enhance confidentiality or security.
- A section can belong to only one task (MAST, FAST, Aux, etc.).

Language Performance in RUN

Applications that use Boolean instructions have these characteristics:

- The performance is identical for the ST, IL, and LD languages.
- FBD performance is reduced while data is copied among blocks.

Applications that use both Boolean and numerical instructions have these characteristics:

- The performance is identical for the ST and IL languages.
- LD performance is reduced 10 percent when compared to ST or IL.
- FBD performance is reduced 30 percent when compared to ST or IL.

Performance considerations:

- LD: If you use ladder logic to optimize run-time performance, uncheck **Generate with LD link animation** in Control Expert (**Tools > Project Settings > General > Build settings > Generate with LD link animation**).
- ST: Observe these guidelines in structured text:
 - To reduce consumption, use *While Do* loops instead of *For Next* loops whenever possible.
 - To optimize cycle times, use the Libset functions that are dedicated to array management. That is, use `MOVE_AREBOOL_INT` instead of assigning *For* loops whenever possible.
- Avoid or limit the assignment of variables with constant values. Instead, use the Init value that is defined in the **Data Editor** whenever possible.
- Limit or avoid the element-by-element exchange of information (`MyStruct1.Element1 := My_Struct2.Element1`, etc.). Whenever possible, use a direct address (`My_Struct := My_Struct2`) to perform the complete exchange of elements.
- Limit the use of large numbers of these functions:
 - `BIT_TO_WORD()` -> Use the assignment on extract bits.
 - `WORD_TO_BIT()` -> Use the assignment on extract bits.
 - `SEL()` -> Use the function `MOVE()`.

Use several smaller SFC sections (Master Chart/Slave Charts) rather than large SFC charts (sections).

Modules and Functional View

Introduction

A functional module is a group of program elements that are intended to perform a controller function. It includes these components:

- task- or event-processing sections and program unit
- animation tables associated with the functional module
- operator screens associated with the functional module
- nested functional modules that perform one or more automated sub-functions

Principle and Advantages

Adopt a development approach that uses modules and the functional view to achieve these goals:

- Bypass some limits of the structure that are imposed by the controller system.
- Develop logic in a consistent and reusable way that is applicable to all projects.
- Implement a direct-access strategy for automation functions.

NOTE: The representation of the application in the functional view does not account for the order in which the controller executes the program. The sections contained within functional modules are executed in the order in which they are organized in the tasks of the conventional structural view (from top to bottom).

Physical Station Configuration

Input and Output Modules

Moving an application-specific module within the configuration automatically reallocates all references to that module within the program. You can, therefore, modify the module configuration at any time without impacting the program.

If the same type of module is used several times in the configuration, configure one module and copy and paste its configuration to other locations. You can adjust the module configuration parameters that are copied for each module at any time.

DDTs for I/O Modules and DDTs for Other Devices

The derived data type for a specific I/O module (IODDT) encapsulates the variables that correspond to the module's application-specific channels in data structures. These structures are predefined and supplied in Control Expert for the various modules of the controller station.

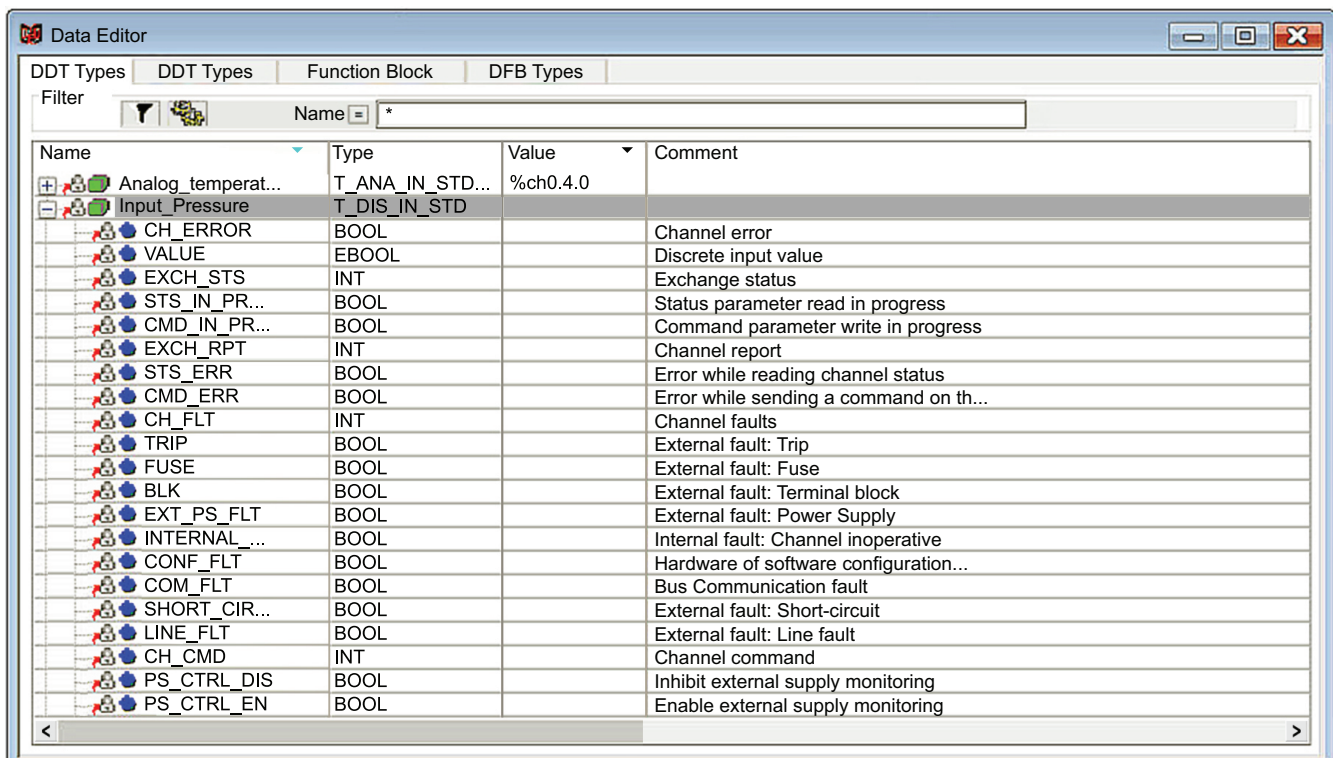
IODDT characteristics:

- The use of IODDTs facilitates access to application-specific channels through predefined elements that can be customized (Alias) and referred to by the program.
- With IODDTs you can program the application without having application-specific modules present in your hardware configuration. This is especially useful in the development of generic applications. Links to the modules are established after they are defined in the physical station configuration.

IODDTs are accessed in the library by selecting the **Catalog** section.

NOTE: IODDT instances are automatically allocated to physical addresses (like %MW).

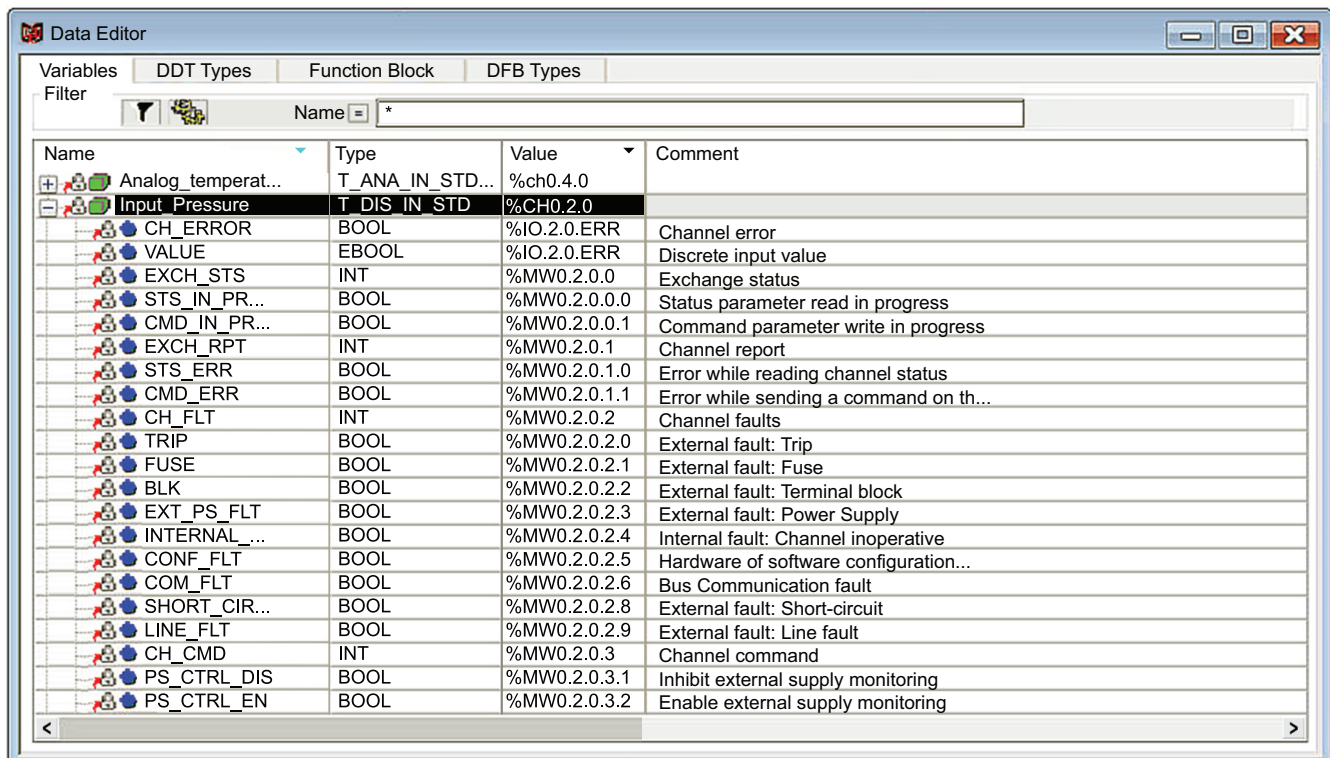
This is an example of an Input_Pressure instance of a standard T_DIS_IN_STD IODDT that is used with discrete input modules:



The screenshot shows the 'Data Editor' window with the 'DDT Types' tab selected. A filter is applied to show only the 'Input_Pressure' instance of the 'T_DIS_IN_STD' type. The table below represents the data shown in the window.

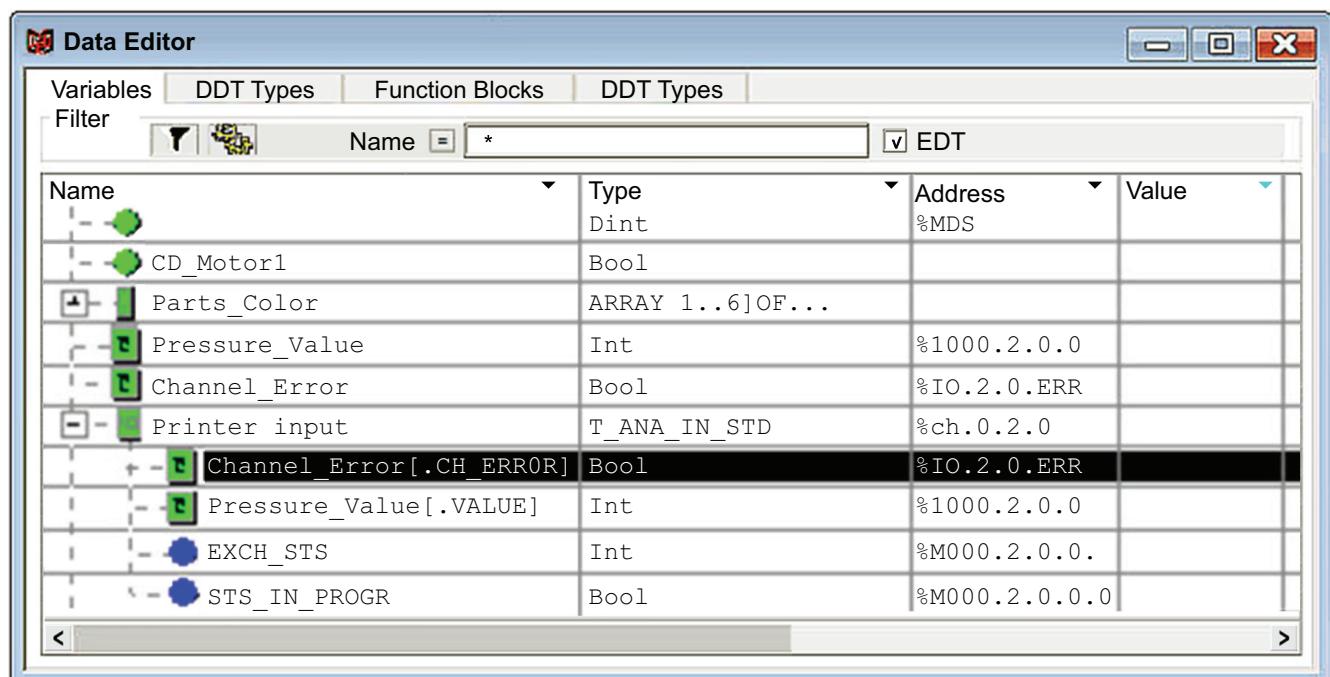
Name	Type	Value	Comment
+	Analog_temperat...	T_ANA_IN_STD...	%ch0.4.0
-	Input_Pressure	T_DIS_IN_STD	
CH_ERROR	BOOL		Channel error
VALUE	EBOOL		Discrete input value
EXCH_STS	INT		Exchange status
STS_IN_PR...	BOOL		Status parameter read in progress
CMD_IN_PR...	BOOL		Command parameter write in progress
EXCH_RPT	INT		Channel report
STS_ERR	BOOL		Error while reading channel status
CMD_ERR	BOOL		Error while sending a command on th...
CH_FLT	INT		Channel faults
TRIP	BOOL		External fault: Trip
FUSE	BOOL		External fault: Fuse
BLK	BOOL		External fault: Terminal block
EXT_PS_FLT	BOOL		External fault: Power Supply
INTERNAL_...	BOOL		Internal fault: Channel inoperative
CONF_FLT	BOOL		Hardware or software configuration...
COM_FLT	BOOL		Bus Communication fault
SHORT_CIR...	BOOL		External fault: Short-circuit
LINE_FLT	BOOL		External fault: Line fault
CH_CMD	INT		Channel command
PS_CTRL_DIS	BOOL		Inhibit external supply monitoring
PS_CTRL_EN	BOOL		Enable external supply monitoring

This sample **Variable** tab shows the link between an instance and a discrete input module located in slot 2 of backplane 0. You can see that Control Expert automatically assigns physical addresses for each element in the data structure:



Name	Type	Value	Comment
Analog_temperat...	T_ANA_IN_STD...	%ch0.4.0	
Input_Pressure	T_DIS_IN_STD	%CH0.2.0	
CH_ERROR	BOOL	%IO.2.0.ERR	Channel error
VALUE	EBOOL	%IO.2.0.ERR	Discrete input value
EXCH_STS	INT	%MW0.2.0.0	Exchange status
STS_IN_PR...	BOOL	%MW0.2.0.0.0	Status parameter read in progress
CMD_IN_PR...	BOOL	%MW0.2.0.0.1	Command parameter write in progress
EXCH_RPT	INT	%MW0.2.0.1	Channel report
STS_ERR	BOOL	%MW0.2.0.1.0	Error while reading channel status
CMD_ERR	BOOL	%MW0.2.0.1.1	Error while sending a command on th...
CH_FLT	INT	%MW0.2.0.2	Channel faults
TRIP	BOOL	%MW0.2.0.2.0	External fault: Trip
FUSE	BOOL	%MW0.2.0.2.1	External fault: Fuse
BLK	BOOL	%MW0.2.0.2.2	External fault: Terminal block
EXT_PS_FLT	BOOL	%MW0.2.0.2.3	External fault: Power Supply
INTERNAL_...	BOOL	%MW0.2.0.2.4	Internal fault: Channel inoperative
CONF_FLT	BOOL	%MW0.2.0.2.5	Hardware of software configuration...
COM_FLT	BOOL	%MW0.2.0.2.6	Bus Communication fault
SHORT_CIR...	BOOL	%MW0.2.0.2.8	External fault: Short-circuit
LINE_FLT	BOOL	%MW0.2.0.2.9	External fault: Line fault
CH_CMD	INT	%MW0.2.0.3	Channel command
PS_CTRL_DIS	BOOL	%MW0.2.0.3.1	Inhibit external supply monitoring
PS_CTRL_EN	BOOL	%MW0.2.0.3.2	Enable external supply monitoring

Two Aliases (symbols) have been defined by the user (Erreur_Voie and Value_Pression) to correspond to two elements in this IODDT structure:



Name	Type	Address	Value
	Dint	%MDS	
CD_Motor1	Bool		
Parts_Color	ARRAY 1..6]OF...		
Pressure_Value	Int	%1000.2.0.0	
Channel_Error	Bool	%IO.2.0.ERR	
Printer input	T_ANA_IN_STD	%ch.0.2.0	
Channel_Error[.CH_ERROR]	Bool	%IO.2.0.ERR	
Pressure_Value[.VALUE]	Int	%1000.2.0.0	
EXCH_STS	Int	%M000.2.0.0.	
STS_IN_PROGR	Bool	%M000.2.0.0.0	

Communication Configuration for M340 and Legacy Platforms (Premium, Quantum)

You can define the logical configuration of the networks (Ethernet, Modbus Plus, Fipway, etc.) to which the controller station is connected.

Define the communication directory for a particular network in the **Project Browser (Project Browser > Communication > Networks)**.

This is the configuration process for the logical network:

Step	Action
1	Create the network.
2	Configure the network.
3	Declare a module or the communication module.
4	Associate the communication module with the network.

You do not have to complete all of the above steps to start programming. You can start programming as soon as you create and configure the logical network. The declaration and association actions for the communication module can be done at any time after the physical locations of the network modules have been determined in the station configuration.

To confirm the new network and see its name, expand **(+) Networks** in the project structure (**Project Browser > Communication > Networks**).

FDT/DTM

Field Device Tool/Device Type Manager (FDT/DTM) technology was introduced to assist in the simplification of the integration of field devices in distributed control systems (DCSs). It has these advantages:

- An open communication interface integrates equipment from different third-party manufacturers through the use of descriptive files called DTMs to configure, monitor, and manage the lifecycle of such equipment.
- A standard user interface operates the equipment (engineering, operations, maintenance).
- FDT/DTM is platform-independent, so each device is accessed in the same way through its required corresponding DTM. This is true for configurations and lifecycle management systems.
- Because FDT/DTM is protocol-independent, FDT containers and DTM devices can have communications interfaces for HART, Profibus, Foundation Fieldbus, AS-I, and other industrial protocols.

For example, the sample architecture described below is managed by FDT/DTM. Communications are handled by EtherNet/IP for the Advantys STB Distributed I/O and Modbus TCP/Profibus DP for the Profibus DP Remote Master (PRM) gateway:

For an architecture that is managed by FDT/DTM, communications for an STB island could be handled by EtherNet/IP, and communications for a PRM gateway could be handled by Modbus TCP/Profibus DP.

After you install the DTM, update the **DTM catalog** in Control Expert:

Step	Action
1	Open the Hardware Catalog (Tools > Hardware Catalog).
2	Select the DTM catalog tab.
3	Click the Update button.
4	Follow the prompts to update the catalog.
5	Confirm that the DTM for the module appears in the DTM Browser for the project (Tools > DTM Browser).

The IO scanning table is automatically populated for devices that communicate through Profibus DP/PA to the PRM gateway and are connected to Modbus TCP.

Device Name		Unit ID	Slave Syntax	Health Timeout (ms)	Repetitive rate (ms)	RD Master Object	RD Ref Slave	RD length	Last value (Input)	WR Master Object	WR Ref Slave	WR length
My_PRM_Master	...	255	Index	1500	280	%MW0	0	50	Set to 0	%MW100	4096	50
	...					%MW0	0	28		%MW100	4096	4
	...					%MW28	2	22		%MW104	4100	46

In this case, the device variables that are managed by DTMs are automatically generated in Control Expert, as confirmed on the **Variables** tab (**Tools > Project Browser > Variables & FB instances > Device DDT Variables**).

Project Management

Build a Project

When to Rebuild

Rebuild the entire Control Expert project (**Build > Rebuild All Project**) in these situations:

- The project is built for the first time.
- You switch from a controller to a simulator.
- The controller or communication module was replaced.
- Certain modifications were made to the **Project Settings** (example: **Tools > Project Settings > General > PLC embedded data > Upload information**).

Optimize Generation Times

To optimize generation times, use a local incremental build (**Build Changes**). This is required in connected mode controller in RUN (On line modification) or controller in STOP.

Increase Performance

Observe these best practices to help enhance performance:

- After commissioning, the application is debugged and defined. Rebuild the entire project at this time (**Build > Rebuild All Project**) to maximize the available memory.
- When Control Expert detects excessive memory fragmentation, the **Mem** indicator in the bottom status bar appears. To maximize the memory consumption, go to **PLC > Memory Consumption** and click the **Pack** button.
- Select **Virtual connected mode** in the **Property Settings** dialog box (**Tools > Project Settings > General > Build Settings**) to modify the project locally so that it can be transferred later without stopping the controller. In this mode, the **Build Changes** and **Rebuild All Projects** commands are disabled to avoid the modification of the project signature before there is a connection to the controller.

Project Generation

Project Generation

Build Changes Types

The variety of possible modifications (project settings, configurations, variables, types, programs) can result in these different build types:

- Build changes in connected mode equal in RUN, what we call online changes (downloaded to the controller without interruption).
- Build changes in local mode.
- Build all.

Rebuild All Requirements

A complete re-build is required in these situations (the build changes are not proposed automatically):

- The application is created.
- The application is imported.
- The application was previously built:
 - Changing of some project settings (all of the language sheets, some of the build (upload, application diagnostic, code generation))
 - controller change
 - memory card change
 - memory card layout change
 - switching from real target to simulator and vice versa

When you edit variables or DFB variables and build the project changes (**Build > Build Changes**), the modifications may create a hole in the memory layout for unlocated variables or DFB instances. In this case, compact and optimize these data by rebuilding the *entire* project (**Build > Rebuild All Project**). For other memory items, the **Build Changes** command is sufficient to optimize the layout.

Build in Online Mode (Permissions)

This table presents authorized and unauthorized modifications in online mode.

It describes the modifications that lead to build changes and indicates if an online change or offline change is possible.

NOTE: An offline change implies a full download and requires an application reset. An online change implies a partial download.

Modification	Modify controller Memory?	Allowed in online mode (RUN/STOP)?	Application Download Required?
General			
Name of station, programs, sections	YES if Upload info is selected	YES	No
Comment of Station, configuration, programs, sections	YES if Upload info is selected	YES	No
Documentation summary	NO	YES	NO
Animation table	NO	YES	NO
Integrated Operator Screens	NO	YES	NO
Functional view	YES	YES	NO
Security information: passwords, protection attributes, ...	NO	YES	NO
Program			
Modify the code of a section (task section, SR, Transition, Action, DFB sections)	YES	YES	NO
Sections of programs: add, delete, change execution order	YES ⁽¹⁾	YES	NO
Modify the code of EVT sections	YES	NO	YES
Modify the code of SFC Chart	YES	YES	NO ⁽⁴⁾
Configuration/communication			
M340 configuration: add/move/remove a module	YES	NO	YES
M340 configuration: change model configuration	YES		
M580 configuration: add or move a module	YES	YES	YES
M580 configuration: remove a module	YES	NO	YES
M580 configuration: change model configuration	YES	NO	YES
Premium Configuration: add/move/remove a module	YES	NO	YES
Premium Configuration: change module parameters	YES	YES	NO
Quantum Configuration: add/move/remove a module	YES	in STOP only	NO Limitations?
Quantum Configuration: change module parameters	YES	in STOP only	NO
Changing memory sizes in configuration screen	YES	NO	YES
Global variables (used even in animation table or Operator Screen)			
Remove a used variable	YES	NO	YES
Rename a used variable	NO (Increase Fragmentation)	YES	NO
Change the type of a used variable	YES	YES	NO
Change the topological address of a used variable	NO	YES	NO
Change the initial value of a used variable	YES (Cart)	YES	NO
Change a comment of variables for a used variable	YES if Upload info is selected	YES	NO
Create, remove or modify unused variables (EDT,DDT)	YES (In unlocated variable for variable creation)	YES	NO
Create, remove or modify unused variables (FB)	YES (add/Remove membloc)	YES	NO
Used DFBs			
Type name of used DFB	YES if Upload info is selected	YES	YES
All comments	YES if Upload info is selected	YES	NO
Add a parameter (IN / OUT / IN-OUT)	YES	NO	YES

Modification	Modify controller Memory?	Allowed in online mode (RUN/STOP)?	Application Download Required?
Add a private or public variable	YES	YES	NO
Delete or change unused private variable	NO if Online	YES	NO
Initial value of parameters and variables	YES (Cart)	YES	NO
Other attributes of parameters and variables (public and private)			NO
Sections of DFB: add, delete, change execution order	YES (Cart)	YES	NO
Modify the code of a section	YES (Cart)	YES	NO
Create a new DFB type	YES (Cart)	YES	NO
Delete an unused DFB type	YES (Cart)	YES	NO
Used DDT			
Create a new DDT type	YES if Upload info is selected	YES	NO
Delete an unused DDT type	YES if Upload info is selected	YES	NO

Download the Project to the Controller

Optimal communications between the PC and the controller are achieved through the exclusive use of controllers that have embedded USB or Ethernet ports.

- Quantum: For Quantum controllers without embedded Ethernet, use Modbus Plus.
- Premium: For Premium controllers without embedded Ethernet, include an Ethernet module in your design and commissioning phases. The module can be removed for operations.

Online Modification

Introduction

You can modify the different sections of the program in Control Expert when the controller is in RUN.

Sections

To confirm that the modifications are consistent, all modifications are transferred at once when the build is executed in the same cycle.

These are the sections:

- Sections
- SR Sections
- Timer Event Sections
- Sections of a DFB type

Considerations

- You cannot modify I/O events in RUN mode.
- Perform several small, consistent modifications to sections at one time. This facilitates program maintenance and optimizes performance during online modifications.
- The modification of SFC actions or transitions in a single SFC section can result in the transfer of all *Action* sections and *Transition* sections of the corresponding SFC section.

Improve Online Build Performance

General Project Settings

Configure these settings (**Tools > Project Settings > General**):

Setting	Description
<i>Management of build messages</i>	Multiple use of FB instances generates: Make a selection from this pull-down menu to confirm that system control of all DFB\EFB instances are not used twice in the code: • nothing • a warning Turn off any messages that are not required.
<i>Build settings</i>	Virtual connected mode: Select (check) this box to modify the project locally so that it can be transferred later without stopping the controller. NOTE: In this mode, the Build and Rebuild commands are disabled to avoid the modification of the project signature before there is a connection to the controller.
<i>Project autosaving on download</i>	• save STA: Check this box. • save STU: Uncheck this box. • save XVM: Uncheck this box. NOTE: Use the save STA setting to help enhance the security of application modifications.
<i>PLC embedded data</i>	Data dictionary: Uncheck this box when it is not required to reduce the amount of memory that is sent. This reduces the amount of information that is associated with specific variables. NOTE: • The data dictionary is useful for SCADA system or Vijeo Designer HMI. • To reduce the number of variables, use the HMI variable only when it is required. Upload information: If the Upload information options that are configured in the zone are embedded in the controller after the transfer of the binary (Download), they consume memory in the program zone. Configure Upload information to execute these functions: • Authorize the project upload (Project Settings > PLC embedded data). • Reconstruct the graphical part of the project. Upload information management: If the Upload information management option is in Automatic mode, the upload information is transferred automatically to the controller with each online modification. To optimize the online modification, select the On user request mode to manually transfer the upload information to the controller before the offline mode by the user (Project Settings > PLC embedded data > Upload information). NOTE: Choose the Automatic mode to help enhance the security of application modifications. Optimize data on-line change: Check this box to reduce the time that is consumed to generate the upload information that corresponds to project variables and function blocks.

Considerations

- Perform several small consistent modifications at once. When a section is modified, the complete section is rebuilt. Small modifications to multiple individual small sections can result in long build times.
- When Ethernet communications are used in the controller configuration (**Maximum Unity Data exchanged by Plc Scan**): If you change it to 12x1024, the build time speeds up, but scan rates are negatively affected. You may want to experiment with this setting during the commissioning phase before switching back for RUN mode. In general, use 12x1024 only during commissioning, not in production.
- The modification of SFC actions or transitions in a single SFC section can result in the transfer of all *Action* sections and *Transition* sections of the corresponding SFC section.

Improve Offline Build Performance

Settings

To optimize offline build-time performance, adjust these **Project Settings (Tools > Project Settings > General)**:

Setting	Description
<i>Management of build messages</i>	Turn off any messages that are not required.
<i>PLC diagnostics</i>	Turn off application diagnostics when they are not in use.

NOTE: Perform several small consistent sections modification at once. When any modification is done in a section, this entire section is rebuilt. A few small modifications on several sections can therefore result in long build times.

Import Functions

Introduction

Control Expert allows you to export and import .xml code (either the entire application or selected sections only).

File Extensions

These file types can be imported to Control Expert:

Extension	Application
.xef	complete project
.xhw	hardware configuration
.xpg	tasks (Mast \ Fast \ AUX)
.xld	ladder sections or subroutines (SR)
.xil	list sections or subroutines (SR)
.xst	Structured Text sections or subroutines (SR)
.xbd	Function Block Diagram sections or subroutines (SR)
.xsf	Sequential Functional Chart\ Macro
.xdb	DFB types
.xdd	DDT types
.xsy	all variables (EDT\DDT\DFB\EFB\IODDT)
.xfm	functional modules
.xtb	animation tables
.xcr	run time screens
.xcm	network configuration

To maintain consistency among the exported files, Control Expert exports the data that is used in the exported file. You can also export the used SRs, DDTs, and DFBs. Every time that an .xml file that contains data is imported, a check of the cross-reference in the project database is applied, considerably increasing the import time.

General Considerations

- A partial import facilitates the updating of the application by adding an independent object.
- Do not use the partial import in these situations:
 - You create new variables with an .xsy file that contains all of the data in the existing application.
 - You import a complete .xpg file to the existing application.

NOTE: In these cases, use a complete import of the .xef file.

Save and Restore the Latest Variable Values

Introduction

In these cases, it may be necessary to restore the latest variable values in the controller:

- You modify an application that requires local generation.
- You are debugging an application.
- There is a cold restart of the controller.
- The product recipe is changed.

These are the characteristics of the scenarios that are described below:

- There are different methods for restoring values in these scenarios.
- You can define a list of variable types to be restored for each scenario.

Variable Support for Restoring Variable Values

Located variables:

- These internal located variables support the restoration of values:
%M, %MW, %MD, %MF

NOTE: Constants (%KW, %KD, %KF) are not included in the list of variables because their values do not change.

Unlocated variables:

- simple (EDT)
- structured (DDT)
 - arrays (anonymous or with defined types)
 - structures
 - SFC-dedicated structures
- derived function blocks
 - input parameters
 - output parameters
 - public and private variables
- elementary function blocks (EFBs):
 - input parameters
 - output parameters
 - public variables

NOTE: Depending on the particular circumstances, it may not be possible to restore the values for some of these variable types. When applicable, the variables are mentioned specifically below.

Restore Variable Values

Methods According to Requirement: Use this table to select a method for restoring the variable values:

Requirement	Method
Save a specific context from the application (recipe) and select a new production context.	.dat files
	communication module (data storage)
Save the adjustment parameters for the application so that it can be restarted with these initial parameters.	manual command/system bit
Transfer a new version of the application without losing the latest production data.	specific macro
	.dat files
Perform a cold restart.	controller configuration option

Restore Values with a DAT File

Restore Values with a DAT File. Follow the steps below to restore the values of all located (configurable number) and unlocated variables in Control Expert.

Change a Context (Recipe):

Step	Action
1	Save the latest variable values in the .dat file (PLC > File Transfer). NOTE: The latest values for located variables can be restored to different applications.
2	Select the .dat file that contains the new production recipe. NOTE: The .dat file is independent of the controller application.
3	Restore the variable values (PLC > File Transfer).

Considerations:

- If the controller is in RUN, it can take several controller scan cycles to save the latest values to the .dat file, which can desynchronize the variable values. In this case, perform one of these actions:
 - Launch the save operation in a phase of the process at which the variable values are stable.
 - Launch the save operation in STOP mode.
- Unlocated variables of the types *Timer* and *Counter* restart with their latest values, as restored from the .dat file.
- For applications that use SFC sections, save the latest values of structures (SFCSTEP_STATE, SFCSTEP_TIME, SFCCHART_STATE) that are linked to each SFC section to create an image of how the harts have evolved at any given time. Application charts evolve between the times the last values are saved and restored, yet it is the state saved at the time of the save operation that is restored. This can be problematic for the process that is currently running. For these applications, set the controller to STOP and initialize the charts (initial steps) before performing the save operation. The controller can be set to RUN after the latest values are restored.

Now you can transfer a new version of the application.

If a modified application requires you to RebuildAll and Download, restart with the latest values:

Step	Action
1	Save the latest variable values in the .dat file (Save Data from PLC to File).
2	Modify the application and rebuild Rebuild All Project and transfer (Transfer Project to PLC) the project.
3	Download the application to the controller.
3	Restore the latest variable values from the .dat file using the File > PLC Transfer command.

NOTE: The restoration of the latest values of unlocated variables stops when an application modification affects the database after the deletion of unlocated variables.

Restore Values with a Controller Configuration Option

This method restores the values of these located variables:
%MW, %MD, %MF (not %M)

This solution is used after a cold start, a download, or whenever a requirement to restore the latest values is applicable to located variables.

Instructions:

Step	Action
1	Open the controller configuration screen.
2	Deselect the Initialize %MWi on cold start option. NOTE: This function is available in Unity Pro version 2.3 and subsequent supporting versions of Unity Pro and Control Expert.

Restore Values with a Control Expert Command or System Bit %S94

Use this method to restore the latest values of located and unlocated variables.

This solution is used when debugging or adjusting the application if the user wishes to replace the initial values with the latest values upon the restart of the application.

Instructions:

Step	Action
1	Select the Save option in the variables editor for each unlocated variable.
2	Use one of these methods to replace the initial values with the latest values: • Update the initial values from the PLC menu (Update Init Values with Current Values). • Set system bit %S94 to 1 in the application.
3	Restart the application.

NOTE:

- If an application modification requires a download, it is necessary to upload the application after replacing the initial values with one of the methods described above. For large applications, these different stages in the procedure take a considerable amount of time, and the restored (latest) values may not represent the values of the process at the time that the modified application was downloaded.
- The "Save" option on unlocated variables is available for:
 - simple EDT variables
 - DDT instances (including their elements)
 - DFB-type public and private variables

Restore Values with a Specific Macro

Use a specific macro to restore the latest values of located and unlocated variables. Use this solution when a modification to the application could affect the unlocated variables (creation/deletion) and requires a download.

The **VB.NET macro to Save and Restore current values** is available on the CoP (Community of Practice). It is accompanied by a manual that describes the use of the macro. This macro is applicable to applications developed using Unity Pro V2.2 or any subsequent supporting version(s) of Unity Pro or Control Expert.

Observe these limitations:

- The latest values of these variables are not saved:
 - EFBs that belong to the obsolete library: PL7_TON, PL7_TOF, PL7_TP, PL7_3_Timer, PL7_Monostable, PL7_Register_32, PL7_Counter
 - DDT structures that are dedicated to steps in SFC section charts
 - diagnostic DFBs
- For the TON, TOF, and TP timer EFBs, neither of these values is accounted for when the application restarts:
 - elapsed time before the application is set to STOP (to save the latest values)
 - elapsed time between the save and restore operations

In these cases, the timers are reinitialized and re-triggered.

- Values are not restored for variables with a modified type. For a structured variable (DDT) or a derived function block (DFB) variable, none of the latest values of any of the other elements belonging to the variable are restored when the type of an element is modified.

Restore Values with a Communication Module

Use a communication module to restore the latest variable values of these located variables:

%MW, %MD, %MF (not %M)

This solution is used when modifying a production recipe.

Platform	Description
Quantum	Use <code>Write_PCMCIA</code> and <code>Read_PCMCIA</code> to store and restore data in the PCMCIA data storage zone.
Premium	
M580	Use <code>Wr_data_to_file</code> and <code>Rd_file_to_data</code> to store and restore data in the SD Card data storage zone.
M340	

Migration of Control Expert Projects

Introduction

An embedded database is rebuilt for each different version of Control Expert. You cannot, therefore, use the Control Expert project file (.stu) to migrate a project from one version of Control Expert to another. To migrate the project, use a Control Expert archive file (.sta).

File Characteristics

A file in the .sta format offers these advantages:

- The .sta file is smaller than the corresponding .stu file because it does not contain database information. It can therefore be exchanged.
- After a migration, you can connect directly to the controller without additional project generations or download actions.

Library Version

After you install Control Expert, all of the libraries that belong to different versions of Control Expert are present on the station. When you open the .sta file, you do not have to generate or download the project unless the original project library (from the .sta file) has changed. The functionality of the previous application is preserved through the retention of the original library.

Select a specific library version when you open the .sta file:

Step	Action
1	Access the Open dialog box (File > Open).
2	In the pull-down menu next to the File name field, select Archived Application Files (.STA) .
3	Use standard Windows commands to navigate to the appropriate .sta file and select it. NOTE: When you select a file in the .sta format, the radial buttons under Libset version become active and available.
4	Highlight this radial button: Select a specific target version NOTE: The version pull-down menu appears.
5	Select the appropriate Control Expert version from the pull-down menu.
6	Click the OK button and follow the prompts to implement the selected library version.

Duplicate a Function Block Instance

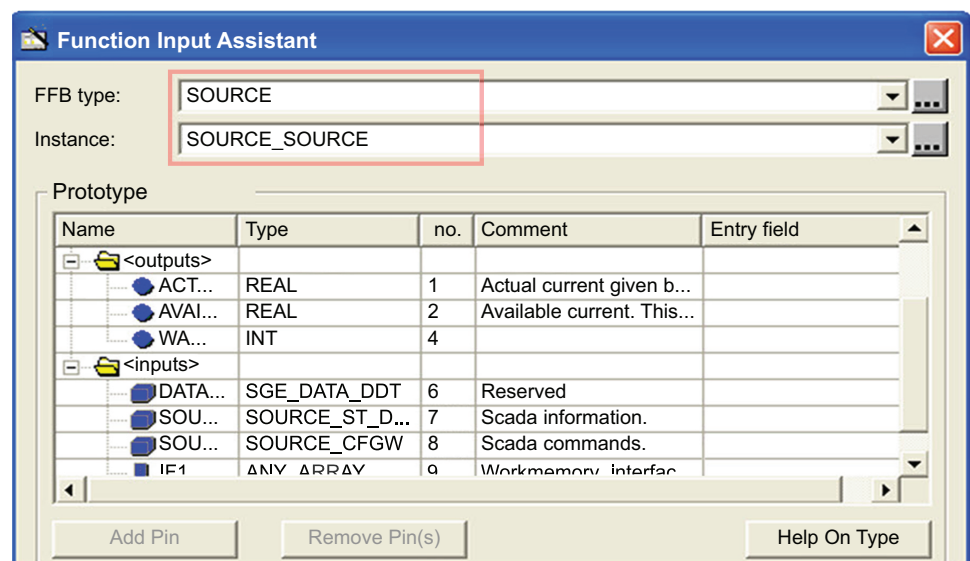
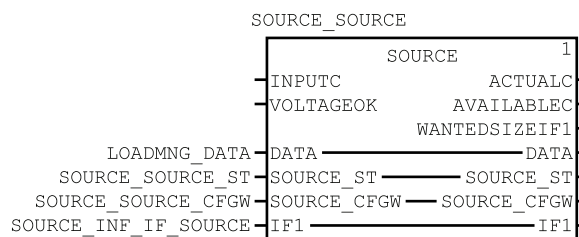
Introduction

Use these instructions to duplicate an instance of an elementary function block (EFB) in a graphical section.

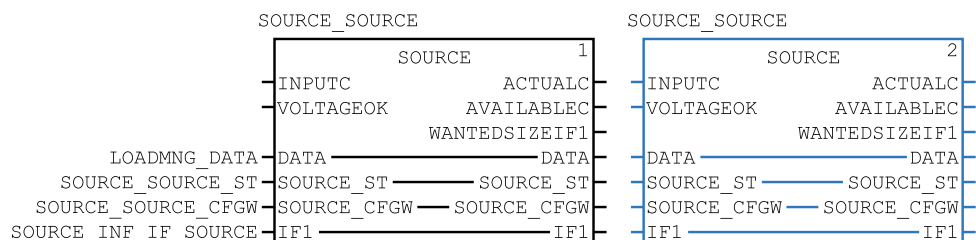
Copy and Paste

In the graphical IEC languages (LD and FBD), copy and paste a function block instance (EFB \ DFB) to automatically increment the name of the instance. For example, the instance `My_Instance_Name` becomes `My_Instance_Name_0`.

This can be useful when you want to use the same instance several times in the program. In that case, call the *FFB Input Assistant* from the contextual menu of the section.



Select the EFB type and the instance name to see the same instance block in the section:



OFS and Control Expert Synchronization FB Instances

Introduction

This chapter describes the synchronization of FB Instances values between OFS and Control Expert.

To confirm that SCADA systems account for all of the online modifications in the Control Expert application, observe these considerations. Otherwise, SCADA animation can stop on some variables, or some screens may not be refreshed.

Instructions

Configure dynamic consistency:

Step	Action
1	Select (check) Dynamic consistency in the OFS configuration tool.
2	Confirm that the Reload Database parameter is selected (checked) in the <i>Project files options</i> .
3	Select (check) the Project autosaving on download option for the .stu and/or .vxm files in the Options dialog box.

NOTE: A data dictionary that contains the description of all located and unlocated variables can be configured (**Tools > Project Settings > General > PLC embedded data > Data dictionary**) in the internal memory or in the extended memory if configured. When configured, the OFS server uses the Data Dictionary instead of the .vxm or .stu files.

Modify the IP Address

Invoke New IP Parameters

Invoke the new IP address for the controller

Step	Action
1	Power down the controller.
2	Power up the controller.
3	Confirm that the connector is present on the controller to render its previous IP address obsolete.

Glossary

B

bridge:

A bridge device connects two or more physical networks that use the same protocol. Bridges read frames and decide whether to transmit or block them based on their destination address.

D

DFB:

(*derived function block*) DFB types are function blocks that can be defined by the user in ST, IL, LD or FBD language.

Using these DFB types in an application makes it possible to:

- simplify the design and entry of the program
- make the program easier to read
- make it easier to debug
- reduce the amount of code generated

DTM:

(*device type manager*) A DTM is a device driver running on the host PC. It provides a unified structure for accessing device parameters, configuring and operating the devices, and troubleshooting devices. DTMs can range from a simple graphical user interface (GUI) for setting device parameters to a highly sophisticated application capable of performing complex real-time calculations for diagnosis and maintenance purposes. In the context of a DTM, a device can be a communication module or a remote device on the network.

See FDT.

E

EFB:

(*elementary function block*) This is a block used in a program which performs a predefined logical function.

EFBs have states and internal parameters. Even if the inputs are identical, the output values may differ. For example, a counter has an output indicating that the preselection value has been reached. This output is set to 1 when the latest implemented value is equal to the preselection value.

EtherNet/IP:

A network communication protocol for industrial automation applications that combines the standard internet transmission protocols of TCP/IP and UDP with the application layer common industrial protocol (CIP) to support both high speed data exchange and industrial control. EtherNet/IP employs electronic data sheets (EDS) to classify each network device and its functionality.

Ethernet:

A LAN cabling and signaling specification used to connect devices within a defined area, e.g., a building. Ethernet uses a bus or a star topology to connect different nodes on a network.

F

FDR:

(*fast device replacement*) A service that uses configuration software to replace an inoperable product.

G

gateway:

A device that connects networks with dissimilar network architectures and which operates at the Application Layer of the OSI model. This term may refer to a router.

gateway:

A gateway device interconnects two different networks, sometimes through different network protocols. When it connects networks based on different protocols, a gateway converts a datagram from one protocol stack into the other. When used to connect two IP-based networks, a gateway (also called a router) has two separate IP addresses, one on each network.

H

Hot Standby:

A Hot Standby system uses a primary PAC (PLC) and a standby PAC. The two PAC racks have identical hardware and software configurations. The standby PAC monitors the current system status of the primary PAC. If the primary PAC becomes inoperable, high-availability control is maintained when the standby PAC takes control of the system.

HTTP server:

The installed HTTP server transmits Web pages between a server and a browser, providing Ethernet communication modules with easy access to devices anywhere in the world from standard browsers such as Internet Explorer or Netscape Navigator.

I

IP address:

Internet protocol address. This 32-bit address is assigned to hosts that use TCP/IP.

IP address:

The 32-bit identifier, consisting of both a network address and a host address assigned to a device connected to a TCP/IP network.

L

local rack:

An M580 rack containing the CPU and a power supply. A local rack consists of one or two racks: the main rack and the extended rack, which belongs to the same family as the main rack. The extended rack is optional.

M

MAC address:

media access control address. A 48-bit number, unique on a network, that is programmed into each network card or device when it is manufactured.

MB/TCP:

(Modbus over TCP protocol) This is a Modbus variant used for communications over TCP/IP networks.

P

PLC:

programmable logic controller. The PLC is the brain of an industrial manufacturing process. It automates a process as opposed to relay control systems. PLCs are computers suited to survive the harsh conditions of the industrial environment.

port 502:

TCP/IP reserves specific server ports for specific applications through IANA (Internet Assigned Numbers Authority). Modbus requests are sent to registered software port 502.

R

RIO network:

An Ethernet-based network that contains three types of RIO devices: a local rack, an RIO drop, and a ConneXium extended dual-ring switch (DRS). Distributed equipment may also participate in an RIO network through a connection to DRSs or BMENOS0300 network option switch modules.

router:

A router device connects two or more sections of a network and allows information to flow between them. A router examines every packet it receives and decides whether to block the packet from the rest of the network or transmit it. The router attempts to send the packet through the network on an efficient path.

S

SNMP agent:

The SNMP application that runs on a network device.

SNMP:

(simple network management protocol) Protocol used in network management systems to monitor network-attached devices. The protocol is part of the internet protocol suite (IP) as defined by the internet engineering task force (IETF), which consists of network management guidelines, including an application layer protocol, a database schema, and a set of data objects.

SNMP:

simple network management protocol. The UDP/IP standard protocol used to monitor and manage devices on an IP network.

SNTP:

(simple network time protocol) See NTP.

SOE:

(sequence of events) SOE software helps users understand a chain of occurrences that can lead to unsafe process conditions and possible shutdowns. SOEs can be critical to resolving or preventing such conditions.

subnet mask:

The 32-bit value used to hide (or mask) the network portion of the IP address and thereby reveal the host address of a device on a network using the IP protocol.

subnet mask:

The subnet mask is a bit mask that identifies or determines which bits in an IP address correspond to the network address and which correspond to the subnet portions of the address. The subnet mask comprises the network address plus the bits reserved for identifying the subnetwork.

subnet:

The subnet is that portion of the network that shares a network address with the other parts of the network. A subnet may be physically or logically independent from the rest of the network. A part of an Internet address called a subnet number, which is ignored in IP routing, distinguishes the subnet.

switch:

A network switch connects two or more separate network segments and allows traffic to be passed between them. A switch determines whether a frame should be blocked or transmitted based on its destination address.

T**Transparent Ready:**

Schneider Electric's Transparent Ready products (based on universal Ethernet TCP/IP and Web technologies) can be integrated into real-time, data sharing systems, with no need for interfaces.

U**UDP:**

user datagram protocol. UDP is an Internet communications protocol defined by IETF RFC 768. This protocol facilitates the direct transmission of datagrams on IP networks. UDP/IP messages do not expect a response, and are therefore ideal for applications in which dropped packets do not require retransmission (such as streaming video and networks that demand real-time performance).

C

Control Expert	
data types	14
derived	14
elementary	14
download project.....	30
file extensions.....	33
function blocks	16
addressing mode	18
derived	16
duplicate instance	40
elementary	16, 18
purge	19
synchronize instances	41
generate project.....	27
guidelines.....	8
IP address	
modification.....	42
online modification	41
program organization	
addressing mode	18
data types	14
function blocks	16
multitasking.....	12
program sections	20
subroutines	13
units	13
variables	14
programming languages	
functional view.....	21
performance in RUN	20
physical station configuration	22
project management	
build project	26
build requirements	27
download to controller	30
duplicate function block instance.....	40
file extensions	33
generate project	27
import function.....	33
increase performance	26
IP address modification	42
migrate projects.....	39
offline build performance	32
online build performance	31
online mode build	28
online modification	30
restore latest values	34
purge	
function block instances	19
function block types	19
unused types.....	19
variables	19
restore latest values	
command bit	37
communication module	38
configuration option	36
.dat files	35
macro support	37
system bit %94	37
variables	14
function blocks	16
located variables	14
purge variables.....	19
restore values.....	34

Schneider Electric
35 rue Joseph Monier
92500 Rueil Malmaison
France

+ 33 (0) 1 41 29 70 00

www.se.com

As standards, specifications, and design change from time to time,
please ask for confirmation of the information given in this publication.

© 2023 Schneider Electric. All rights reserved.

EIO0000004875.00