

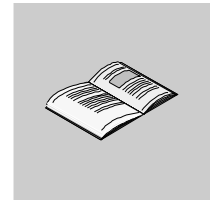
Unity Application Generator (UAG)

Version 2.1

User Manual

eng

Table of Contents



About the Book	11
Part I Understanding Unity Application Generator	13
Overview	13
Chapter 1 Introduction to Unity Application Generator	15
Overview	15
What is Unity Application Generator?	16
The foundations of Unity Application Generator	19
Building an Application	22
Chapter 2 The Physical Model	25
Overview	25
2.1 Overview of the Physical Model	27
Structure of the Physical Model	27
2.2 The Elements Area, Process Cell and Unit	29
Area, Process Cell and Unit Description and Properties	29
2.3 The Element Equipment Module	30
Overview	30
What is an Equipment Module?	31
Equipment Module Features	32
Equipment Module Properties	33
2.4 The Element Control Module and Control Module Types (Smart Control Devices)	35
Overview	35
Introduction	36
Control Module Features	37
Control Module Types or Smart Control Devices (SCoDs)	38
Free Control Module	40
Control Module Properties	41
Instruments	42
Interlocks for Control Modules	43
Links between SCoDs	46

2.5	Variables	47
	Overview	47
	Introduction	48
	Connection Types and Data Types of Variables	49
	General Variable Properties	51
	HMI Related Variable Properties	53
Chapter 3	The Topological Model	59
	Overview	59
3.1	Control System Topology and Topological Model	61
	Overview	61
	The Topology of a Control System	62
	Structure of the Topological Model	67
	Communication via Modbus Plus and Ethernet	69
	Additional Information Concerning Ethernet	71
	Quantum Hot Standby Configuration (HSBY)	74
3.2	The Groups Network Segments and Routing Paths	78
	Overview	78
	Network Segment Description and Properties	79
	List of Network Nodes	80
	Routing Paths Description and Properties	82
3.3	The PLC Group	84
	Overview	84
	Introduction	85
	PLC Properties	87
	PLC Channels	90
	Copy and Paste of PLCs	92
	PLC <-> PLC Communication via Modbus Plus	93
	PLC <-> PLC Communication via Ethernet	95
	Additional Racks	97
	Racks and Modules	98
	Enhanced Ethernet Module	107
3.4	The HMI Group	110
	Overview	110
	Introduction	111
	HMI Properties	112
	Control Domain Properties	113
3.5	The Data Server Group	114
	Data Server - Description and Properties	114
3.6	The Network Nodes Group	117
	Overview	117
	Net Partner Description	118
	Net Partner Properties	119
	Other Node Description	120
	Other Node Properties	121

Part II	Working with Unity Application Generator	123
	Overview	123
Chapter 4	Rules for Working with Unity Application Generator	125
	Overview	125
	General Rules for Project Configuration and Generation	126
	Open Customization and SCoD Editor	127
	Rules Concerning the PLC and Concept	128
	Rules Concerning the PLC and Unity Pro	130
	Rules Concerning HMI	131
Chapter 5	Tool Handling and Features for Effective Work.	133
	Overview	133
	Concepts of the User Interface	134
	Working with Tables (Lists)	137
	Drag & Drop of Objects, Modules and Variables.	139
	How to Find Objects with Search Criteria	140
	Working with the Instrument List	142
	Copy and Paste in the Physical and Topological Model	145
	How to Build an Interlock Definition.	147
	Working with the Topological Viewer	151
	How to View the Generation Status	154
Chapter 6	Workflow to Build an Application	155
	Overview	155
	General Workflow to Build an Application	156
	Defining the Customization	158
	Defining the Physical Model	159
	Defining the Topological Model.	160
	Complete the Physical Model	163
	Generate PLC and HMI Applications	164
	Generate Import File for Net Partners.	166
	Document the Application or Individual Objects (Report)	167
Part III	Project Management	169
	Overview	169
Chapter 7	Managing Distributed Project Development - Project Merge	171
	Overview	171
	Overview	172
	Code Generation for Individual PLCs	173
	Workflow of Distributed Project Development.	174
	General Preconditions for Project Merge	177
	Merging of Physical Models	178
	Merging of Topological Models	180

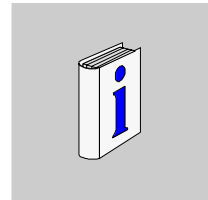
Chapter 8	Interfaces with other Tools (Import and Export Features)	183
	Overview	183
	Concept of Open Interfaces	184
	How to Import/Export CSV Files	186
	Example: Import Instruments / Physical Model Hierarchy	188
	Example: Import Initial Values for Variables	191
Chapter 9	Supported HMIs and their Setup	193
	Overview	193
9.1	Monitor Pro and Unity Application Generator	195
	Overview	195
	Introduction	196
	Monitor Pro - Default Server Application	197
	Generation Modes of Monitor Pro Server Application	200
	Data Conversion Monitor Pro vs. Unity Application Generator	202
	Monitor Pro Drivers and Communication	206
	Alarming	210
	Archiving	215
	Monitor Pro Client Application	221
9.2	iFIX and Unity Application Generator	226
	Overview	226
	Introduction	227
	How to Configure iFIX for the Use with Unity Application Generator	228
	Manual Configurations Before Generation for iFIX	230
	How to Generate a New iFIX Application	233
	How to Generate an iFIX Application Incrementally	234
	Unity Application Generator and iFIX Drivers	235
	How to Deploy the Generated Application to the iFIX Nodes	236
	How to Run an Existing Unity Application Generator Project	237
	Configuring iFIX Redundancy	238
9.3	Generic HMI and Unity Application Generator	239
	Using Unity Application Generator with a Generic HMI	239
Chapter 10	Customization and Project Maintenance	241
	Overview	241
10.1	Customizing Unity Application Generator	243
	Overview	243
	Introduction	244
	Working with the Customization Editor	245
	The Customization Options	247
	Defining Naming Conventions	249
	User Defined Modules - Overview	251
	User Defined Modules - Properties	253
	How to Define a Generic Module	255

	How to Define a ModConnect Partner Module	256
	How to Change the Customization	257
10.2	Project Maintenance	258
	Overview	258
	Setting Options for Analysis and Generation	259
	Version Management and Change Tracking.	260
	Project Documentation (Report Generator).	262
	Trouble Shooting.	262
Appendices	263
	Overview	263
Appendix A	Release Notes Version 2.1	265
	Overview	265
	New Features in Unity Application Generator Version 2.1	266
	Hardware Requirements	266
	Software Requirements.	267
	Installation information for new Users	268
	Upgrade of Existing Projects to UAG 2.1 and Concept V2.6	271
Appendix B	Generated Code	273
	Overview	273
B.1	Overview of Generated Code and Generation Principles	275
	Overview	275
	Overview of Generated Code	276
	Generation Principles	277
B.2	Generation for Concept.	278
	Overview	278
	What is generated?	279
	Generation from General Project Settings - Overview	280
	Generation from the Topological Model - Overview	281
	Generation from the Physical Model - Overview.	287
	Generated PLC Configuration.	294
	Generated Variables	295
	Generated Code: Equipment Modules, Control Modules and Interlocks.	297
	Generated Code: Communication.	300
	Generated Code: Initialization.	302
	Generated Code: Scaling of Analog Values (Quantum only)	303
	Generated Code: Hot Standby	303
B.3	Generation for Unity Pro	304
	Overview	304
	What is generated?	305
	Generation from General Project Settings - Overview	306
	Generation from the Topological Model - Overview	307
	Generation from the Physical Model - Overview.	313

	Generated PLC Configuration	320
	Generated Variables	320
	Generated Code: Equipment Modules, Control Modules and Interlocks	323
	Generated Code: Communication	326
	Generated Code: Initialization (Quantum only)	328
	Generated Code: Scaling of Analog Values (Quantum only).	328
	Generated Code: Discrete Configuration (Premium only)	329
B.4	Generation for Monitor Pro	332
	Overview	332
	Introduction	333
	Generated Variables and their Graphical Representation in the HMI	333
	Generated Screens	335
	Generated Monitor Pro Database Objects.	336
	Generated Monitor Pro Pictures	340
B.5	Generation for iFIX.	344
	Overview	344
	Characterization.	345
	Generated Variables and their Graphical Representation in the HMI	346
	Generated Screens	347
	Generated iFIX Database Objects.	348
	Generated iFIX Pictures.	354
	Generated iFIX Driver Configuration from Unity Application Generator Point of View	359
	Generated iFIX Driver Configuration from the Driver Point of View.	361
B.6	Generation for a Generic HMI	363
	Generation for a Generic HMI	363
B.7	Generation for Net Partners.	364
	Overview	364
	Generation for Net Partners.	365
	Net Partner Variables: CSV File Format	366
Appendix C	Format of the CSV Files for Import and Export.	367
	Overview	367
	General Format	368
	Physical Model Elements: CSV File Format	370
	Topological Model Elements: CSV File Format	376
	Instruments: CSV File Format	385
	Initial Values: CSV File Format	386
Appendix D	Format of the XML File for Generic HMI	389
	XML File Format for Generic HMI	389

Glossary	401
Index	409

About the Book



At a Glance

Document Scope

This manual is

- to let you understand what Unity Application Generator (UAG) is and what it can do for you.
- to get an overview of the functionality of UAG.
- to explain all elements which are used to build an application.
- to give you a clear roadmap how to create an application with UAG.
- to give reference to all elements of UAG.

It is **not** the intention of the manual

- to give a reference to all menus and dialogs of Unity Application Generator. This you will find in the Help of the software.

Validity Note

This manual is valid for Unity Application Generator (UAG) V2.1 in conjunction with:

- Unity Pro V2.0
 - Concept V2.6 SR2
 - Concept V2.6 SR1
 - Concept V2.5 SR2
 - Monitor Pro V7.2
 - iFIX V2.6
 - Modbus/Ethernet MBT V2.0 Driver and V3.0
-

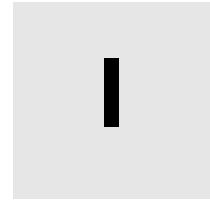
Related Documents

Title of Documentation	Reference Number
SCoD Library IATBASIC10; refer to separate Word documents provided for each SCoD in the DOC subdirectory of the Unity Application Generator directory	-
SCoD Editor V2.1 User Manual	-
Unity Pro Software Reference Manual	-
Quantum Hot Standby with Unity Pro User Manual	UNY USE 10710
Concept User Manual	840 USE 503 00
Quantum Hot Standby Planning and Installation Guide with Concept	840 USE 106 00
D. W. Fleming, V. Pillai: S88 Implementation Guide	McGraw Hill, ISBN 0-07-021697-5

User Comments

We welcome your comments about this document. You can reach us by e-mail at techpub@schneider-electric.com

Understanding Unity Application Generator



Overview

Introduction

UAG is a new kind of process design tool for process automation. For efficient work it is required to understand the basic principles of the tool. This part will explain these principles to you.

In addition it will explain the details of all elements within UAG, which are used to describe the process.

What's in this Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
1	Introduction to Unity Application Generator	15
2	The Physical Model	25
3	The Topological Model	59

Introduction to Unity Application Generator

1

Overview

Introduction

This chapter introduces the principles of the Unity Application Generator and the benefits for the user. It explains the general concept and the process to develop an application with UAG.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
What is Unity Application Generator?	16
The foundations of Unity Application Generator	19
Building an Application	22

What is Unity Application Generator?

What is Unity Application Generator?

Unity Application Generator is a new kind of process design tool for process automation. It closes the gap which normally exists between the process engineer and the control engineer. In the past, both groups have worked with specialized tools which were not compatible. Changes to the process design defined by the process engineer had to be repeated by the control engineer. This situation is now changed with UAG.

For the process engineer UAG allows the user:

- to define a general layout of the process based on objects defined in the Physical Model of ISA S88.01 standard like Area, Process Cell, Unit, Equipment Module and Control Module and
- to link from UAG objects to his basic tools like E-plan, Autocad, P&ID drawings etc.

For the control engineer UAG allows the user:

- to build the control architecture with PLCs, HMIs and networks as the Topological Model
- to assign the control logic to the objects the process engineer has defined
- to generate 30 - 50% of the control logic for the PLC and the HMI from the process design

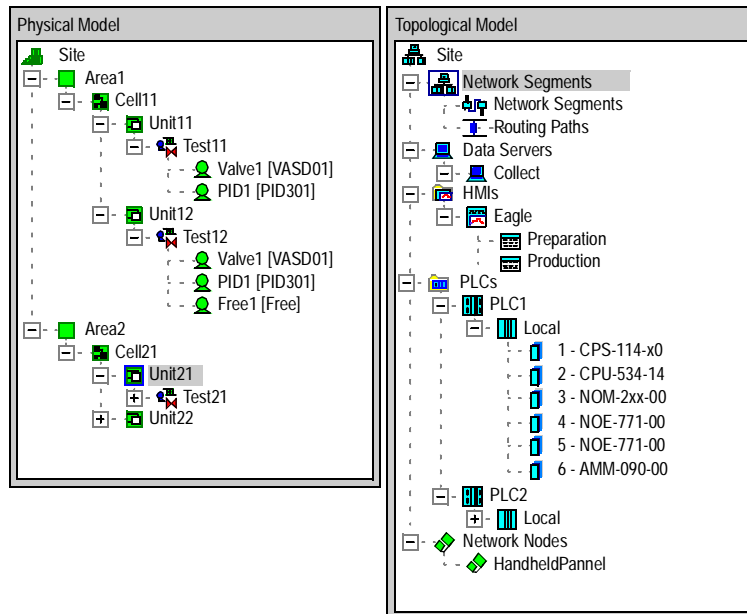
Supporting this approach, UAG allows the user to design a distributed control system with multiple PLCs and HMIs in one step.

What does it look like?

The frame window of Unity Application Generator displays the Physical Model and/or the Topological Model as an object tree in a similar way to Windows Explorer.

- The Physical Model is typically designed by the process engineer. The elements of the Physical Model are mapped to the resources in the Topological Model.
- The Topological Model is typically designed by the control engineer.

The following figure shows the frame windows of the Physical Model and the Topological Model of an example process:



What is the Output of UAG?

Unity Application Generator generates code for:

- PLC
Logic for Concept/Unity Pro, the IEC 61131 programming software of Schneider Electric.
- Monitor Pro
HMI tags and graphics related to the objects created with UAG for Monitor Pro.
- iFIX
HMI tags and graphics related to the objects created with UAG for Intellution's iFIX.
- Generic HMI
HMI tags with all relevant information stored in a data exchange file to be used by any HMI that has an import functionality. See also *Using Unity Application Generator with a Generic HMI*, p. 239.

The output of UAG is downloaded directly to the PLC or the HMI. It builds the static process design (about 30% of the total design).

To build the final application, the output of UAG has to be completed by adding the dynamic behaviour of the process to the PLC and the HMI system.

Additional Features and Benefits

Additional features of the Unity Application Generator are:

- One database for PLC and HMI parameters. The communication is always consistently defined between PLC and HMI.
- Object oriented design with reusable technological objects like motors, valves, pumps etc.
- Change tracking system allowing the user to keep track of all changes done by multiple users who can work on the project in parallel.
- Use of industry standards and quasi-standards like ActiveX.
- Documentation like CAD drawings, text documents etc. can be linked to the respective objects.

Benefits for the user of the Unity Application Generator are:

- Reduced application and test effort of the system due to reusable objects.
 - Faster design, thus reduced total life cycle costs.
 - Quicker maintenance of the system.
 - Faster re-validation of the process design with the help of the change tracking system. Every change of the process will be logged by the `Change History`.
 - Facilitated validation effort for projects which must be validated due to state regulation.
 - Significantly increased application software quality.
 - Reduced time required for installation and commissioning.
-

The foundations of Unity Application Generator

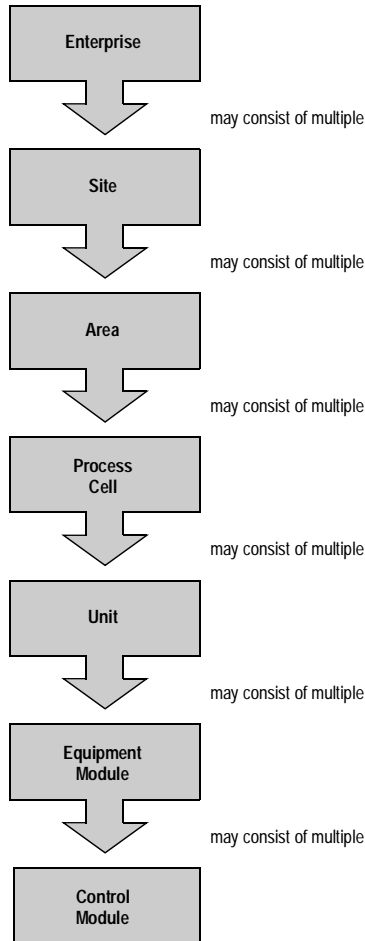
ISA S88.01 compliance

Unity Application Generator uses the terminology of the ISA Standard S88.01 for Batch Control Part 1 "Models and Terminology". By adopting the S88 structure, the user of Unity Application Generator is able to break his process tasks down in-line with the standard, and then carry out a bottom up implementation, assisted by Unity Application Generator.

The S88 standard describes a hierarchy to structure the complete automation facilities in an enterprise as the Physical Model. In this release of Unity Application Generator the decomposition of an enterprise starts at the Site level.

The figure shows the complete hierarchy the designer can define to describe the automation of the Site. The lines symbolize a one-to-many relation, e.g. a Unit consists of one or multiple Equipment Modules. The Areas, Process Cells and Units are used to structure the plant into parts that perform different tasks.

Physical Model according to ISA S88.01



Single Process Database

Building an automation application using Unity Application Generator produces in one step the output for:

- the HMI
- the PLC
- the communication

This enables the user to have one single process database which stores all the process data for both HMI and PLC. Thus both applications, HMI and PLC, which had been separated in the past now work with the same data model and construct one single process control solution.

**New Object:
Smart Control
Devices (SCoD)**

A Control Module is a generic term used to define sensors, actuators and regulatory control equipment that, from a control viewpoint, are operated as a single entity. A Control Module may be an object in the real world, which can be picked up and looked at, e.g. motor, valve, temperature transmitter, etc. or it can be a logical software object, which is used for regulation control or other control functions, e.g. PID loop, timer, counter.

A Smart Control Device (SCoD) is a predefined Control Module of a special type within Unity Application Generator, a so called Control Module Type.

A SCoD can be instantiated as many times as required by the process. It is fully tested. This will reduce the costs for testing the whole application significantly.

A SCoD contains:

- The PLC logic required to control a real world device of a defined type.
- Animated HMI symbols.
- Full manual control of the Control Module through the HMI.
- Access to all variables required for the PLC and the HMI.
- Connection to the HMI alarm management.

**Features for 21
CFR Part 11**

To fulfill the 21 CFR Part 11 requirements UAG provides the following information:

- Concept timestamp in `Change History` shown.
- UAG timestamp in `Change History` and PLC property dialog shown.
- Customization timestamp in project property dialog shown.
- UAG timestamp in Concept / Unity Pro log file shown.
- UTC timezone shown when ever times are logged.

**Adaptable to
Customer
Standards**

UAG can be adapted to the specific needs of a project according to machinery, products, automation configuration, design tools etc.

The system administrator can define individually the naming conventions or tagging conventions for all named objects using the configurable fields:

- Area
- Process Cell
- PLC
- HMI
- ...

In addition it is possible to define valid automation configurations, for example:

- PLC families (Quantum, Premium, ...)
- PLC hardware (racks, modules, processors, ...)

For detailed information see *Customization and Project Maintenance*, p. 241.

Building an Application

The Different Roles of People to Build an Application

To make the methodology of the tool transparent, different roles are used in this document.

To build an application with Unity Application Generator requires different kinds of users or user groups. Each group is responsible for a different aspect of the application development.

The main roles are:

- The administrator
- The process engineer
- The control engineer

Note: The roles can be defined slightly different in each project according to the project organization.

Customization; Task of the Administrator

The administrator sets up UAG according to the company standards. A lot of different options can be defined before the project really starts (nevertheless it is also possible to start a project with the default settings), for example:

- Naming conventions
 - PLC standard configurations
 - Specific engineering units
 - ...
-

Process Design; Task of the Process Engineer

The process engineer is the person who defines the general decomposition of the Site into logical Units according to the process he is designing.

The elements are:

- Enterprise
- Site
- Area
- Process Cell
- Unit
- Equipment Module
- Control Module

The process engineer defines each element, its constituents and interlocks. He can also link information to the different elements which already exist, for example CAD drawings. On the other hand he does not define the control architecture.

System Architecture; Tasks of the Control Engineer	<p>The tasks of the control engineer are the following:</p> <ul style="list-style-type: none">● Definition of the topological model of the process control system.● Allocation of all inputs and outputs.● Assignment of information from the topological model to the physical model.● Set up of the communication, for example PLC-PLC. <p>By performing these tasks, the control engineer is defining in one single step the different HMIs, PLCs and the system network. UAG will generate 30 - 50% of the code for the PLCs and HMI. The rest of the logic has to be added manually by using the respective PLC and HMI programming tools.</p> <p>Examples are</p> <ul style="list-style-type: none">● Dynamical behaviour as Sequential Function Charts (SFC) in the PLC logic.● Additional graphics for pipes, vessels, etc.● Navigation between operator screens in the HMI application.● etc.
Advantages of Unity Application Generator	<p>By joining the tasks of a process engineer and a control engineer into one single software, the consistency between formerly separate tasks is guaranteed. Additionally the following tasks are also integrated within UAG:</p> <ul style="list-style-type: none">● Definition of the network.● Configure and program the PLCs.● Configure and program the HMIs.● Define the communications between PLCs and HMIs.

The Physical Model

2

Overview

Introduction

This chapter describes the general structure of the basic model of ISA S88.01 and contains detailed information about the elements and their properties. The knowledge about the Physical Model is necessary for the process engineer to make the definition of the Physical Model of the process.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
2.1	Overview of the Physical Model	27
2.2	The Elements Area, Process Cell and Unit	29
2.3	The Element Equipment Module	30
2.4	The Element Control Module and Control Module Types (Smart Control Devices)	35
2.5	Variables	47

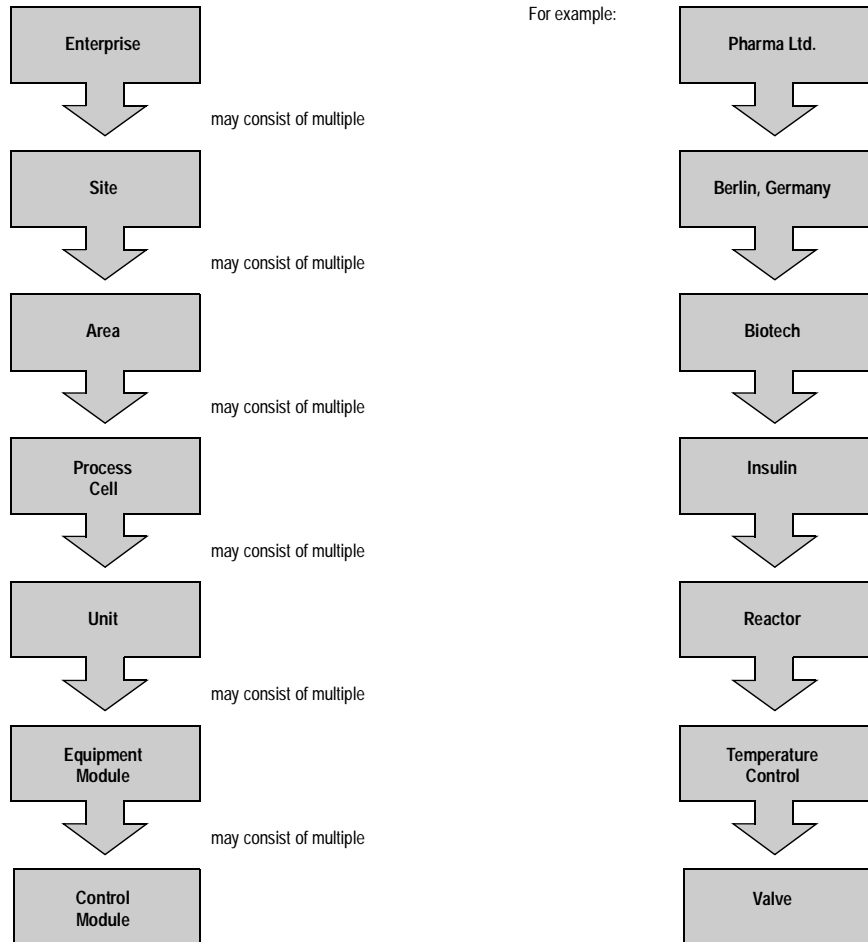
2.1 Overview of the Physical Model

Structure of the Physical Model

Structure

With the help of the Physical Model the process engineer is able to represent the process perspective within Unity Application Generator.

The Physical Model



Elements of the Physical Model

The Physical Model of UAG starts with the element Site. In this version of UAG the Site is always the top node of the S88 Physical Model.

The following elements are used to structure the process:

- Area
- Process cell
- Unit
- Equipment Module
- Control Module

The following list explains the last three elements from bottom-up.

- From the control viewpoint Control Modules represents a single entity build up of sensors, actuators and control modules. A Control Module may represent an object in the real world, e.g. a motor, a valve, a temperature transmitter, etc. or it represents a logical software object, which is used for regulatory control or other control functions, e.g. a PID loop, a timer, a counter. All higher level objects are composed of Control Modules, to form the more complex process objects, like Equipment Module.

Within UAG, Control Modules are instances of Control Module Types (also called SCoDs for Smart Control Devices). All attributes associated with a SCoD are owned by that SCoD. For further details see *The Element Control Module and Control Module Types (Smart Control Devices)*, p. 35.

- An Equipment Module is a functional group of Control Modules that combines the necessary physical processing and control modules required to carry out a finite number of processing activities for example a conveyer. Functionally the scope of the Equipment Module is defined by the finite task it is designed to carry out. An Equipment Module is typically a collection of sensors, actuators, and associated process equipment that, from the process control viewpoint is operated as a single entity.
 - In the PLC it is represented as a section.
 - In the HMI it is represented as one operator screen.
- A Unit is a group of Equipment Modules, Control Modules and other different process equipment. In that groups one or more process functions can be conducted on a batch or can be part of a batch.

Note: Some of the elements described previously are used only to structure the process, others are responsible for the generation of the code in the PLC or HMI etc. Nevertheless all elements can have attachments of different kinds of files which allow the user to document the process design in detail.

Note: In earlier versions of Unity Application Generator (One Step Generator), the Equipment Module was called "Equipment" and the Control Module was called "Device". The naming has been adjusted to the standard ISA S88.01. Control Module Types in UAG are the generic types of Control Modules.

2.2 The Elements Area, Process Cell and Unit

Area, Process Cell and Unit Description and Properties

What is an Area, Process Cell and Unit?

The elements

- Area
- Process cell
- Unit

are used to structure the process. As mentioned in the ISA S88.01 standard, this structure is determined by physical, geographical, or logical reasons.

The boundaries of these elements are usually based on organizational or business criteria as opposed to technical criteria. There are many factors other than process control that affect these boundaries.

For all these elements there is no equivalent created in the HMI.

In Concept these elements are represented as groups in the project browser.

In Unity Pro these elements are represented as functional modules in the project browser (functional view).

Properties

Area, Process Cell and Unit have the properties name. Additionally it is possible to assign a comment and document references

Properties of Areas, Process Cells and Units:

Property	Comment
Name	Unique name in the whole project Like a lot of other conventions, the name used for Area, Process Cell and Unit has to follow the naming conventions defined in the customization. The customization should be adopted before the project begins.
Comment	The comment field is a free text field. It can be used to document the process design
Documents . . .	Any number of documents of different types can be assigned to each element. The possible file types to be assigned are defined in the customization

Details of customization see *Customization and Project Maintenance*, p. 241

2.3 The Element Equipment Module

Overview

Introduction

This section describes the element Equipment Module and its properties.

What's in this Section?

This section contains the following topics:

Topic	Page
What is an Equipment Module?	31
Equipment Module Features	32
Equipment Module Properties	33

What is an Equipment Module?

What is an Equipment Module?

An Equipment Module is a functional group of Control Modules that combines the necessary physical processing and devices required to carry out a finite number of processing activities. Examples of Equipment Modules are conveyers or reactors. Functionally the scope of the Equipment Module is defined by the finite task it is designed to carry out.

An Equipment Module is typically a collection of sensors, actuators, and associated process modules that, from the process control viewpoint is operated as a single entity.

Equipment Module Features

Features

The following list describes the features of an Equipment Module:

- It performs a specific activity of the process for the conversion, transportation or storage of material or energy.
 - It is composed of Control Modules, which functionally depend on each other and are exchanging information to perform the task of the Equipment Module.
 - It does not communicate directly with the process I/O. The Control Modules are connected to the actuators and sensors.
 - It may communicate directly with the HMI that displays the process values or provides the means for the operator to interact with the process.
-

Equipment Module and PLC

- Typically, an Equipment Module and all its Control Modules are assigned to one PLC. Nevertheless it is possible to assign individual Control Modules of the same Equipment Module to different PLCs.
- Within Concept/Unity Pro an Equipment Module is represented as an FBD section. Each Control Module is represented by a Function Block (FB) within this section.

In the Concept project browser the Equipment Module section is included in the Units group.

In the Unity Pro project browser (functional view) this Equipment Module section is included in the Functional Module structure **Area** → **Process Cell** → **Unit** → **Equipment Module section**.

Equipment Module and HMI

For each Equipment Module Unity Application Generator always generates a graphical representation in the HMI. The relation between the Equipment Module and the HMI is done through the Control Domain and not directly through the HMI.

Equipment Module and Variables

After generating the PLC logic and the HMI assignments with Unity Application Generator, it is still necessary to add more logic to the Concept/Unity Pro program and to complete the HMI design. Nevertheless all variables used in the PLC and the HMI are defined within Unity Application Generator. These variables must be assigned to the objects of the Physical Model to which they belong. For this reason, the designer can safely assign additional variables to an Equipment Module.

Equipment Module Properties

Overview

There are three different kinds of properties to be defined for the Equipment Module:

- Process properties
- PLC related properties
- HMI related properties

For the initial creation of an Equipment Module, the process engineer only needs to define the process properties. The PLC and HMI related properties can be applied later by the control engineer.

General Properties

General Equipment Module properties

Property	Comment
Name	Each Equipment Module is identified by a unique name. This name has to follow the naming conventions defined by the administrator. This name is used for the name of the corresponding HMI picture.
Description	A short description of the Equipment Module can be made in the description field.
Comment...	The comment field is a free text field. It can be used to document the process design
Documents...	Any number of documents of different types can be assigned to the Equipment Module element. The possible file types to be assigned are defined in the customization

PLC Related Properties

The PLC related properties define the relation between the Physical Model in which the Equipment Module is created and the PLC Topological Model.

PLC related Equipment Module properties

Property	Comment
PLC Name	Defines the PLC the Equipment Module belongs to and where the logic is executed. The PLC has to be defined in the Topological Model before it can be assigned to the Equipment Module.
PLC Section Name	Defines the name of the section within Concept/Unity Pro. The name has to be unique, and Unity Application Generator will not allow duplicate names to be used.

HMI Related Properties

The HMI related properties define the relation between the Physical Model in which the Equipment Module is created and the Topological Model. The relation between the Equipment Module and the HMI is defined through the Control Domain and not directly through the HMI.

HMI related Equipment Module properties:

Property	Comment
Control Domain	Defines to which Control Domain the Equipment Module belongs. The Control Domain has to be defined in the Topological Model before it can be assigned to the Equipment Module. For details see <i>The HMI Group, p. 110</i>

2.4 The Element Control Module and Control Module Types (Smart Control Devices)

Overview

Introduction This chapter describes the properties of Control Modules and the general functionality of Control Module Types.

Where do I Find Further Information For detailed information on special Control Module Types refer to the specific Control Module Type library documentation.

What's in this Section? This section contains the following topics:

Topic	Page
Introduction	36
Control Module Features	37
Control Module Types or Smart Control Devices (SCoDs)	38
Free Control Module	40
Control Module Properties	41
Instruments	42
Interlocks for Control Modules	43
Links between SCoDs	46

Introduction

What is a Control Module?

From the control viewpoint Control Modules represent a single entity built up of sensors, actuators and control equipment. A Control Module may represent an object in the real world, e.g. a motor, a valve, a temperature transmitter, etc. or it represents a software object, which is used for regulatory control or other control functions, for example a PID loop, a timer, a counter.

All higher level objects are composed of Control Modules, to form the more complex process objects.

The specific behaviour of a Control Module is defined in the Control Module Type, the so called Smart Control Device (SCoD) from which the Control Module is derived, for example

- a two speed drive,
- a three position valve,
- a reversing motor,
- ...

What is a Control Module Type?

A Smart Control Device is a predefined Control Module Type. It describes one specific part of the process and comprises all functional aspects of the automation task. It takes into account all aspects of the technological object it represents.

It contains

- the PLC logic required to control a real world Control Module of a defined type,
- the operator representation in the HMI system,
- full manual control of the Control Module through the HMI,
- all variables required for the PLC and the HMI and
- a connection to the HMI alarm management

Control Module Types are organized in standard libraries which are delivered together with Unity Application Generator. Alternatively, Schneider Electric can implement specific Control Module Types for a customer and include these in customer specific libraries.

If additional project specific Control Module Types are required, please contact your Schneider Electric sales representative.

Control Module Features

Features

The following list describes the features of a Control Module:

- It represents a process object, i.e. an actuator or sensor, or a function of the process which manipulates data.
 - It belongs to one Equipment Module. Control Modules cannot be shared between Equipment Modules.
 - Either it is of a specific Control Module Type (SCoD), or it is a Free Control Module; during the development of a project also type-less Control Modules are possible as an intermediate state.
 - It can communicate with the Control Modules of other Equipment Modules.
 - It provides the (primary) entry point for the HMI for visualization and interaction with the process.
 - It communicates directly between the PLC and the HMI.
-

Control Module and PLC

- Each Control Module is represented by a Function Block within the Concept/Unity Pro section generated for the Equipment Module.
 - A Control Module is always assigned to one PLC. It cannot be split between multiple PLCs.
-

Control Module and HMI

- A Control Module is automatically assigned to the HMI as defined in the Equipment Module properties.
 - Each Control Module is represented by a graphic of its related physical object on the screen generated for the Equipment Module.
-

Control Modules and Variables

After generating the PLC logic and the HMI representation with Unity Application Generator, it is still necessary to add more logic to the Concept/Unity Pro program and to complete the HMI development. Nevertheless all variables used in the PLC and the HMI have to be defined by Physical Model they belong to. For that reason, the designer can assign additional variables to the Control Module, so called free variables.

Control Module Types or Smart Control Devices (SCoDs)

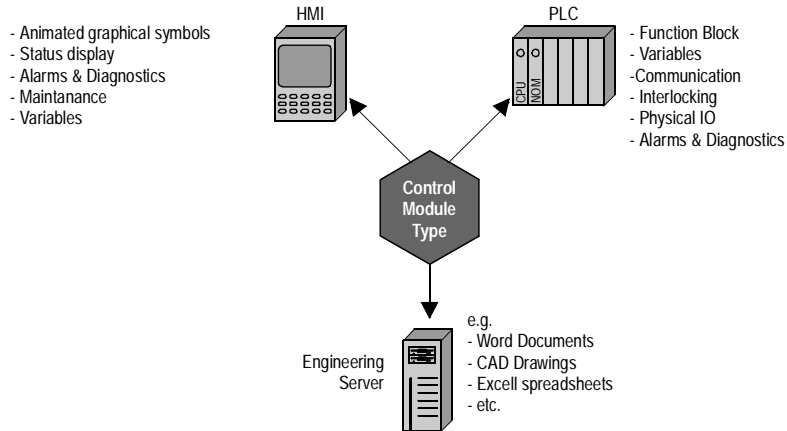
Introduction

A Control Module Type or SCoD comprises all aspects of the automation task. It is a predefined Control Module which takes into account all aspects of the technological object it represents.

What is Defined in a Control Module Type?

A Control Module Type takes into account all aspects of the technological object it represents

Illustration of a Control Module Type



For the PLC:

- The logic which controls the Control Module
- The logic which detects failures and alarms of the PLC logic
- The logic which detects process failures and generates alarms
- The communication with the HMI system
- The attributes and process variables displayed by the HMI
- The I/O connected to the Control Module
- The variables of the Control Module logic
- The commands to control the Control Module by the PLC logic (e.g. reset, start, stop)
- The link with the other Control Modules
- The ability to optimize default values
- The logic to allow the operator to maintain his process.

For the HMI:

- The graphical representation on the operator screen
- The physical units to be displayed on the operator screen
- The communication with the PLC
- The operator commands to control the Control Module, e.g. a start/stop push button, a prompt to adjust a set point etc.
- The management of alarms
- Logging of operator actions

Free Control Module

What is a Free Control Module

A Free Control Module is a Control Module which is not derived from a Control Module Type. It does not define any logic for the PLC nor any predefined functionality for the HMI. It is also possible to define free variables for a Free Control Module.

Purpose of a Free Control Module

Special Control Module Types may not be available for all process devices. Nevertheless these objects have to be part of the process design. The logic in the PLC and the graphics in the HMI will be defined manually but all variables required to perform this task have to be defined in the Free Control Module.

Note: All variables for the entire process design have to be defined within Unity Application Generator to keep the data consistent in the control application.

Free Control Modules allow the user to build the objects logically at their right position in the Physical Model.

Control Module Properties

Overview

For a Control Module only a few properties are necessary. The behaviour of the Control Module is primarily defined by the customization of Control Module properties which are different for every Control Module Type.

Properties

Control Module properties:

Property	Comment
Control Module Type	Defines the Control Module Type of the Control Module. Options depend on the selected Control Module Type library (standard library, custom library, customer specific library).
Name	Each Control Module is identified by a name unique within the Equipment Module the Control Module belongs to. This name has to follow the naming conventions defined in customization.
Description	A short description of the Control Module can be made in the description field.
Comment...	The comment field is a free text field. It can be used to document the process design.
Documents...	Any number of documents of different type can be assigned to each Control Module. The possible file types to be assigned are defined in the customization.
PLC Name	Each Control Module is assigned to one PLC. By default this field shows the PLC defined in the Equipment Module. If required, a different PLC can be selected. The Function Block of the Control Module will be processed on the selected PLC

Instruments

What is an Instrument?

An Instrument is any sensor, actor or associated processing equipment used in the process. An Instrument becomes a Control Module as soon as it is integrated in the Physical Model. Instruments are listed in the Instrument List. From there they can be moved into the Physical Model.

See also: *Working with the Instrument List*, p. 142

What is it for?

The Instrument as a preliminary stage of a Control Module has been defined in order to allow the process engineer to enter a flat list of Instruments **before** defining the complete Physical Model hierarchy.

The creation of Instruments in the Instrument List is very efficient and saves time compared to the creation of Control Modules in the Physical Model tree.

Furthermore, time can be saved by importing Instruments from the P&ID (pipework and instruments drawing).

Properties

An Instrument has the following properties:

- Name
- Description
- Control Module Type
- Comments...
- Documents...
- Properties specific to the Control Module Type
- Variables

Note: Compared to a Control Module, for an Instrument **no** PLC and **no** interlocks can be assigned.

It is possible create an Instrument without specifying the Control Module Type (**Control Module_type=Not_assigned**). A type-less Instrument can be moved to the Physical Model without restriction. It will be a type-less Control Module within the Physical Model.

Interlocks for Control Modules

Characterization For Control Modules with interlock inputs (for example motors, valves, drives, pumps, ...) interlocks can be defined in the Physical Model. A Control Module can have one or more interlock inputs; these are defined as interlock pins in the SCoDs. Unity Application Generator generates PLC code as a Function Block network for all PLC interlock definitions if they are syntactically correct. If the syntax is not valid, Unity Application Generator generates the textual description of the interlock definition as a comment in the corresponding Function Block in Concept/Unity Pro. In this case the PLC logic has to be completed by the control engineer accordingly. See also: *How to Build an Interlock Definition, p. 147*

Restriction Interlocks **cannot** be defined for Control Modules without interlock inputs. Interlock inputs (interlock pins) are defined in the SCoD.

Interlock Definition An interlock definition consists of one or more interlock conditions and is assigned to a specific interlock input. An interlock definition can be stated for each interlock input of a Control Module.

An interlock definition is composed of

- interlock conditions and
- logical operators for the combination of the interlock conditions (AND, OR, XOR, NOT).
- opening and closing parenthesis

Example interlock definition:

```
((Boiler1_Motor1_FTR = 1) And (Boiler1_Valve1_FT >= 100)) Or  
(Boiler1_AIn1_AHH = 1)
```

Interlock Condition

An interlock condition is a logical comparison between variable values (from any other Control Module or Equipment Module) or between a variable value and a constant. The result of such a condition is of type Boolean.

Example:

Boiler1_Motor1_FTR = 1

Meaning: If *Failed To Run* output of Control Module *Motor1* of Equipment Module *Boiler* is ON

The interlock condition is composed of the following elements:

Components of the interlock condition	Description
Interlock variable	The interlock variable is a variable that will be compared with another variable or literal. It can be any variable defined in a Control Module or Equipment Module.
Operator	The comparison operator for the condition (=,<>,<,>,<=,>=).
Condition variable / literal	Either a variable or a literal, with which the interlock variable is compared. If it is a variable: It can be any variable defined in a Control Module or Equipment Module belonging to the same PLC. To interlock with variables from other PLCs these variables have to be communicated via a Channel in advance, see <i>PLC Channels</i> , p. 90.

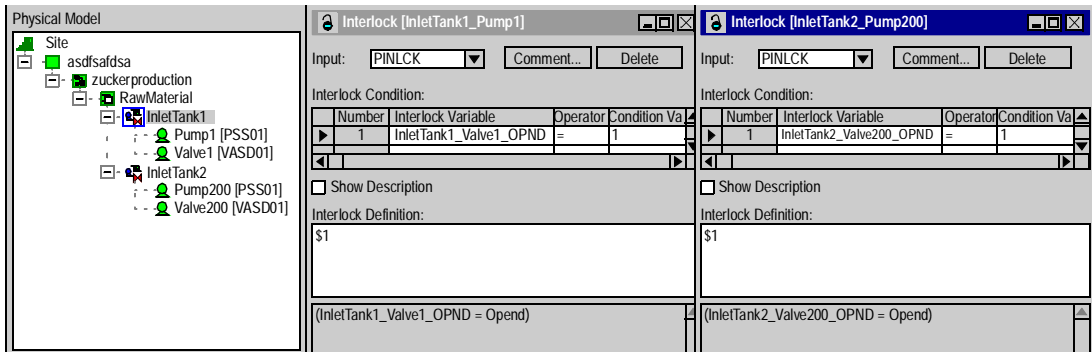
Furthermore the following information belongs to each interlock condition:

Components of the interlock condition	Description
Number	A sequential number defining the number of the condition. This number is necessary for referencing the condition when combining the conditions to a complete interlock definition. The user cannot change this number.
Description	Free text for the description of the condition.

Copy of Interlocks

The interlock definition is part of the Control Module definition. A Control Module belongs typically to an Equipment Module. An Equipment Module implementation can be the inlet part of a Tank, consists of a valve and a pump. The pump has, for example, one interlock. This interlock describes that the pump must not run if the interlock is set. This behavior is general for all inlets part of any tank in the plant. When the user copies an Equipment Module the interlock will be pasted into the new Equipment Module. The variables used in the interlock condition will be replaced with the new Control Module variables.

The following figure shows an example for copied interlocks.



The Equipment Module `InletletTank1` was copied as `InletTank2`.
The Control Module variable `InletTank1_Valve1_OPND` is replaced by the new Control Module variable `InletTank2_Valve200_OPND`.

Links between SCoDs

Introduction

Input or Input / Output pins of SCoDs can be linked with other pins of SCoDs or with free variables.

The context menu of a control module offers a menu item `Open Links`. If this dialog is opened, a table appears that contains all input and in / out pins of the parent control module. To link a variable with a pin of this control module, a variable is moved with drag-and-drop from the variables table to the link table.

A link variable can be used several times in the link table of the same or different control modules, therefore multiple connections are possible.

The following restrictions have to be considered:

- The datatypes of the input pin and the link variable must be the same.
- Variables of datatype `HMI` cannot be linked (they do not exist in the PLC section).
- Variables can only be linked to input pins or in / out pins.
- Variables can only be linked to pins with the `Pin Usage Not Connected`.

Additionally a link variable can be moved in the link table of a control module, also with drag-and-drop.

If a link has been created in UAG, it is generated in Concept / Unity Pro during generation of the PLC program.

If the link variable is a pin of a SCoD that exists in the same section as the input pin, a graphical link is created. Otherwise the pin is connected with the variable.

2.5 Variables

Overview

Introduction

This section describes the Variables.

What's in this Section?

This section contains the following topics:

Topic	Page
Introduction	48
Connection Types and Data Types of Variables	49
General Variable Properties	51
HMI Related Variable Properties	53

Introduction

Significance of Variables

In usual tools for PLC or HMI programming, variables are used everywhere. The disadvantage is, that these variables have to be defined in each system separately. So if a variable is needed in both systems, the variable has to be defined twice. The manual synchronization of variables is a major reason for bugs in a control application. With Unity Application Generator all variables are defined only once and are automatically generated in all of the systems, in which they are needed.

To Which Objects are Variables Assigned?

In general variables are related to a Control Module Type. They are automatically generated for each instance of a Control Module Type. These variables are called Control Module Type variables.

In addition variables can be added to

- Equipment Modules and
- Control Modules.

These variables are called free variables.

Note: Type-less Control Modules have no variables.

Naming Conventions

Variables have to be named according to the naming convention defined during customization. The name must be unique only inside the element it is attached to. The name in the PLC or in the HMI is built by the concatenation of the variable name and the object name it is assigned to.

Example:

Variable name	Failed
Variable belongs to Control Module	Motor1
Control Module belongs to Equipment Module	EQ411A
Resulting variable name in Concept, Unity Pro and HMI	EQ411A_Motor1_Failed

Connection Types and Data Types of Variables

Connection Types

If a variable is specified in Concept, Unity Pro or in an HMI system, it is obvious to which system it belongs. If a variable is defined in Unity Application Generator, it is necessary to define to which part of the control system it belongs. The required connection type depends on where the variable is used.

Simple connection types of variables

Connection Type	Comment
PLC	Used PLC internal only
IO_PLC	Variable is used within the PLC logic and connected to a physical I/O point
HMI	Used HMI internal only

Very often variables are not only used in one part of the control system but in multiple parts, e.g. the PLC and the HMI. In the classical approach to build a control system this means that the variables have to be specified twice. With Unity Application Generator this is done only once by the combination of two connection types. Variables of these connection types are communicated between the different parts of the control system.

Combined connection types of variables

Connection Type	Comment
PLC_HMI	Variable is communicated between the PLC and the HMI
PLC_NET	Variable is communicated between the PLC and a network node, e.g. hand-held panel

Variable Data Types

For each variable it is necessary to define which data type it is. The available data types depends on the category of the variable. In general all data types of the destination system are available. Unity Application Generator also supports the structured data types of Concept and Unity Pro, but not the individual elements of the structure.

Handling of Initial Values in Concept

An initial value can be assigned to each variable (except structured variables). Even though Concept do not allow the user to specify initial values for variables of type BOOL which are located in the 0x address range of the PLC, it is possible to assign an initial value to these variables using Unity Application Generator. Unity Application Generator automatically generates one or more initialization sections which avoid this restriction.

**Handling of
Initial Values in
Unity Pro**

An initial value can be assigned to each variable (except IODDT variables and structured variables). No restrictions exist for initial values for located variables of type BOOL.

**Free properties
for Variables &
Control Modules**

Variables and Control Modules have several properties that can be defined in SCoD editor and used in UAG application. The basic property types are e.g. Command, Initial Value, Alarm, Measurement Unit etc. In specific cases users would like to describe the Variables and Control modules with some other properties which are not predefined in UAG which could be seen in the whole system as regular property with inheritance possibility. The most important feature of the free properties is concerning Generic HMI generation. The properties will be included in the XML and CSV files and in this way accessible for the user to be imported into a specific HMI system.

The free properties will be defined in SCoD editor as normal ones. The number of possible free properties will be fixed to specific number e.g. 20. The user will always see the properties as strings. No data type control will be done on UAG side.

General Variable Properties

Control Module Type Variables

Variables coming from a Control Module Type have most properties already predefined. The designer can only change the required fields.

Free Variables

Free variables defined by the process designer have no predefined properties and these must be specified in detail. What properties are available depends on the categories of the variable. HMI related variables have several additional properties which define their behaviour in the HMI system. If you are using Monitor Pro or iFIX, a free variable can be represented by a graphical symbol (option in basic properties).

Basic Properties

Properties of a variable

Property	Description	Options	Comment
Name	Unique name of variable		The Control Module or Equipment Module name has to be unique. The name has to follow the naming conventions defined during customization.
Variable Used (line is marked with a check)	Defines if a variable of connection type IO_PLC is assigned to a physical IO, if it is used in the communication table to a Net Partner, or if it is not used	<ul style="list-style-type: none"> ● Yes ● No 	A variable can be set to not used, e.g. because a real world object valve has no feedback limit switch but the Control Module Type offers one. Not used variables will not be generated for Concept/Unity Pro.
Description	Description of the variable		Free text field. This text is by default a part of the alarm text.
Variable Inverse	Inverts the variable (only for IO_PLC variables)		
Connection Type	Defines the connection type of the variables	<ul style="list-style-type: none"> ● PLC ● HMI ● IO_PLC ● PLC_HMI ● PLC_NET 	See <i>Connection Types and Data Types of Variables</i> , p. 49.

Property	Description	Options	Comment
Data Type	Data type of variable	<ul style="list-style-type: none"> ● ANL_IN ● ANL_OUT ● BOOL ● BYTE ● DINT ● INT ● REAL ● TIME ● UDINT ● UINT ● WORD ● Structured data types of Concept/Unity Pro 	Available options depend on connection type.
In/Out	Defines the usage of the variable	<ul style="list-style-type: none"> ● Input ● Output ● In/Out 	The variable is connected to an input and an output if the Function Block changes the value of the variable.
Initial Value	Initial value of variable		Depending on the datatype of the variable there may be additional checks related to boundaries and scaling.
Timeout state	Defines failure (timeout) action of an output I/O variable	<ul style="list-style-type: none"> ● Not_Assigned ● Disabled ● Last value ● User defined 	<p>If an error occurs in a PLC output module, the user can define what should happen to the output signals:</p> <ul style="list-style-type: none"> ● Disable: the signal is set to 0. ● Last value: the signal is holding the last value before it failed. ● User defined: the user can define the state of the signal if the hardware module fails.
Timeout value	Defines the value of the variable if a failure (timeout) occurs		Only available if <code>timeout state</code> is set to "user defined".

HMI Related Variable Properties

Overview

HMI related properties are only available for variables of category HMI or PLC_HMI. These properties are divided into the following categories, depending on which aspect they are related to:

- Alarm related properties
 - Command related properties
 - Display related properties
-

Alarm Related Properties

These properties are always available independent of the variable's data type. Up to 8 alarms can be defined for a variable, except for Boolean variables, which can only have one.

Alarm related properties:

Property	Description	Options	Comment
Alarm	Defines if the variable is an alarm	<ul style="list-style-type: none"> ● Yes ● No 	
Alarm Limit	To set the active value for the alarm	Any value of the same data type as the variable (0 or 1 for Boolean variables).	Value to be compared with the variable value.
Alarm text	Message to the operator if alarm occurs	String	Free text field. If no text is specified, the text will be built automatically. The default is generated by concatenating the description of the variable, the description of the device and the description of the equipment.
Alarm Operator	Comparison operator for comparing the variable value against the respective alarm limit.	=, <>, >, <, >=, <=	Not available for Boolean variables (always „=").
Alarm Priority	Defines the alarm's priority	Select from customized list.	The alarm priorities are defined during customization.
Alarm Group	Defines grouping of alarms.	Select from customized list.	The alarm groups are defined during customization.

Command Related Properties

These properties are always available independent of the variable's data type.
Command related properties

Property	Description	Options	Comment
Command	Command Type of the variable	Select from list <ul style="list-style-type: none"> ● View only ● Operator ● Logic ● Constant ● Parameter ● Not assigned 	For device type variables, the command type is predefined: <ul style="list-style-type: none"> ● Variables connected to an input are always Operator or Parameter. ● Variables connected to an output are always View only. ● Variables connected to In/Out could be everything except View only.
Access level	Access level for operator	Select from list.	The access levels are defined during customization.
State 0 text	Text to be displayed in HMI if variable is 0	<ul style="list-style-type: none"> ● String of max 9 characters ● Not assigned 	Only for boolean PLC_HMI variables
State 1 text	Text to be displayed in HMI if variable is 1	<ul style="list-style-type: none"> ● String of max 9 characters ● Not assigned 	Only for boolean PLC_HMI variables

Meaning of the different commands:

Option	Change by operator?	Comment
View only	No	The value of the variable is displayed on the HMI but the operator cannot change it.
Logic	No	The value of the variable is displayed in the HMI but the operator cannot change it (the generated code will be same as for View only, but setting this attribute tells the control engineer to add additional logic).
Operator	Yes	The value of the variable is displayed in the HMI and the operator can change it.
Parameter	Yes	The value of the variable is displayed in the HMI and the operator can change the initial value of the variable. Data transfer between the PLC and HMI is slower than for the option Operator
Constant	No	Depending on the variables type, the methodology to set the value is different: <ul style="list-style-type: none"> ● If it is a DeviceType Variable, the input of the FB of the device will be connected with the value specified in initial value. ● If it is a Free Variable, the initial value will be set in Concept/Unity Pro.

Display Related Properties

These properties are only available for analog values.
Display related properties

Property	Description	Options	Comment
Display format	Format for HMI system to display analog values	<ul style="list-style-type: none"> ● 999.9 ● 99.99 ● 9.999... 	Analog values only.
Scaling min. Scaling max.	The PLC system needs a scale for analog values. Consist of 2 variables in the PLC for the minimum and maximum value		Available to HMI for display
Boundary min. Boundary max.	The boundaries for the operator consist of a minimum and a maximum value.	Depend on the analog value	Attribute used by HMI to check process limits
Unit Group	Group of physical units	<ul style="list-style-type: none"> ● Select from list ● Not assigned 	Analog variables only. The groups are defined during customization.
Unit	Physical unit of variable	<ul style="list-style-type: none"> ● Select from list ● Not assigned 	Analog variables only. The units are defined during customization.

The Topological Model

3

Overview

Introduction

The Topological Model describes the topology of the control system. The Topological Model is used by the control engineer to define the architecture of the control system. Here you will learn about the general topological layout which is supported by Unity Application Generator. In addition you will understand which elements are used within UAG to define this layout.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
3.1	Control System Topology and Topological Model	61
3.2	The Groups Network Segments and Routing Paths	78
3.3	The PLC Group	84
3.4	The HMI Group	110
3.5	The Data Server Group	114
3.6	The Network Nodes Group	117

3.1 Control System Topology and Topological Model

Overview

Introduction

In this section you will learn how a typical control system topology is represented in the Topological Model.

In addition you get the information which elements are supported by Unity Application Generator.

What's in this Section?

This section contains the following topics:

Topic	Page
The Topology of a Control System	62
Structure of the Topological Model	67
Communication via Modbus Plus and Ethernet	69
Additional Information Concerning Ethernet	71
Quantum Hot Standby Configuration (HSBY)	74

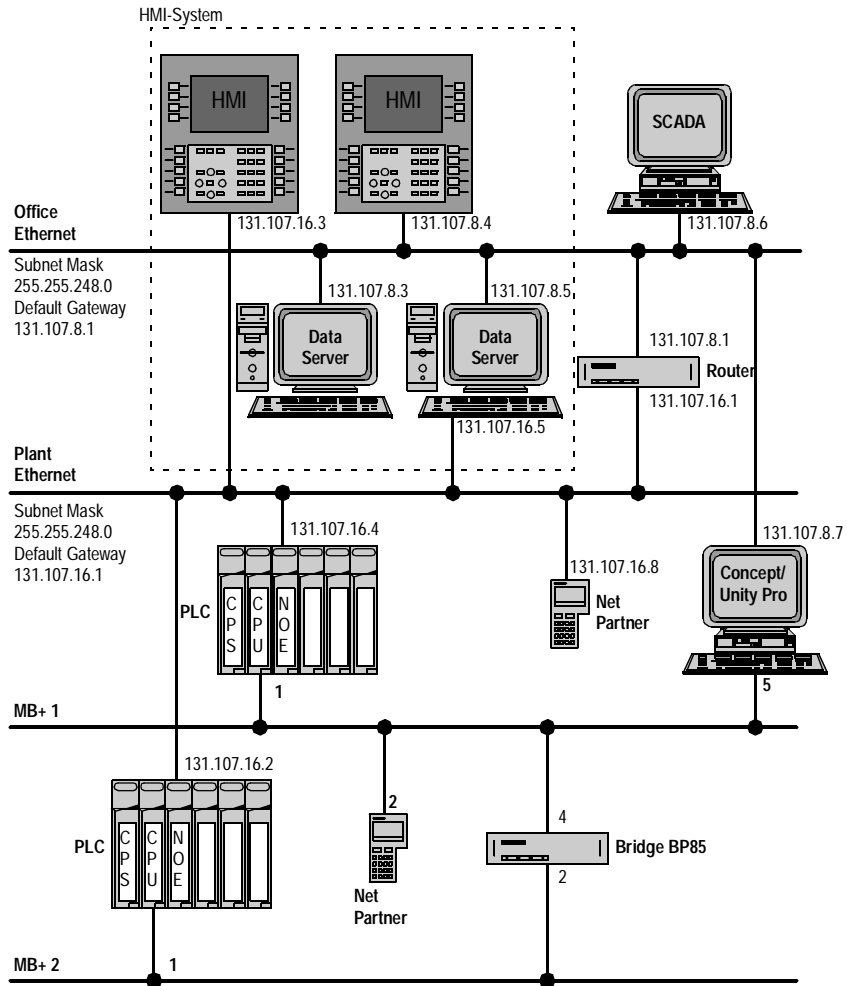
The Topology of a Control System

Introduction

Unity Application Generator flexibly supports complex control system topologies. These topologies are built with elements typical for all control systems. The figure shows a control system topology and gives an overview of the elements supported by UAG. These elements and their relationships are described in the Topological Model of UAG.

**Example of a
Typical Control
System**

Typical Control System



Supported Elements

The following elements are defined within the Topological Model of UAG

- Concept/Quantum PLCs including:
 - Remote and Distributed I/O
 - hot standby systems
 - Momentum Modbus Plus I/O
 - Momentum Ethernet I/O
 - user defined HW Modules (generic modules and ModConnect partner modules)
- Concept/Momentum PLCs including:
 - Momentum I/O Bus
 - Momentum Modbus Plus I/O
 - Momentum Ethernet I/O
 - user defined HW Modules (generic modules and ModConnect partner modules)
- Unity/Quantum PLCs including:
 - Remote and Distributed I/O
 - Momentum Modbus Plus I/O
 - Momentum Ethernet I/O
 - user defined HW Modules (generic modules)
- Unity/Premium PLCs including:
 - Momentum Modbus Plus I/O
 - Momentum Ethernet I/O
 - XBus I/O
 - user defined HW Modules (generic modules)
- Data Servers (Monitor Pro, iFIX and others)
- HMIs (Monitor Pro, iFIX and others)
- Network Nodes (Net Partners and Other Nodes)
- Routers
- Bridges
- Network Segments of type Modbus or Ethernet

The following elements are defined as Other Nodes within the Topological Model

- The engineering workstations shown in the figure (Concept/Unity Pro, Monitor Pro, iFIX) can be created as Other Network Nodes.
 - The HMI devices and their addresses
-

The Network

UAG supports Modbus Plus and Ethernet networks.

The communication capabilities of the network types are different

- For Monitor Pro and iFIX, the Data Servers can be connected to Ethernet or Modbus Plus (iFIX only)

For a generic HMI, the Data Server can be connected via Modbus Plus or Ethernet

- Network Nodes can be connected via Modbus Plus or Ethernet
- PLCs can be connected via Modbus Plus or Ethernet
- Modbus Plus Bridges can be configured.

Ethernet Routers are not configured explicitly in UAG, but it is assumed that an existing default Gateway in the network segment definition represents a router.

Note: Quantum

All modules can be used for communication between PLCs and to Data Servers or to Net Partners. But only the listed CPU/NOE modules can be used in the sending PLC for an PLC <-> PLC communication via Ethernet.

- 140-NOE-771-00
- 140-NOE-771-01
- 140-NOE-771-11

The other NOE modules can only be used in the receiving PLC.

Note: Momentum

All Ethernet processor modules can be used for communication between PLCs and to Data Servers or to Net Partners without restrictions.

Note: Premium

All modules can be used for communication between PLCs and to Data Servers or to Net Partners. But only the listed Ethernet modules can be used in the sending PLC for an PLC <-> PLC communication via Ethernet.

- TSX ETY4103
- TSX ETY5103
- TSX P57 5634M
- TSX WMY 100

The other Ethernet modules can only be used in the receiving PLC.

**Redundancy
with UAG**

UAG supports to set-up redundancy in several parts of the control topology:

- Multiple Network Segments can be defined.
- For communication between Quantum and Momentum PLCs redundant network paths can be defined.
- Concept / Quantum PLCs can be defined as redundant using the 140 CHS 110 00 see *Quantum Hot Standby Configuration (HSBY)*, p. 74
- Unity Pro / Quantum PLCs can be defined as redundant using the 140 CPU 671 60 see *Quantum Hot Standby Configuration (HSBY)*, p. 74
- Data servers with redundant network card
- Data Servers can be set up redundant; nevertheless, this is not defined within UAG, it is done e. g. using Monitor Pro or iFIX functionality. For more details please refer to the iFix and Monitor Pro documentation.
- For communication between Quantum, Premium and Momentum PLCs and Data Servers redundant network paths can be defined.

**Restrictions
Fipio and
CANOpen**

The configuration of Fipio and CANOpen is not supported within Unity Application Generator.

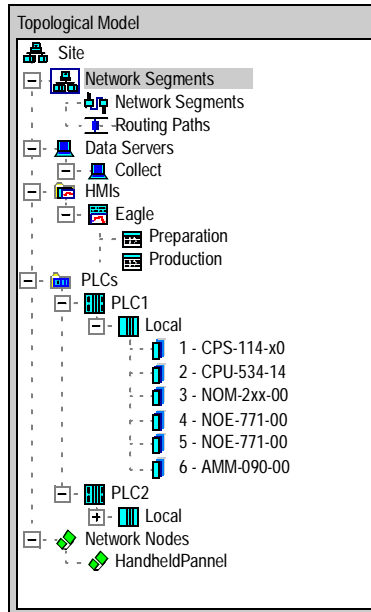
Structure of the Topological Model

Introduction

Within the Topological Model the control engineer defines the architecture of the control system. Doing this, the control engineer defines resources which are used by the elements of the Physical Model.

How does it look like?

The Topological Model with some user defined elements.



Overview of the Elements

The Topological Model contains the following groups:

- The PLC group
- The HMI group
- The Data Server group
- The Network Nodes group
- The network group

PLC Group	<p>In the PLC group the PLCs are defined.</p> <ul style="list-style-type: none">● For each PLC the complete configuration with racks is defined.● For each rack the complete configuration with modules is defined.● For communication modules their network addresses are defined● For each PLC the communication channels are defined. These channels define which data are exchanged between which objects of the control system.● Routing paths, which defines connections between different Network Segments via bridges or routers.
HMI Group	<p>In the HMI group multiple HMI applications are defined.</p> <ul style="list-style-type: none">● For each HMI multiple Control Domains can be defined.
Data Server Group	<p>In the Data Server group, the Data Servers and their connection to the network (network type and address) are defined.</p>
Network Nodes Group	<p>In the Network Nodes, the Net Partners and/or Other Nodes, their connection to the network (network type and address) and their symbol in the Topological Viewer are defined.</p>
Miscellaneous	<p>In the network group the complete communication architecture is defined. The elements of the network group are the</p> <ul style="list-style-type: none">● Network segments, which defines names and types of Network Segments.● For each I/O module the I/O points are defined

Communication via Modbus Plus and Ethernet

What are Network Segments?

A Network Segment is a single network line of a defined type (e.g. Modbus Plus) with the full range of addresses available. Network nodes like PLCs, HMIs, Data Servers, Network Partners and Other Nodes are attached to one or more Network Segment(s).

What Kind of Segments are Supported?

Unity Application Generator supports networks of type

- Modbus Plus
- Ethernet

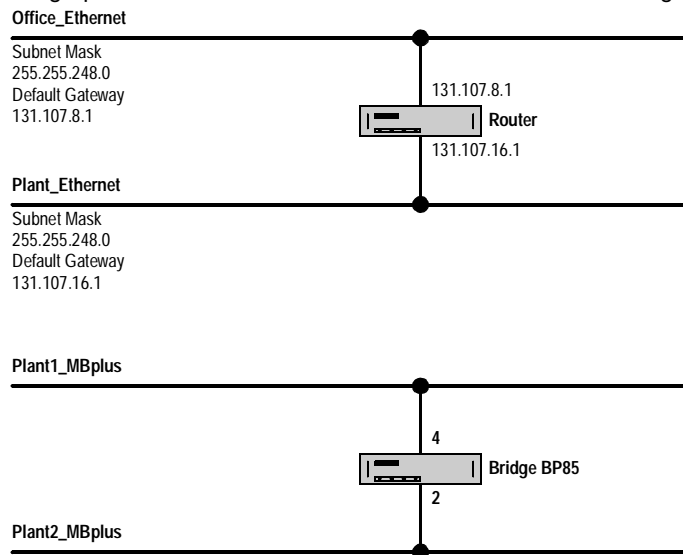
Routers and Bridges

In general, routers and bridges connect different Network Segments or different kinds of networks like Modbus Plus and Ethernet.

Within UAG bridges can be defined between Modbus Plus Segments.

Unity Application Generator does not distinguish between a router or a bridge. From UAG point of view both perform the same task; they connect different Network Segments.

The graphic shows an Ethernet router and a Modbus Plus bridge



Defining a Router between Ethernet Segments

In a Windows environments, the address of a router is defined as the default gateway address.
In UAG this is similar. For each Ethernet segment, a default gateway address can be defined together with the subnet mask in the segment properties.

Defining a Bridge between Modbus Segments

Modbus Plus bridges are defined separately from the Network Segments in the *Routing Paths*. The Routing Path simply defines the address of the bridge from the segments point of view. For details see *Routing Paths Description and Properties, p. 82*

What is a Channel?

A Channel describes

- to which objects of the control system the PLC communicates,
- via which communication path the communication takes place,
- an alternative, on which communication path the communication takes place and
- which variables have to be exchanged.

The communication is always done through a PLC. This means, one communication partner in a Channel is always a PLC.

How to Define the Communication

The definition of the network between the different parts of the control system is done with multiple objects of UAG.
One major element in the network design is to define which data are exchanged

- between which objects of the control system (communication partners),
- via which communication paths.

This is done mainly through the Channel definition of the PLC.
The Channel definition is based on information about the network design, defined through the network elements of the Topological Model. These elements have to be defined first, before the Channel definition can be performed.
These elements are

- the network addresses of the communication modules of the PLC, see *Module Properties, p. 105*,
- the Network Segment, see *Network Segment Description and Properties, p. 79*,
- the Routing Paths, see *Routing Paths Description and Properties, p. 82*,
- the network addresses of the Data Servers, see *Data Server - Description and Properties, p. 114*, and Net Partners, see *Net Partner Properties, p. 119*.

Additional Information Concerning Ethernet

Introduction

Additional Information are:

- Specific information concerning Ethernet for Unity Application Generator.
- No general information concerning Ethernet. Nevertheless, it is required to know how to configure an Ethernet network to set up the network definitions properly within UAG.

Requirements for Ethernet segment

An Ethernet Network Segment is specified by

- Subnet Mask
The subnet mask defines which part of the IP address belongs to the network and which to nodes
 - Default Gateway
The default gateway defines the routing address to other network segments.
- All nodes on an Ethernet segment have the same subnet mask and the same gateway.

Addressing rules checked by UAG

The following conditions are checked:

- The Network Segment name must be unique.
- The subnet mask and gateway have the format of an IP-address (4 bytes)
- Different Ethernet segments must not have the same default gateway address, but the subnet mask may be equal

Note: UAG does not check if a node address is a valid address concerning the subnet mask!

Local Ethernet Network Segments

In addition to the segments, which belong to the plant network or even the Internet, local Network Segments can be defined. These networks are not connected to the plant network (via routers), but are only connected locally to a Ethernet module (NOE, M1 Ethernet, ETY, etc.).

The local networks are defined with an empty default gateway (not connection to the other networks). In this case the IP addresses need not to be unique within all segments but only on this local segment.

The IP standard recommends using the following address ranges for private (local) networks:

- 10.0.0.0 to 10.255.255.254: Class A
- 172.16.0.0 to 172.31.255.254: Class B
- 192.168.0.0 to 192.168.255.254: Class C

Nevertheless, Unity Application Generator does not enforce the use of these addresses for local Network Segments.

Ethernet Nodes

For a node connected to the Ethernet, only the IP address has to be defined. UAG will check the following rules

- Each node must be assigned to a network segment
 - Each node has a TCP/IP address (4 byte N1.N2.N3.N4)
Ranges of the the addresses are:
 - N1 between 1-126 or 128 - 223
 - N2 between 0-254
 - N3 between 0-254
 - N4 between 1-254
 - The address 1.1.1.1 is not allowed
 - The TCP/IP address of the nodes on all segments must be unique, except local networks, see above.
-

**Supported
Quantum
Ethernet
Modules**

The following Quantum Ethernet communication modules are supported:

- 140-NOE-771-00
- 140-NOE-771-01
- 140-NOE-771-10
- 140-NOE-771-11

Note:

All modules can be used for communication between PLCs and to Data Servers or to Network Nodes. But only the listed CPU/NOE modules can be used in the sending PLC for an PLC <-> PLC communication via Ethernet.

- 140-NOE-771-00
- 140-NOE-771-01
- 140-NOE-771-11

The other NOE modules can only be used in the receiving PLC.

**Supported
Momentum
Ethernet
processor
adapters**

The following M1 Ethernet processor adapters are supported:

- 171-CCC-960-30-IEC
- 171-CCC-960-91
- 171-CCC-980-30-IEC
- 171-CCC-980-91

All modules can be used for communication between PLCs and to Data Servers or to Network Nodes.

**Supported
Premium
Ethernet
Modules**

The following Premium Ethernet communication modules are supported:

- TSX ETY4103
- TSX ETY5103
- TSX P57 5634M
- TSX WMY 100

Note:

All modules can be used for communication between PLCs and to Data Servers or to Network Nodes. But only the listed Ethernet modules can be used in the sending PLC for an PLC <-> PLC communication via Ethernet.

- TSX ETY4103
- TSX ETY5103
- TSX P57 5634M
- TSX WMY 100

The other Ethernet modules can only be used in the receiving PLC.

Quantum Hot Standby Configuration (HSBY)

What is a Hot Standby System

The Quantum Hot Standby system is designed for use where downtime cannot be tolerated. The system delivers high availability through redundancy.

Two backplanes are configured with identical hardware and software.

One of the PLCs acts as the Primary controller. It runs the application by scanning user logic and operating remote I/O.

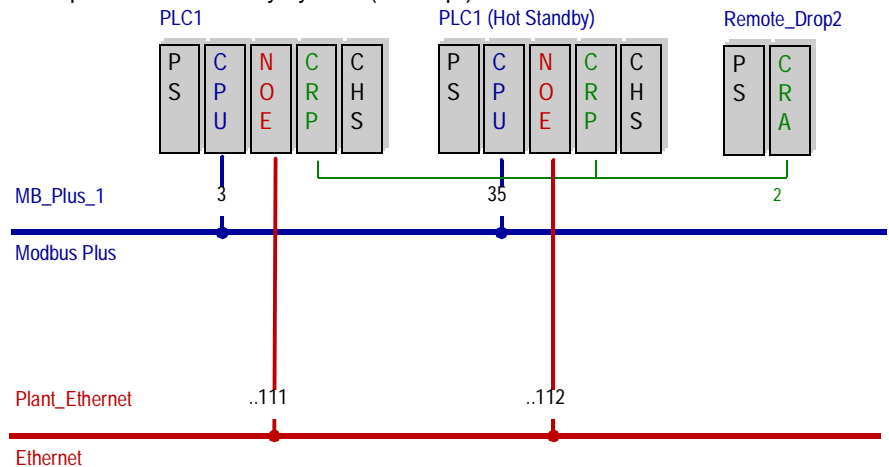
The other PLC acts as the Standby controller. The Primary controller updates the Standby controller after each scan. The Standby is ready to assume control within one scan if the Primary fails.

Primary and Standby states are switchable. Either controller can be put into the Primary state, but to do this, the other must be in the Standby state. The remote I/O network is always operated by the Primary controller.

Note: A Quantum Hot Standby system supports only remote I/O. It does not support local I/O or distributed I/O (DIO). Except in Unity Pro Hot Standby. In Unity Pro it is allowed to use local I/Os. For data integrity it is not advisable to use local I/O within a Hot Standby controller.

For details about Quantum Hot Standby, refer to the Quantum Hot Standby Planning and Installation Guide.

Example of a hot standby system (Concept):



How does it Work?	The Quantum Hot Standby system is designed for use where downtime cannot be tolerated. The system delivers high availability through redundancy. Two backplanes are configured with identical hardware and software. Each controller is paired with a 140 CHS 110 00 Hot Standby module (Concept). Use the 140 CPU 671 60 controller for Hot Standby with Unity Pro without the 140 CHS 110 00 Hot Standby module. It is included within the controller. The module monitors its own controller and communicates with the other hot standby module. The system monitors itself continuously.
Primary and Standby Controller	One of the PLCs acts as the primary controller. It runs the application by scanning the logic and operating remote I/O. The other PLC acts as the standby controller. The primary controller updates the standby controller after each scan. The standby is ready to assume control within one scan if the primary fails.
Hot Standby and Unity Application Generator	Within Unity Application Generator a hot standby system is defined at the PLC group. As soon as a 140 CHS 110 00 module is placed into a local rack of a Concept PLC, this PLC automatically is treated as a hot standby system. The same with the 140 CPU 671 60 module within a Unity Pro PLC. For more information on placing modules refer to <i>The PLC Group, p. 84</i> .
Limitations	General restrictions of a Hot Standby system: <ul style="list-style-type: none">● A Quantum Hot Standby system supports only remote I/O. It does not support local I/O or distributed I/O. Except in Unity Pro Hot Standby. In Unity Pro it is allowed to use local I/Os. For data integrity it is not advisable to use local I/O within a Hot Standby controller.● Concept Only PLCs of type 140 CPU 534 14(A) and 140 CPU 434 12(A) are supported.● Unity Pro Only the controller 140 CPU 671 60 is supported for Hot Standby.

Note: As soon as a 140 CHS 110 00 (Concept) or 140 CPU 671 60 (Unity Pro) module is placed into a PLC, Unity Application Generator will enforce these limitations. This implies that as soon as a module is placed into a PLC which is beyond these limitations, it is not longer possible to insert a 140 CHS 110 00 module into this PLC.

Minimal Configuration of a Hot Standby PLC

The minimal configuration for Concept consists of the following modules:

- Backplane
- Power supply
- Remote I/O Head Process or module
- 140 CHS 110 00 Hot Standby module
- At least one RIO drop

The minimal configuration for Unity Pro consists of the following modules:

- Backplane
 - Power supply
 - Unity Pro Hot Standby CPU (140 CPU 671 60)
 - At least one RIO drop
-

Hot Standby and Communication

Even though a hot standby system consists of two PLCs, from the communication point of view it works like a single PLC. Nevertheless both PLCs are connected to the network. To enable this, a special network address is automatically assigned to the standby PLC. These addresses cannot be used for other network participants.

Automatically assigned address for Modbus Plus

- Primary PLC address = n
- Standby PLC address = $n+32$
If $n+32 > 64$, then standby PLC address = $n - 32$

Automatically assigned address for Ethernet

- Primary PLC address = $x.y.z.n$
 - Standby PLC address = $x.y.z.n+1$
If $n+1 = 255$, then standby PLC address = $x.y.z.1$
-

Allocation of Registers


To run a Quantum HSBY system, at least five 4x registers have to be allocated (Concept). In Unity Pro the system word %SW60 will be used as the command register, %SW62 and %SW63 are used as transfer registers. This is done automatically by Unity Application Generator. UAG will make no further use of these registers and will not manipulate the content of these registers. The following list splits-up the registers within Concept.

- The first register is the command register
- The next 4 registers are part of the so called non transfer area of the state RAM
 - The first 2 registers are the "reverse transfer" registers
 - The third register is the status register
 - The fourth has no special meaning

Unity Application Generator does not allocate any further registers for the non transfer area.

The Hot Standby Command Register

This register will allow the user to manipulate the various modes of HSBY operation during runtime. This may either be done through programmed logic or just by access through the reference data editor (RDE) of Concept or the animation table in Unity Pro.

	CAUTION
	Risk of unintended behavior! Don't use the command register for any other purpose throughout the programmed application, otherwise the system will end up with unintended behavior. Failure to follow this precaution can result in injury or equipment damage.

The Reverse Transfer Registers

The reverse transfer registers can be used to transmit diagnostic data from the standby controller to the primary controller. They are copied from the standby to the primary controller.

The Status Register

This register provides information to the user logic about the current situation of the hot standby system. Use this register to monitor the current machine status.

3.2 The Groups Network Segments and Routing Paths

Overview

Introduction

The network group contains the general layout for the network.

In the network group are defined:

- The Network Segments
- The Routing Paths, and implicitly the routers

An overview of a typical control system and its network you will find in *Example of a Typical Control System, p. 63*.

What's in this Section?

This section contains the following topics:

Topic	Page
Network Segment Description and Properties	79
List of Network Nodes	80
Routing Paths Description and Properties	82

Network Segment Description and Properties

What are Network Segments?

A Network Segment is a single network line of a defined type (Modbus Plus or Ethernet) with the full range of addresses available. Network nodes like PLCs, HMIs, Data Servers, Net Partners and Other Nodes are attached to one or more Network Segment(s).

How are Network Nodes Associated to a Segment?

The definition which hardware components are attached to a segment is not done under the Network Segment node. This kind of definition is always done as a property of the respective object group.

How are Multiple Segments Connected?

Network Segments are connected via bridges. The definition of bridges depends on the network type.

- For Ethernet, the bridges are defined by assigning the default gateway
- For Modbus Plus, the bridges are defined under the Routing Paths group.

Properties

Segment properties:

Property	Comment
Name	Unique name of segment The name has to follow the naming conventions defined during customization.
Type	Defines the network type Select from list.

Additional properties for Ethernet segments:

Property	Comment
Subnet Mask	The subnet mask defines which part of the IP address belongs to the network and which to the nodes.
Default Gateway	The default gateway defines the routing address to other Network Segments.

List of Network Nodes

Overview

For a better overview of the network partners, a table will display all network nodes connected to a selected network segment. It allows additionally a simple change of multiple network addresses.

Both network types, Ethernet or Modbus Plus, can be displayed.

How to get the Network Nodes

The following table shows how to get the information of the `Network Nodes`.

Step	Action
1	Open the explorer View → Explorer
2	Select the Network Segments with in the topological view and expand the <code>Network Segments tree</code> .
3	Select a <code>Network Segment</code> .
4	Open the context sensitive menu by clicking the right mouse button on the selected segment and <code>Open Network Nodes</code> .

Table of Network Nodes

The table of network nodes implies the following columns:

- **Address**
Shows the network address of the connected network node.
This attribute can be modified.
- **Type**
Shows the type of the network node, e.g. `HW module`, `Data Server` or others.
This attribute cannot be modified.
- **PLC**
If the type is a `HW Module`, the name of the PLC will be displayed.
This attribute cannot be modified.
- **Name**
Shows the name of the network node. E.g. the name of a `Dataserver` or the identifier of a hardware module with rack type, slot number and module type.
This attribute cannot be modified.

The following figure shows the list of network nodes for an Ethernet segment.

The screenshot shows the Unity Application Generator window for a project named 'training01'. The window title is 'Unity Application Generator - training01 [Exclusive] - [Network Nodes [ETHSTRAN...]]'. The menu bar includes 'File', 'View', 'Generate', 'Window', and 'Help'. Below the menu bar is a toolbar with various icons. The main area displays network configuration details for an Ethernet segment:

- Network Segment:** ETHSTRANG01
- Network Type:** Ethernet
- Subnet Mask:** 255.255.255.0
- Default Gateway:** (empty)

Below these details is a table with the following data:

Address	Type	PLC	Name
192.168.0.11	HW Module	PLC01	Local_Slot7_140-NOE-771-11
192.168.0.12	HW Module	PLCtst	Local_Slot14_140-NOE-771-01
192.168.0.13	HW Module	PLCtst	Ethernet I/O_Drop14_Slot1_PowerMeter
192.168.0.14	HW Module	PLC02	Local_Slot7_140-NOE-771-11
192.168.0.20	Data Server		FIX

At the bottom of the window, there are buttons for 'Generate PLC(s)' and 'Generate HMI(s)', along with a date '13.06.2004' and a time '13:31'.

The header of the table contains the name of the `Network Segment` and the `Network Type` (`Modbus Plus` or `Ethernet`). Additionally the `Subnet Mask` and the `Default Gateway` will be displayed for `Ethernet` segments.

Routing Paths Description and Properties

What are Routing Paths?

The Routing Paths define the bridges and their addresses which connect different Modbus Plus Network Segments.

The Routing Paths should be defined early in the project. The definitions made here will appear as options for other elements.

Properties

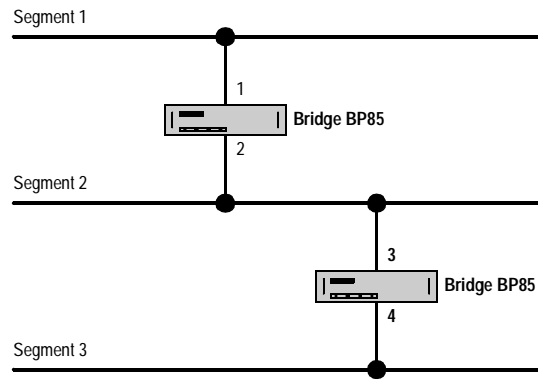
Properties of the Routing Paths:

Property	Description	Options	Comment
Send	Name of transmitting segment	Select from list	Segments have to be defined before the bridges can be defined
Receive	Name of receiving segment	Select from list	Segments have to be defined before the bridges can be defined
Path	Address of the bridge or bridges at the transmitting segment		According to the respective network addressing options.

Example Path Definition

Paths between segments that are connected via several bridges have to be defined explicitly.

Example path definition:



Paths for different send/receive combinations

Send	Receive	Path
Segment 1	Segment 2	1
Segment 2	Segment 1	2
Segment 2	Segment 3	3
Segment 3	Segment 2	4
Segment 1	Segment 3	1.3
Segment 3	Segment 1	4.2

3.3 The PLC Group

Overview

Introduction

This section describes the PLC Group and their properties.

What's in this Section?

This section contains the following topics:

Topic	Page
Introduction	85
PLC Properties	87
PLC Channels	90
Copy and Paste of PLCs	92
PLC <-> PLC Communication via Modbus Plus	93
PLC <-> PLC Communication via Ethernet	95
Additional Racks	97
Racks and Modules	98
Enhanced Ethernet Module	107

Introduction

Introduction

In the PLC group the PLCs are defined.

- A PLC is built up of Racks.
- A PLC has properties which define the type of the PLC, CPU, Local rack, ...
- A PLC has communication channels assigned which define to which other network nodes a communication takes place.

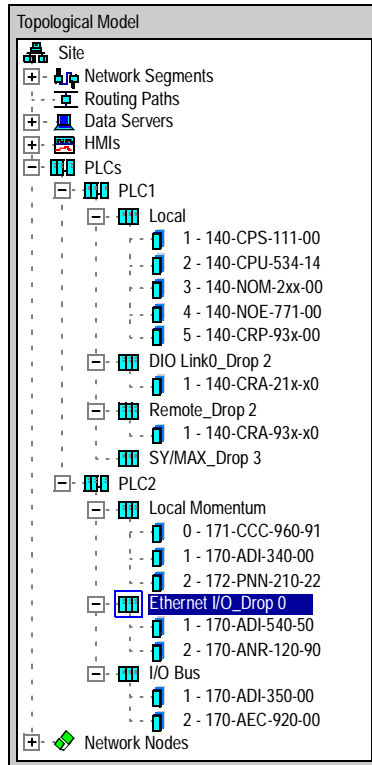
For each PLC the complete configuration with racks and modules inside the racks is defined. In addition the communication of the modules with the process is also defined, e.g. how the variables are connected to the I/O modules.

Structure of the PLC Group

The PLC groups contain

- PLCs,
 which contain
- Racks,
 which contain
- Modules.

Example of a PLCs group



PLC Properties

Overview A PLC has different types of properties. Some of the properties have to be defined explicitly by the designer, some are derived from definitions made in other objects.

Structure of a PLC A PLC always represents one Concept/Unity Pro project. If there are several PLCs, each of them will have its own Concept/Unity Pro project. Concept and Unity Pro PLCs can be mixed in the same UAG project. For each Concept/Unity Pro project, the PLC node consists of:

- the PLC type,
- the PLC configuration,
- the IO map and
- communication channels.

Properties to be Defined

The following properties have to be defined for each PLC:

Basic	Comment
Name	Unique name for the PLC
Description	Short description of the PLC
PLC Project Name	Unique name for the Concept/Unity Pro project. The project will be generated inside the <code>PLC Project Path</code> defined in the <code>Options</code> dialog of Unity Application Generator. Each project will be stored in its own directory.
PLC Family	Defines the programming software (Concept or Unity Pro) and the PLC family the PLC comes from. The valid PLC families are displayed in a list.
Local Rack	<p>For Concept: Defines the type of the local rack. For Quantum PLCs rack types with a size between 2 and 16 can be defined. For Momentum PLCs the rack type can not be changed. The rack size is fixed to 2, representing the I/O base and the communication adapter, which can be added to the processor.</p> <p>For Unity Pro: For Quantum and Premium PLCs the <code>Local Rack</code> can be selected during the generation of the new PLC. The rack can be changed within the rack properties.</p>
CPU	<p>Defines the CPU type. The valid CPU types are displayed in a list.</p> <p>For Concept/Quantum the CPU model can be changed later if necessary.</p> <p>For Concept/Momentum the CPU can not be changed after the PLC has been created.</p> <p>For Unity/Quantum the CPU can not be changed after the PLC has been created.</p> <p>For Unity/Premium the CPU can not be changed after the PLC has been created.</p>

Addresses	Comment
Address ranges	<p>Default values for the address ranges are predefined. For optimum configuration, address ranges can be assigned by the user by filling out the columns <i>Start</i> and <i>Length</i>. Within these ranges the addresses are automatically assigned during the generate process of memory mapping.</p> <p>Quantum/Momentum: I/O (local I/O, Ethernet I/O, Momentum I/O, etc.) are mapped to the <i>HW Modules</i> address range.</p> <p>Premium: For I/O located on the XBus no address ranges are assigned, because topological addresses are used. I/O located on a bus (Ethernet I/O, Modbus Plus I/O, Momentum I/O, etc.) are mapped to the <i>PLC Communication</i> address range.</p> <p>Note: A change of address ranges leads to a remapping of addresses; this may require to stop the PLC.</p>
Reserve	<p>For the different kinds of registers (0x, 1x, 3x and 4x for Concept; %M, %I, %IW and %MW for Unity/Quantum and %M and %MW for Unity/Premium) reserved register ranges can be defined. These ranges will not be used by Unity Application Generator. These ranges can be used in Concept/Unity Pro for functionality that is not defined in Unity Application Generator, for example IO modules on field busses.</p>

Properties which are Displayed

The following properties can be displayed for each PLC:

IO Statistics	Comment
IO statistics	<p>The <i>IO statistics</i> offers the information how many IO modules are needed for the IO variables of the existing Equipment Modules and Control Modules. It shows the number of all IO variables, the mapped IO variables and free IO points separately for inputs/outputs, digital/analog.</p>

PLC Channels

What is a Channel?

A Channel describes

- to which objects of the control system the PLC communicates,
- via which communication path the communication takes place,
- an alternative communication path on which the communication takes place and
- which variables have to be exchanged.

The communication is always done through a PLC. This means, one communication partner in a Channel is always a PLC.

Communication Definition

How it is defined and what has to be communicated via a Channel depends on the communication partner.

Communication definition:

Communication Partner	Definition Method
Data Server	A Channel to exactly one Data Server must be assigned for each PLC. The definition of which variables have to be communicated to the Data Server is implicitly defined by the relation of an Equipment Module to a PLC, the Control Domain (and thus the HMI) to which the Equipment Module is assigned, and the communication type of the variable. All variables of the communication type PLC_HMI are communicated.
PLC	Each variable to be communicated is put into the PLC communication table by drag and drop.
Net Partner	Each variable to be communicated has to be put into the net communication table by drag and drop (only variables of type PLC_NET).

Communication Methods

The communication method depends on the communication partner.
Communication methods

Communication Partner	Method	Comment
PLC	for Modbus: Peer Cop	Peer Cop data is only transferred between nodes on the local network. Transfer through bridges to other nodes is not possible. Peer Cop data is restricted to 32 words. 1 word is always used to check the health status of the communication Note: The communication between different networks with MSTR is not supported.
	for Ethernet: I/O Scanner	IO scanner data is restricted to 100 words. 1 word is always used to check the health status of the communication Note: The communication between different networks with MSTR is not supported.
HMI	Via located variables	Operator inputs are transmitted immediately to the PLC. The refresh rate for reading PLC values depends on the refresh category. There are three categories for the refresh rate (how often the communication takes place). The category is specified by command type and alarm of the variable: <ul style="list-style-type: none"> ● Event Events have the highest priority because events like alarms have to be transmitted as soon as possible. ● Synoptic Synoptic variables typically represent the real-time values of animated screens. They are communicated when the screen is displayed ● Parameter Parameters are values which are transmitted less often.
Net Partners	Via located variables	In the programming tool of the Net Partner the control engineer has to take care to poll the required data. Net Partners communicate via PLC addresses, thus the variables to be communicated have to be located to the appropriate memory range. For each Channel to a Net Partner the ranges for bits (0x/%M) and words (4x/%MW) have to be configured. It is possible to create a comma separated value (CSV) file with these variables and addresses. This file can be imported into the programming software of the Net Partner.

Supported Data Types

Supported data types for PLC <-> HMI communication:

- BOOL(DIGITAL)
- BYTE (ASCII)
- WORD
- INT
- DINT
- UINT
- UDINT
- REAL (FLOAT)

Supported data types for PLC <-> PLC communication:

- All data types with the exception of structured variables
-

Copy and Paste of PLCs

Copy and Paste of PLCs

The user of Unity Application Generator is able to copy and paste every PLC within the Topological Model. The copy of the PLC includes every local or remote rack which is belonging to the PLC.

Note: After copy and paste of a PLC all network addresses of Modbus Plus and/or Ethernet segments are lost to avoid multiple addresses.

PLC <-> PLC Communication via Modbus Plus

Communication Method

For PLC <-> PLC communication via Modbus Plus, Unity Application Generator always builds up the communication as Peer Cop. Communication with MSTR Function Block is not supported.

Unity Application Generator automatically generates the Peer Cop configuration and logic necessary for communication. To define the Peer Cop communication with Unity Application Generator, it is necessary to understand some basic facts of Peer Cop.

Supported Communication Capabilities

Peer Cop communication is restricted to 32 words per Channel. Unity Application Generator always uses one word for the health check. The remaining 31 words are free.

16 bits (BOOL) are communicated in one word (WORD). Unity Application Generator builds the logic to transfer the BOOL as WORD.

Also variables of other types are transformed to WORD, e.g. a REAL is split into two WORD.

Each Modbus Plus connection to the Modbus Plus network can be used for:

- Quantum, Momentum:
One Global Channel, which means that this information is sent to / received by all PLCs on the same segment
- Quantum, Momentum, Premium:
One Specific Channel, which means that the information is explicitly sent to / received by one other PLC

A Quantum PLC can have up to 3 connections to the Modbus Plus network (via the CPU and 2 NOMs).

A Momentum PLC can have 1 connection to the Modbus Plus network (via the communication adapter 172-PNN-210-22 or 172-PNN-260-22).

Health Check

For Quantum and Momentum:

Unity Application Generator creates its own communication logic. One word is used for a counter to check if the communication is valid or has errors. For this purpose, one word of each Channel is allocated.

For Premium:

For health check use the IODDTs in Unity Pro.

Redundancy for Channels

For Quantum and Momentum:

To set up a redundant communication Channel, it is possible to define two alternative Routing Paths. If redundancy is selected, one Channel uses 2 Peer Cop connections.

Communication Failure If the first Modbus Plus connection fails, the generated logic switches to the redundant communication Channels, which means that the values from the second Peer Cop connection are used.
If the second Modbus Plus connection also fails, the values specified as failure values are used. If no failure values are specified, 0 respectively false are used.
If no redundant communication is specified in case of communication failures the generated logic switches directly to the failure values.

Channel Settings for a Specific Channel To set up a Specific Channel, the following rules apply:

- The Channel has only to be defined for the transmitting PLC.
- A Specific Channel is defined by selecting a specific PLC as "Communication Partner".
- A Channel for the receiving PLC automatically is set up by defining the Channel for the transmitting PLC.

Channel Settings for a Global Channel To set up a Global Channel, the following rules apply:

- A Channel has to be defined for the transmitting PLC and for all receiving PLCs.
- A Global Channel for the transmitting PLC is defined by selecting "ALL" as "Communication Partner".
- A Global Channel for the receiving PLC is defined by selecting the option "Select from existing global input" and afterwards by selecting an existing Channel from the list.

PLC <-> PLC Communication via Ethernet

Communication Method For PLC <-> PLC communication via Ethernet, Unity Application Generator always builds up the communication as Ethernet I/O Scanner. Communication with MSTR Function Block is not supported. Unity Application Generator automatically generates the Ethernet I/O Scanner configuration and logic necessary for communication. To define the Ethernet I/O Scanner communication with Unity Application Generator, it is necessary to understand some basic facts of Ethernet I/O Scanner.

Supported Communication Capabilities Communication capabilities

- Depending on the module up to 64 or 128 Channels can be defined.
- Up to 100 words can be exchanged on each Channel. One of these words is used to check the health of the communication Channel.
- 16 bits (BOOL) are communicated in one word (WORD). Unity Application Generator builds the logic to transfer the BOOL as WORD. Also variables of other types are transformed to WORD, e.g. e REAL is split into two WORD.
- In general, each entry in the Ethernet I/O Scanner configuration table allows to define read and write of defined registers. Unity Application Generator only uses the write functionality of each channel/entry.

Health Check **For Quantum and Momentum:** Unity Application Generator creates its own communication logic. One word is used for a counter to check if the communication is valid or has errors. For this purpose, one word of each Channel is allocated. The Quantum Ethernet I/O Scanner requires as a standard the definition of a health block of 8 words. Therefore Unity Application Generator allocates 8 words in the 3x register range. The Quantum Ethernet I/O Scanner requires as a standard the definition of a diagnosis block of 128 words. Therefore Unity Application Generator allocates 128 words in the 4x register range. The Momentum Ethernet I/O Scanner requires as a standard the definition of a health block of 4 words. Therefore Unity Application Generator allocates 4 words in the 3x register range. The Momentum Ethernet I/O Scanner requires as a standard the definition of a diagnosis block of 64 words. Therefore Unity Application Generator allocates 64 words in the 4x register range. This ranges will not be used by Unity Application Generator to check the health, see above.

For Premium:
For health check use the IODDTs in Unity Pro.

Redundancy for Channels

For Quantum and Momentum:
 To set up a redundant communication Channel, it is possible to define two alternative Routing Paths. If redundancy is selected, one Channel uses 2 Ethernet I/O Scanner entries.

Communication Failure


If the first Ethernet connection fails, the generated logic switches to the redundant communication Channels, which means that the values from the second Ethernet I/O Scanner entry are used.
 If the second Ethernet I/O Scanner connection also fails, the values specified as failure values are used. If no failure values are specified, 0 respectively false are used.
 If no redundant communication is specified in case of communication failures the generated logic switches directly to the failure values.

Generated Ethernet I/O Scanner Configuration

To set up a Channel with the Ethernet I/O Scanner, Unity Application Generator sets up the Ethernet I/O Scanner configuration according to the following definitions:

- Slave IP Address (destination node address)
- Health Timeout: Unity Application Generator uses it's own logic to check the health.
- Rep Rate is set by Unity Application Generator to 0 in that case the transaction will be repeated continuously
- Write Ref Master (start address in sending PLC)
- Write Ref Slave (start address in the receiving PLC)
- Write Length (number of words to communicate, max. is 100)

These definitions only take place in the sending PLC. The receiving communication partner does not require any setting in it's Ethernet I/O Scanner.

	CAUTION
	<p>Risk of communication failures</p> <p>Within Concept/Unity Pro, manual modification of the Ethernet I/O Scanner is prohibited except for the <code>Rep Rate</code> setting.</p> <p>Failure to follow this precaution can result in injury or equipment damage.</p>

Channel Settings for an Ethernet Channel

To set up a Ethernet Channel, the following rules apply:

- The Channel has only to be defined for the transmitting PLC.
- A Channel for the receiving PLC automatically is set up by defining the Channel for the sending PLC.

Additional Racks

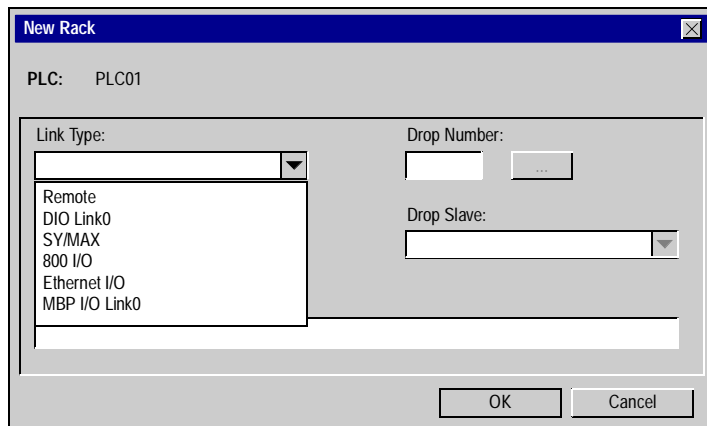
Overview

It is not possible to plug in Ethernet or Modbus Plus modules directly into the local or remote racks. Therefore the user has to arrange „virtual" racks within UAG for Peer Cop (Modbus Plus) and/or I/O Scanner (Ethernet) communication.

Create an additional Rack

After the creation of a new PLC with a local rack it could be necessary for the user to create additional racks. The local PLC rack is automatically equipped with the CPU module without any adjustments for the communication (Modbus Plus, Ethernet). The adjustments are required to get the possibility to create an additional rack. These racks could be necessary for using enhanced Ethernet modules, Modbus Plus modules or other ones.

The following figure shows an example for the `Link Types` of new racks.



New Rack vs. New Extension Rack

The user is able to select a New Rack or a New Extension Rack. For a New Extension Rack the user needs for different links an expander module on the local rack. For a new Ethernet I/O or Modbus Plus I/O rack parallel to an existing rack, the user needs a Modbus Plus or Ethernet communication. These racks are virtual, just for Peer Cop and/or I/O Scanner communication. The result of these racks after the code generation in Unity Pro or Concept are not physical racks but register adjustments for Peer Cop or I/O Scanner communication with the plugged modules.

Communication Parameters For the creation of a new rack parallel to an existing rack, the configuration for the communication is required. That means, the user has to set-up the communication parameters at the local rack, e. g. the user has to add an Ethernet module for the communication with an enhanced Ethernet module or has to adjust the Modbus Plus communication. The user has to create *Network Segments* first to realize this. Remote I/O require only a drop head, but they are not connected to a network.

Possible Racks The user is able to add different racks depending on the communication possibilities and parameters of the PLC. See also *Racks and Modules, p. 98*

Racks and Modules

Overview In Unity Application Generator for Concept and Unity Pro applications all PLCs with their racks and modules are defined. Which racks and modules are available depend on the PLC family defined in the PLC properties and the settings in the customization file.

In Unity Application Generator for Unity applications all PLCs with default racks are defined.

Depending on the selected module type, there are additional dialogs and windows available in Unity Application Generator:

- With the I/O module window and the variable window the I/O variables can be assigned to the I/O points (drag a variable from the variable table and drop on an I/O point).
- Communication modules or CPUs have a property dialog to assign the network segment and network address.
- Digital output modules have a property dialog to assign the timeout failure behaviour.

The definition of racks and modules creates the PLC configuration and the I/O map as well as the assignment of the variables to the I/O points of the modules.

In the following cases no entries will be generated in the I/O map:

- Momentum Modbus Plus I/O Modules,
 - Momentum Ethernet I/O Modules and
 - Generic Modules
-

Rack Properties The following properties have to be defined
(Concept)

Property	Description	Quantum Options	Momentum Options	Comment
Link Type	Defines how the rack is connected to the PLC	<ul style="list-style-type: none"> ● Local ● Local (Ex) ● Remote ● Remote (Ex) ● DIOLink 0 ● DIOLink 1 ● DIOLink 2 ● SY/MAX ● 800 I/O ● MBP I/O Link 0 ● MBP I/O Link 1 ● MBP I/O Link 2 ● Ethernet I/O ● Generic 	<ul style="list-style-type: none"> ● Local Momentum ● Ethernet I/O ● MBP I/O Link0 ● I/O Bus ● Generic 	For comments on the options see tables below.
Drop Number	Defines the drop number	<ul style="list-style-type: none"> ● Remote I/O: 2...32 ● Distributed I/O: 1...64 ● MBP I/O: 1 ● Ethernet I/O: Drop number corresponds to NOE slot 	<ul style="list-style-type: none"> ● Local Momentum: 1 ● MBP I/O: 1 ● Ethernet I/O: 0 ● I/O Bus: 2 ● Generic: 0...9999 	Drop number defines Modbus Plus address of this segment in case of distributed IO.

Property	Description	Quantum Options	Momentum Options	Comment
Rack Type	Defines the rack type	<ul style="list-style-type: none"> ● Local, RIO, DIO: 140 XBP 002 00, 140 XBP 003 00, 140 XBP 004 00, 140 XBP 006 00, 140 XBP 010 00, 140 XBP 016 00 ● SY/MAX: 8030-RRK-100, 8030-RRK-200, 8030-RRK-300 ● 800 I/O: AS-H810-100 , AS-H810-208, AS-H810- 209, AS-H819-100, AS- H819-103, AS-H819-209, AS-H827-100, AS-H827- 103, AS-H827-209, AS- H810-100, 800 I/O (55) ● MBP I/O: MBP I/O (64 slots) ● Ethernet I/O: Ethernet I/O (128) ● Generic: Generic (255 slots) 	<ul style="list-style-type: none"> ● Local: Local Momentum (2 slots) ● MBP I/O: MBP I/O (64 slots) ● Ethernet I/O: Ethernet I/O (64) ● I/O Bus: I/O Bus (44), I/O Bus (128) (CPU depending) ● Generic: Generic (255 slots) 	-

Comments on the options for `Link Type` for Quantum:

Link Type Option	Comment
Local	The local rack is automatically created together with the PLC. Note: Communication modules are only allowed in this rack.
Local (Ex)	A rack that is connected via a backplane expander (XBE) to the local rack.
Remote	A rack that is connected as remote IO. It is required to configure a remote head module (CRP-93x-00) in the local rack to connect remote IO.
Remote (Ex)	A rack that is connected via a backplane expander (XBE) to the remote IO rack.
DIOLink	A rack that is connected as distributed IO. A rack connected to the Modbus Plus segment connected to the CPU is called DIOLink0. Additional distributed IO can be connected with one or two NOM-2xx-00 as DIOLink1 and DIOLink2.
MBP I/O Link	A virtual rack that is connected as Modbus Plus I/O. A rack connected to the CPU is called MBP Link 0. Additional Modbus Plus I/O can be connected with one or two NOM-2xx-00 as MBP Link 1 and MBP Link 2.
Ethernet I/O	A virtual rack that is connected as Momentum Ethernet I/O. For each Ethernet I/O rack it is required to configure an Ethernet communication module (NOE) in the local rack.
SY/MAX	A rack belonging to the SY/MAX product family of IO hardware. It is required to configure a remote head module (140-CRP-93x-00) in the local rack.
800 I/O	800 I/O needs a remote head module (140-CRP-93x-00) in the local rack.
Generic	A generic rack is a rack which is not entered in the Concept I/O map. It serves for integrating generic modules into the configuration. Unity Application Generator assigns StateRAM addresses to the modules and the I/O variables mapped to the modules.

Comments on the options for Link Type for Momentum:

Link Type Option	Comment
Local Momentum	The local rack is automatically created together with the PLC.
MBP I/O Link 0	A rack that is connected as Modbus Plus I/O. It is required to configure a Modbus Plus adapter (171-PNN-xxx) in the local rack.
Ethernet I/O	A rack that is connected as Momentum Ethernet I/O. (Only available for Ethernet processors.)
I/O Bus	A rack that is connected as I/O Bus. (Only available for I/O Bus processors.)
Generic	A generic rack is a rack which is not entered in the Concept I/O map. It serves for integrating generic modules into the configuration. Unity Application Generator assigns State RAM addresses to the modules and the I/O variables mapped to the modules.

Rack Properties (Unity Pro)

The following properties have to be defined

Property	Description	Quantum Options	Premium Options	Comment
Link Type	Defines how the rack is connected to the PLC	<ul style="list-style-type: none"> ● Local ● Local (Ex) ● Remote ● Remote (Ex) ● DIOLink 0 ● DIOLink 1 ● DIOLink 2 ● SY/MAX ● 800 I/O ● MBP I/O Link 0 ● MBP I/O Link 1 ● MBP I/O Link 2 ● Ethernet I/O ● Generic 	<ul style="list-style-type: none"> ● XBus ● Ethernet I/O ● MBP I/O Link0 ● Generic 	For comments on the options see tables below.
Drop Number	Defines the drop number	<ul style="list-style-type: none"> ● Remote I/O: 2...32 ● Distributed I/O: 1...64 ● MBP I/O: 1 ● Ethernet I/O: Drop number corresponds to the slot of the Ethernet module 	<ul style="list-style-type: none"> ● XBus: 0...7 ● MBP I/O: 1 ● Ethernet I/O: Drop number corresponds to Processor/ ETY slot ● Generic: 0...9999 	Drop number (equipment number) defines Modbus Plus address of this segment in case of distributed IO.
Rack Type	Defines the rack type	<ul style="list-style-type: none"> ● Local, RIO, DIO ● SY/MAX ● 800 I/O ● MBP I/O: MBP I/O (64 slots) ● Ethernet I/O: Ethernet I/O (128) ● Generic: Generic (255 slots) 	<ul style="list-style-type: none"> ● XBus ● MBP I/O: MBP I/O (64 slots) ● Ethernet I/O: Ethernet I/O (64) ● Generic: Generic (255 slots) 	-

Comments on the options for `Link Type` for Quantum:

Link Type Option	Comment
Local	The local rack is automatically created together with the PLC.
Local (Ex)	A rack that is connected via a backplane expander (XBE) to the local rack.
Remote	A rack that is connected as remote IO. It is required to configure a remote head module (CRP-93x-00) in the local rack to connect remote IO.
Remote (Ex)	A rack that is connected via a backplane expander (XBE) to the remote IO rack.
DIOLink	A rack that is connected as distributed IO. A rack connected to the Modbus Plus segment connected to the CPU is called DIOLink0. Additional distributed IO can be connected with one or two NOM-2xx-00 as DIOLink1 and DIOLink2.
MBP I/O Link	A virtual rack that is connected as Modbus Plus I/O. A rack connected to the CPU is called MBP Link 0. Additional Modbus Plus I/O can be connected with one or two NOM-2xx-00 as MBP Link 1 and MBP Link 2.
Ethernet I/O	A virtual rack that is connected as Momentum Ethernet I/O. For each Ethernet I/O rack it is required to configure an Ethernet communication module in the local rack.
SY/MAX	A rack belonging to the SY/MAX product family of IO hardware. It is required to configure a remote head module (140-CRP-93x-00) in the local rack.
800 I/O	800 I/O needs a remote head module (140-CRP-93x-00) in the local rack.

Comments on the options for `Link Type` for Premium:

Link Type Option	Comment
XBus	The XBus rack is automatically created together with the PLC. Note: Communication modules are only allowed in XBus_Rack 0 and extension of it.
MBP I/O Link 0	A rack that is connected as Modbus Plus I/O. It is required to configure a Modbus Plus adapter in the local rack.
Ethernet I/O	A rack that is connected as Momentum Ethernet I/O. (Only available for Ethernet processors.)

Module Properties

The properties depend on the selected module category.
Properties of a communication modules/adapters or CPUs:

Property	Description	Options	Comment
Link Type	The link type is defined by the system. Link types for Quantum: <ul style="list-style-type: none"> • The CPU is always Link 0 • Two additional NOMs are Link 1 or Link 2 Link types for Momentum: <ul style="list-style-type: none"> • The communication adapter on the CPU is always Link 0 	-	This property is available for Quantum with Modbus Plus only
Network Segment	Defines the segments to which the module is attached.	Choose from list.	The list depends on the definitions defined in the Network Segment setup.
Modbus Plus Address or IP Address	Defines the address of the module	Network address	The valid network addresses depend on the type of network.
Local Remote Bus Children	Defines the number of bus children for IO Bus.	Number of bus children	This property is available for Momentum BNO modules only.

I/O points spreadsheet for I/O modules:

Property	Description	Options	Comment
IOPoint	Displays the position of I/O point in I/O module	None	Display only
IO Type	Shows the type of I/O	<ul style="list-style-type: none"> ● Digital In/Out ● Analog In/Out 	Display only
Variable	Defines the variable assigned to the I/O point	Any variable of the appropriate data type	<p>The assignment can be performed by dragging and dropping appropriate variables.</p> <p>Note: To change the association of a variable to an IO point, the variable can be dragged and dropped within a module or even between modules.</p>
Address	Displays the address of the I/O point, if the memory has already been mapped.	-	Display only.
Timeout state	Defines failure action of a digital output in case of timeout	<ul style="list-style-type: none"> ● Not_Assigned ● Disabled ● Last value ● User defined 	<p>If an error occurs in a PLC output module, the user can define what should happen to the output signals:</p> <ul style="list-style-type: none"> ● Disable: the signal is set to 0. ● Last value: The signal holds the last value it had before it failed. ● User defined: The user can define the state of the signal if the hardware module fails.
Timeout value	Defines the value of the variable if a failure (timeout) occurs		Only available if <code>timeout state</code> is set to "user defined".

Note: The timeout state of the module has to be the same as for the variables which are assigned to the module.

Enhanced Ethernet Module

Overview

It is possible to create user-defined Ethernet modules which are not supported by default within UAG. An Ethernet rack belongs to an Ethernet communication module (like ETY 4103 or NOE 771-xx). After the creation of an Ethernet rack, the user can assign these Ethernet communication modules. See also *Additional Racks, p. 97*

Possible Modules for the Enhanced Ethernet Rack

Possible modules for the integration into the Enhanced Ethernet Rack are:

- All Momentum bases with an Ethernet communication head.
 - User-defined Modules, like Altivar 58 with Ethernet connection.
-

Special Modules

All Momentum modules communicate (read and write) on the 4x00001 address. But other communication devices like for example the SPANG Power Electronics - Power Controller Unit; 850 Series; Type 853 0040 E1 00 use an offset for the communication.

These registers and their offsets are:

- Read Ref Register Starting Address 4:200, Length 15
- Write Ref Register Starting Address 4:200, Length 5

Therefore it is possible to enter "Read Ref Slave" and "Write Ref Slave" register-starting address. UAG generates these addresses into the IO_Scanner table automatically.

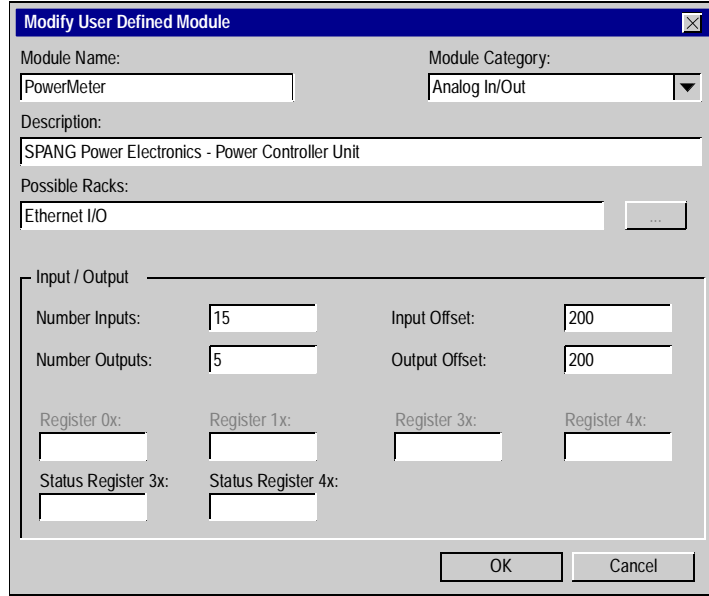
Input/Output Offset

The input offset ("Read Ref Slave") and output offset ("Write Ref Slave") can be entered in the customization editor. Thus all modules can be generated with the configured offsets.

Enhanced Ethernet Modules Customization

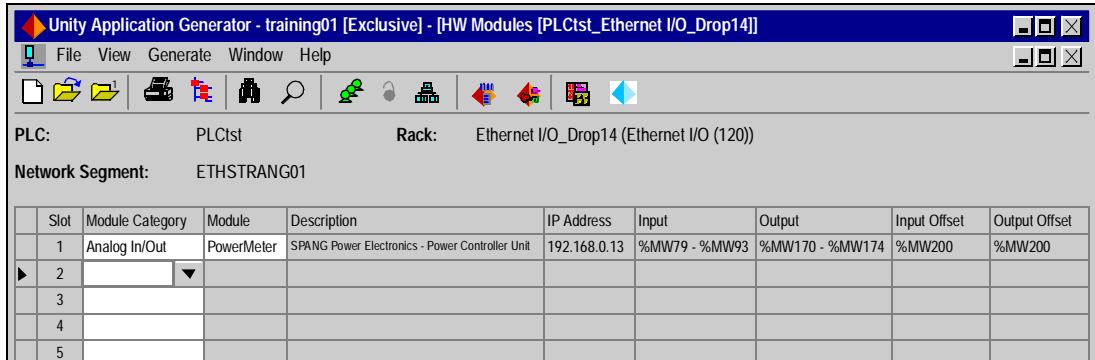
[To announce an Enhanced Ethernet Module within Unity Application Generator, the user has to costumize the module in the UAG Costumization as an user-defined module.]

The following figure shows an example for the configuration of an Enhanced Ethernet Module within the UAG Costumization.



Enhanced Ethernet Modules in UAG

[After the costumization the modules are usable in Unity Application Generator.] The following figure shows an example for an Enhanced Ethernet Module within an additional Ethernet rack.



**Enhanced
Ethernet
Modules in
Concept and
Unity Pro**

The following figure shows an example for the automatically generated I/O Scanner table (Concept/Unity Pro):

	Slave IP Address	Unit ID	Health Time-out (ms)	Rep Rate (ms)	Read Ref Master	Read Ref Slave	Read Length	Last Value (Input)	Write Ref Master	Write Ref Slave	Write Length	Diag Code	D
1	123.34.111.11	▼	0	2000	0	100001	400200	15 Hold Last	▼	000049	400200	5	Power
2	123.34.111.12	▼	0	2000	0	100017	400001	1 Hold Last	▼	000065	400001	1	TIO

3.4 The HMI Group

Overview

Introduction

This section describes the HMI Group and its properties.

What's in this Section?

This section contains the following topics:

Topic	Page
Introduction	111
HMI Properties	112
Control Domain Properties	113

Introduction

- Introduction** In the HMI group, the HMIs are defined. An HMI contains all screens of the related Equipment Module and their Control Modules.
- For each HMI several Control Domains can be defined.
 - Each Equipment Module is linked to exactly one HMI.
 - The relation of an Equipment Module to an HMI is defined indirectly by assigning an Equipment Module to a Control Domain.
-

What is an HMI? An HMI represents the controlled process for the operator and service engineers. An HMI contains all of the screens of all related Equipment Module and their Control Modules.

What is a Control Domain? Control Domains are created to allow that different operators control different parts of the process. In this way the complete HMI application is segmented into different Control Domains.

Associated with the Control Domain are the access rights of the operator. Thus, a Control Domain is part of the complete HMI application for which the operator is responsible. For this Control Domain the operator can control the Control Modules (start, stop, change setpoints, ...). The operator will only receive the alarms of his Control Domain(s). Which operator controls the Control Domain is not defined within Unity Application Generator but has to be configured in the HMI system.

HMI Properties

Overview

Unity Application Generator generates one screen for each Equipment Module in the HMI application. All Control Modules of each Equipment Module belong to the same HMI and are generated in the same Equipment Module screen.

Properties of an HMI

Properties of an HMI:

Property	Description	Options	Comment
HMI Name	Unique name of the HMI		The name has to follow the naming conventions defined during customization.
HMI Type	Defines the type of HMI Unity Application Generator will generate for	<ul style="list-style-type: none">● Monitor Pro● iFIX● Generic HMI	This setting has a major influence to Unity Application Generator. The functionality of different HMI differs widely. So HMI options in other dialog boxes depend on the HMI type selected here.
Comment...			The comment field is a free text field. It can be used to document the process design.
Documents...			Any number of files of different document types can be assigned to each element. The possible file types to be assigned are defined in the customization.

Control Domain Properties

Overview

Control Domains are related to one HMI. A Control Domain gathers the screens which represent a logically related part of the process which will be controlled by an operator. In this way they describe logical parts of the HMI and the structure of the HMI application. Thus, a Control Domain is part of the complete HMI application for which an operator is responsible. For this the operator can control the Control Modules (start, stop, change setpoints, etc.). The operator will only receive the alarms of his Control Domain(s). Which operator controls which Control Domain is not defined in Unity Application Generator but has to be configured in the HMI system.

Properties of a Control Domain

Properties of a Control Domain:

Property	Description	Options	Comment
Control Domain Name	Name of Control Domain		The name has to follow the naming conventions defined during customization.
Control Domain Description	Text field		
Comment	The comment field is a free text field. It can be used to document the process design.		

3.5 The Data Server Group

Data Server - Description and Properties

What is a Data Server?

The Data Server communicates data from one or several PLCs to HMIs. One Data Server can be used by multiple HMIs. A PLC is assigned to only one Data Server (polled only by one Data Server).

Relation to other Elements

The Data Server is linked to other elements of the Topological Model, to HMIs and PLCs:

- The assignment of a PLC to the Data Server is done via the communication Channel associated with the PLC.
- The assignment of the HMI to the Data Server is not done directly, it is done through elements of the Physical Model.
 - Each Equipment Module is managed within a Control Domain.
 - Each Equipment Module is therefore linked to an HMI.
 - Each Equipment Module is controlled within a PLC.
 - The PLC is polled by a Data Server via a communication Channel.

In this way the Data Server is assigned to an HMI through the relation of the Equipment Module.

Properties

The properties of the Data Server are mainly network oriented. There are no definitions to be done concerning other elements.

Properties of a Data Server:

Property	Description	Options	Comment
Data Server Name	Unique name of Data Server. Unity Application Generator will generate the Monitor Pro server application in the subdirectory DATASERVERNAME of the dataserver application path specified in the Options dialog. For iFIX, UAG will generate the tag database as DATASERVERNAME.PDB.		The name has to follow the naming conventions defined during customization.
Comment...	The comment field is a free text field. It can be used to document the process design.		
Documents...	Any number of files of different document types can be assigned to each element. The possible file types to be assigned are defined in customization.		
Generic Export Format	A name associated with a stylesheet file that defines how data is exported for this Data Server. The format names are defined in the customization.	Select from list.	Only available in projects for Generic HMLs.

Property	Description	Options	Comment
Network Timeout	Timeout for health status of communication		Can be defined from seconds down to ms
Network Type	-	<ul style="list-style-type: none"> ● Ethernet ● Modbus + 	If Monitor Pro or iFIX is used, only Ethernet is supported.
Network Segment	Defines the Network Segment the Data Server is connected to.	Select from list.	The Network Segments have to be defined before they are available in the list.
Network Address or IP Address	-	Valid addresses	The address has to be unique within one Segment.

3.6 The Network Nodes Group

Overview

Introduction

This section describes what Network Nodes are and their properties.

What's in this Section?

This section contains the following topics:

Topic	Page
Net Partner Description	118
Net Partner Properties	119
Other Node Description	120
Other Node Properties	121

Net Partner Description

What are Net Partners?

In a control system multiple hardware components are used. Components, which can be connected via the network with other components but are neither of type PLC nor of type HMI nor Data Server cannot be directly generated with Unity Application Generator, for example Magelis hand-held panels. For those so called Net Partners Unity Application Generator can export a CSV file containing the variables with the information needed for the polling by the Net Partner (addresses, data type, initial value and so on).

See also: *Net Partner Variables: CSV File Format, p. 366*

Requirements

For the inclusion of Net Partners the following conditions must be fulfilled:

- The Net Partners have to be defined in Unity Application Generator.
 - All required variables for the Net Partners have to be defined (in the Equipment Module or the Control Module) as PLC_NET variables.
 - A Channel has to be defined for the communication between the PLC and the Net Partner.
 - In the programming tool for the Net Partner an option has to be available to import a list of variables as a CSV file.
-

What is Generated for Net Partners?

For the PLC

- Unity Application Generator creates located variables so that the Net Partner can access them.

For the Net Partner

- Unity Application Generator creates a variable list in comma separated value format (CSV) for further processing, for example for the import into the Net Partner, see *Generation for Net Partners, p. 364*.
-

Magelis Export

UAG is able to generate CSV files for import into Magelis panels via a new button `Magelis Export` within the properties dialog box for `Network Nodes`. The following table shows the differences between the regular and Magelis export.

	Regular Export	Magelis Export
PLC Files	Creates a single CSV file for all PLCs.	Magelis export creates one file per PLC.
File Format	plc;networktype;networkpath; var;datatype;984addr; initial;alarmtext	N;var;datatype;984addr; initial;alarmtext

Net Partner Properties

Memory Range of Net Partners

The overall memory range for all Net Partners is defined in the PLC properties. The memory ranges for each Net Partner are defined in the Channel properties. Each Net Partner uses a 0x/%M and a 4x/%MW range in the PLC.

Properties

Properties of a Net Partner:

Property	Comment
Type	Net Partner (Type of the Network Node.)
Name	Unique name of Net Partner. The name has to follow the naming conventions defined during customization.
Description	A short description of the Net Partner can be made in the description field.
Bitmap file	For each Net Partner a bitmap file can be chosen, which is displayed in the Topological Viewer. The possible bitmap formats are: <ul style="list-style-type: none"> ● .bmp ● .jpg ● .gif ● .ico If no bitmap file is chosen, a default symbol is displayed in the Topological Viewer.
Network Type	Modbus Plus or Ethernet
Network Segment	Defines the segment the Net Partner is connected to
Network address or IP address	Modbus Plus or Ethernet network address
Comment...	The comment is a free text field. It can be used to document the process design.
Documents...	Any number of files of different document types can be assigned to each element. The possible file types to be assigned are defined in the customization.
Export Var...	A .CSV file containing the variables assigned to the current Net Partner is generated.

Other Node Description

What are Other Nodes?

In a control system multiple hardware components are used. Components, which can be connected via the network with other components but are neither of type PLC nor of type HMI nor Data Server nor Net Partner cannot be generated with Unity Application Generator. For those so called Other Nodes Unity Application Generator can reservate a network address to prevent the multi-assignment of network addresses.

Requirements

For the inclusion of Other Nodes the following conditions must be fulfilled:

- The Other Nodes have to be defined in Unity Application Generator.

What is Generated for Other Nodes?

Nothing. A network address is reservated, only.

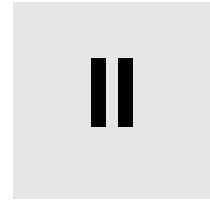
Other Node Properties

Properties

Properties of a Other Node:

Property	Comment
Type	Other Node (Type of the Network Node.)
Name	Unique name of Other Node. The name has to follow the naming conventions defined during customization.
Description	A short description of the Other Node can be made in the description field.
Bitmap file	<p>For each Other Node a bitmap file can be chosen, which is displayed in the Topological Viewer.</p> <p>The possible bitmap formats are:</p> <ul style="list-style-type: none"> ● .bmp ● .jpg ● .gif ● .ico <p>If no bitmap file is chosen, a default symbol is displayed in the Topological Viewer.</p>
Network Type	Modbus Plus or Ethernet
Network Segment	Defines the segment the Other Node is connected to
Network address or IP address	Modbus Plus or Ethernet network address
Comment...	The comment is a free text field. It can be used to document the process design.
Documents...	Any number of files of different document types can be assigned to each element. The possible file types to be assigned are defined in the customization.

Working with Unity Application Generator



Overview

Introduction

This part enables you to work with Unity Application Generator in a correct and effective way. For building an application follow the steps of the workflow and look up the detail information needed to perform the steps in the other parts of the manual.

What's in this Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
4	Rules for Working with Unity Application Generator	125
5	Tool Handling and Features for Effective Work	133
6	Workflow to Build an Application	155


Rules for Working with Unity Application Generator

4

Overview

Introduction

This chapter contains the rules, guidelines and warnings that you should keep in mind when working with Unity Application Generator.

	WARNING
	Risk of erroneous project configurations or loss of code Please read these rules carefully and have them in mind whenever you are working with Unity Application Generator. If you don't follow these rules, you risk erroneous project configurations, double work or even loss of code. Failure to follow this precaution can result in death, serious injury, or equipment damage.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
General Rules for Project Configuration and Generation	126
Open Customization and SCoD Editor	127
Rules Concerning the PLC and Concept	128
Rules Concerning the PLC and Unity Pro	130
Rules Concerning HMI	131

General Rules for Project Configuration and Generation

Define Process only Once

Note: Use the Unity Application Generator always to define the process and for making changes to the configuration later on! This assures the consistency of the definitions.

In classical automation environments, in which the definition of the PLC logic, the HMI screens and the network is done by separate tools, a lot of the definitions have to be repeated twice or three times. In UAGUAG these definitions are done only once and therefore are always consistent. This requires that all changes to the process defined by UAG are only defined with UAG and nowhere else e.g. all located variables have to be defined in UAG and must not be changed in Concept/Unity Pro.

New or Incremental Generation

Note: Be careful with the option for a completely new generation! If you have already started to complete the logic after an earlier generation, perform an incremental generation. Otherwise your manual changes will be lost!

A significant part of the PLC logic and the HMI screens is generated by UAG automatically. Nevertheless, it is necessary to add more PLC logic and to complete the HMI screens later on. Even though the logic and the screens are changed with the PLC and HMI programming tools, it is possible to make changes to all parts of the process defined in UAG and to rebuild the application incrementally **without** destroying any changes made outside of UAG.

Equipment and Control Moduls

If an Equipment or Control Modul is already generated, it is not possible to change the PLC anymore.

Open Customization and SCoD Editor

- Introduction** In UAG 2.1 the user is able to open the SCoD and Customization Editors within the user interface of UAG. To do this, two menu items - **View** → **Customization Editor** and **View** → **SCoD Editor** - as well as two toolbar buttons are available.
-
- Read-only Open** If the SCoD or Customization Editor are opened within UAG, the editors are in read-only mode. That means the user is able to browse the data, but cannot change it. But it is possible to save the data with a new filename.
- When the Customization Editor is started within UAG, it shows the current project's customization. If no project is open, the Customization Editor opens without a loaded customization.
- When the SCoD Editor is started within UAG, it shows the currently selected SCoD in View FFB mode. If a control module variable is selected in UAG, then the SCoD editor will open the respective SCoD and shows the selected variable. If no project is open in UAG, or no SCoD and no control module variable is selected, the SCoD editor opens without a loaded library.
- Both SCoD editor and Customization Editor will allow to open the databases (libraries or customization) in read-only mode also when working stand-alone. An additional option (check-box) will be added to the standard open file dialog. If the library of customization database is already opened exclusively by Unity Application Generator the database will be opened automatically in read-only mode.
-

Rules Concerning the PLC and Concept

Overview

In general all that can be defined in Unity Application Generator must be defined in UAG and nowhere else. It allows the user to define all necessary information about the memory of the PLC, e.g. variables, I/Os etc. If anything is changed to these definitions by Concept, UAG will overwrite these changes during the next generation of the code for Concept.

Please keep in mind ...

UAG offers a simple and fast way to update the Concept project with information modified by UAG by means of Incremental Generation. Using this feature without doing any manual changes in Concept you ensure that your process design - defined in UAG - is always in synchronization with the PLC program and configuration.

Rules Concerning Generated Logic

Apply the following rules:

- Do not manually create PLC logic that can be specified by means of UAG.
 - Don't change literals connected to inputs of Function Blocks, because their values are defined in UAG.
 - Do not remove variables from function blocks which have been generated by UAG.
 - If you change values in Concept, don't forget to update in UAG.
-

Rules Concerning Variables

Apply the following rules:

- Do not declare located variables in Concept, all located variables must be declared in UAG in order to ensure the consistency of the memory layout generated by UAG.
 - For additional logic, unlocated variables can be created in Concept, but it is even possible to define them in UAG because UAG offers the functionality to add free variables to either Equipment Modules or Control Modules.
-

Rules Concerning PLC Configuration

Apply the following rules:

- Whatever could be edited in the PLC configuration by UAG should be edited only by UAG.
Items which are not covered by UAG (and therefore allowed to be defined in Concept):
 - ASCII setup
 - Loadables other than IEC
 - Data protection
 - RTU extensions
-

Rules Concerning Names

Apply the following rules:

- Never change names of objects in Concept that have been generated by UAG, for example the name of a variable, the instance name of a Control Module function, or the name of a section.
If you do change the name UAG will not be able to find the objects any more in the Concept project and thus cannot generate subsequent changes that have been made to the process design.

Changes and PLC Stop

Changes to you project can be either changes of logic or changes of configuration. Concept allows the user to download changes to a running PLC without stopping the PLC. A change of configuration however (for example new definition of address ranges) is not possible without stopping the PLCs.

Changes of you project have different consequences:

If you ...	Then ...
Change only logic	You can download the new logic in the running PLCs. No production stop is necessary!
Change the configuration For example: <ul style="list-style-type: none"> ● Change of 0x and 4x address ranges ● Adding new Modules 	The change can only be carried out by stopping the PLCs. You have to accept a production stop!

Rules Concerning the PLC and Unity Pro

Overview

In general all that can be defined in UAG must be defined in UAG and nowhere else. UAG allows the user to define all necessary information about the memory of the PLC, e.g. variables, I/Os etc. If anything is changed to these definitions by Unity Pro, UAG will overwrite these changes during the next generation of the code for Unity Pro.

Please keep in mind ...

UAG offers a simple and fast way to update the Unity Pro project with information modified by UAG by means of Incremental Generation. Using this feature without doing any manual changes in Unity Pro you ensure that your process design - defined in UAG - is always in synchronization with the PLC program and configuration.

Rules Concerning Generated Logic

Apply the following rules:

- Do not manually create PLC logic that can be specified by means of UAG.
 - Don't change literals connected to inputs of Function Blocks, because their values are defined in UAG.
 - Do not remove variables from function blocks which have been generated by UAG.
 - If you change values in Unity Pro, don't forget to update in UAG.
-

Rules Concerning Variables

Apply the following rules:

- Do not declare located variables in Unity Pro, all located variables must be declared in UAG in order to ensure the consistency of the memory layout generated by UAG.
 - For additional logic, unlocated variables can be created in Unity Pro, but it is even possible to define them in UAG because UAG offers the functionality to add free variables to either Equipment Modules or Control Modules.
-

Rules Concerning Names

Apply the following rules:

- Never change names of objects in Unity Pro that have been generated by UAG, for example the name of a variable, the instance name of a Control Module function, or the name of a section.
If you do change the name UAG will not be able to find the objects any more in the Unity Pro project and thus cannot generate subsequent changes that have been made to the process design.
-

Rules Concerning HMI

Overview

In the following you find rules for the work with Unity Application Generator combined with a special HMI.

Rules for Monitor Pro / iFIX

When combining an Monitor Pro / iFIX HMI with Unity Application Generator, apply the following rules:

- If a new Control Domain is added in Unity Application Generator, add the corresponding Alarm Area and Security Areas in Monitor Pro / iFIX.
 - If the access levels are changed in Unity Application Generator customization, adjust the existing Security Areas in Monitor Pro / iFIX.
 - Do not modify in Monitor Pro / iFIX what has been generated by Unity Application Generator, for example:
 - Do not rename pictures.
 - Do not modify properties of AxtiveX controls (except color setting).
 - Do not delete tags.
 - And so on.
-

Tool Handling and Features for Effective Work

5

Overview

Introduction

Unity Application Generator is designed with the overall goal to enhance the efficiency of configuring a process control. Submitted to this goal, the user interface has been designed in a way that you will be able to use it intuitively if you are used to other Windows applications. This chapter explains the basic concepts of the user interface and features for enhancing your productivity when configuring your project.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Concepts of the User Interface	134
Working with Tables (Lists)	137
Drag & Drop of Objects, Modules and Variables	139
How to Find Objects with Search Criteria	140
Working with the Instrument List	142
Copy and Paste in the Physical and Topological Model	145
How to Build an Interlock Definition	147
Working with the Topological Viewer	151
How to View the Generation Status	154

Concepts of the User Interface

Introduction

The main goal of Unity Application Generator is to increase the efficiency of the process engineer and the control engineer in order to build the automation application. This is not only provided by the totally new concept of Unity Application Generator as a process design tool but is also the main goal for the design of the user interface.

Windows Look and Feel

The handling of Unity Application Generator is similar to other Windows programs like Word, Excel and so on.

The following general concepts are specific to Unity Application Generator

- **Object oriented properties:** The properties are accessible from the object.
 - **Context menus:** There are only a few menu items in the main menu. Most of the items are provided by context menus, accessible by right-clicking the mouse.
-

Powerful Navigation

Unity Application Generator offers the following navigation possibilities:

- **From the Physical Model and the Topological Model:** Context menus for all objects for navigation to tables/lists and dialog boxes for property definition
 - **From tables and lists:** Context menus for each row for navigation to other tables/lists and dialog boxes for property definition
 - **From Analyzer report:** Navigation to the erroneous object by double-clicking on the entry
 - **From Find report:** Navigation to one of the found objects by double-clicking on the entry
 - **From the Topological Viewer:** Navigation to property dialogs from the topological view.
-

Multiple User Tool

Multiple users are able to work on the same project at the same time. All changes made by a single user are directly available by the other users.

Note: If code generation is intended, the project has to be opened exclusively to prevent other users from changing the project while it is generated. This means that other users are excluded from opening the same project.

No Save Command

Unity Application Generator works with a database in the background and has multi-user capabilities and therefore has **no** save command. Otherwise a user could work with old data already changed by another user. Another advantage of this method is, that you don't lose any information if you have for example a PC crash.

Note: Because Unity Application Generator does not have a special save command all changes you make are immediately stored in the database.

Checking all Inputs

While defining an application with Unity Application Generator, all user inputs are checked directly as much as possible for correctness. Consequently the debugging time for the application is decreased.

In addition list boxes are used wherever possible to make the entry as easy as possible and to avoid faulty inputs. Many of the limits used to check inputs are set up during the customization process, which is carried out with a special tool, the Unity Application Generator Customization Editor.

Assigning Default Values

Wherever feasible, default values are assigned. Even though a user's input is sometimes required, it is not necessary to assign values to an object as soon as it is entered. In principle, three options can be assigned:

- A value
 - None/zero/not used
This means in general that the field is not used.
 - Not assigned
This means that a value is required but is not yet defined, e.g. the process engineer has defined something which has to be completed by the control engineer later on. During the analysis of the project all not assigned values are reported.
-

Refresh by User

The displayed project design is refreshed actively by the user with the Refresh function (in context menu or by pressing F5 button).

Since you are working in a multi-user environment, the project design can be modified by several users concurrently. This is very similar to the file and directory behavior on a shared file server.

Therefore it can happen that you see information on your screen, for example in the tree structure of the Physical Model or in a table, that is not up-to-date because it has already been changed by another user.

Note: Thus, from time to time or if you are not certain of the correctness of the display, you should refresh the information by selecting the refresh menu item in the different popup menus or simply press the F5 button.

Working with Tables (Lists)

Tables	Tables are used to enter and display lists of a group of objects with their attributes. Examples <ul style="list-style-type: none">● List of objects: Equipment Modules, Control Modules, Variables, HW modules ...● Instrument List
Features	<ul style="list-style-type: none">● Direct editing of fields● Copy and paste● Navigation to property dialog boxes or other tables
Context Menus	Context menus appear by right-clicking in a line or in the table dialog box. A context menu exists for example: <ul style="list-style-type: none">● For each line of the table with items like Delete (for deleting the line) or Properties (for accessing the properties dialog box of the object).● For the table dialog box with items like New (for adding a new line) or Paste for pasting a copied object into the table.
Add New Lines	A new line in a table is added by the New item from the context menu. With New you enter a dialog.
New... Dialogs	In the New... dialog you define the properties of the object(s) to be created. Apply saves the entered information, OK save the information and close the dialog.

Create Free Variables

In the **New Variable** dialog it is possible to create only one or several free variables in one step. Follow the steps:

If you want to ...	Then ...
Create a free variable or Create a number of equivalent free variables	<ol style="list-style-type: none"> 1. Right-click somewhere in the variables list. 2. Choose New from the context menu. 3. Enter the <i>Name</i>, <i>Description</i> and <i>Connection Type</i>, <i>Data Type</i> and <i>In/Out</i> of the variable(s). 4. Optional: Enter the <i>Startnumber</i> and <i>Number</i> of equivalent variables to create. 5. Press Apply. Result: In the variables list the new variable(s) appear. In the case of more than one variable, the names are numbered consecutively. 6. If necessary, enter other properties by moving through the tabs (<i>Alarm</i>, <i>Command</i>, <i>Display</i>). 7. Confirm with OK. The properties are saved as entered for all variables just having been created. The dialog closes.

Save Line

Whenever you enter information in a line of a table, the changes are stored in the database when you switch to another line. As long as you work in the different fields of the same line the data is **not** stored.

Select and Delete Lines

The following table explains how you select and delete lines in a table: :

If you want to...	Then...
Select a line	Click into the most left column of the line.
Select a range of lines	<ul style="list-style-type: none"> ● Click the left most column of the first line and drag the mouse over several lines without releasing the mouse button. Or: ● Click the left most column of the first line, press the Shift key while clicking the last line to be selected.
Select several non-consecutive lines	Click into the left most column of each line to be selected while pressing the Ctrl key.
Delete several selected lines	Use the del key.

Drag & Drop of Objects, Modules and Variables

Drag & Drop Editing Feature

Drag & Drop makes the configuration of a project easier for both the project engineer and the control engineer.

Drag & Drop in the Physical Model Tree

In the Physical Model tree you can move objects from one position to another by Drag & Drop . Valid/unvalid target positions are indicated while you are moving the object with the mouse.

What happens when an object is moved by Drag & Drop?

- All variable names are automatically changed corresponding to the naming conventions.
- Concept/Unity Pro sections, HMI features and so on are automatically moved and adapted.
- The changes are documented in the change history file.

Note: After moving an object within the Physical Model tree it is necessary to carry out a code generation in order to update the code.

Note: Moving a Unit will influence the execution order in Concept. The execution order in Unity Pro will not be influenced.

Moving HW Modules

For Concept/Unity Pro PLCs:

Hardware Modules can be moved within a rack and from one rack to another by Drag & Drop within the rack table or from one rack table to another (within the same PLC).

Note: If Communication or Analog Modules are moved in Unity Pro, the modules at the old positions are deleted and new modules are created at the new position. This has the effect that all data (channels, I/O, etc.) which was assigned to this modules is deleted.

The same happens if you are changing a RIO Drop number in Unity Pro. The complete rack including all modules is deleted and a new one is created.

Assigning Variables

From the variables table you assign variables by Drag & Drop for the following connections:

- Variable type IO_PLC to IO modules
 - Variable of all types to PLC-PLC-Channels
 - Variable type PLC_NET to Net Partner Channels
 - Variables to the interlock conditions
-

How to Find Objects with Search Criteria

Find Feature

Unity Application Generator offers the ability to search for objects, which meet certain criteria. A search is based on a specific object type (Control Module, variable, PLC...). One or several search criteria, combined by AND or OR function as filters. The criteria are based on the object attributes. In text strings the use of wildcard characters is possible.

Example: You can search for variables, which are alarm variables with an alarm priority `high`.

Components of a Search Condition

A search is based on the specification of the

- **Object type**

A single search criterion is composed of the following parts:

- **Attribute:** Attribute of the selected object type. Unity Application Generator provides a list box.
- **Operator:** Operator for the comparison with the criterion. Unity Application Generator provides a list box with operators corresponding to the kind of attribute.
- **Criterion:** Value of the attribute searched for. Depending on the kind of attribute Unity Application Generator provides a list box or a free field for entering a number or a text string.

Several search criteria are logically combined by

- **And/Or:** Operator for the combination of several search criteria. The first search criterion does not need an AND or OR. In a search condition with AND and OR, AND has the higher priority.
-

LIKE Operator

The LIKE operator can be used if the criterion is a text. The use of wildcard characters for the definition of string patterns is possible.

Note: In the text string or pattern the special characters `|` and `"` are **not** allowed.

Example for using wildcard characters to find a range of values: `LIKE Va*`

But: Example for specifying the complete value: `=Valve`

Wildcard Characters

The wildcard characters used in string patterns of LIKE expressions are defined as follows:

Wildcard Character	Kind of match	Example	Match	No match
*	Multiple characters	a*a	Aa, aBa, aBBBa	ABC
		ab	w	AZb, bac
		ab*	Abcdefg, abc	Cab, aab
[*]	Special character	a[*]a	A*a	Aaa
?	Single character	a?a	aaa, a3a, aBa	ABBBa
#	Single digit	a#a	a0a, a1a, a2a	Aaa, a10a
[...-...]	Range of characters	[a-z]	f, p, j	2, _
[!...-...]	Outside a range	[!a-z]	9, _	b, a
	Not a digit	[!0-9]	A, a, &, ~	0, 1, 9

Find Procedure

For finding objects fitting your search criteria follow the steps:

Step	Action
1	Open the Find dialog with View → Find .
2	Choose the object type you are looking for from the pull down list box.
3	Enter your first search criterion by filling out the fields Attribute , Operator , Criteria .
4	Add a new line in the table of search criteria (right-click in the table and left-click New).
5	Repeat Steps 3 and 4 until you have entered all your search criteria.
6	Combine the search criteria by AND or OR by filling out the And/Or column.
7	Start the search with Find .

Result

All objects meeting the search condition are displayed in the message window.

Note: From the message window you can navigate directly to specific objects by double-clicking on the entry.

Working with the Instrument List

Instrument List

The Instrument List is a flat list of Instruments (Control Modules). It is a feature for the process engineer allowing to enter the Instruments **before** defining the complete Physical Model hierarchy. Once the hierarchy down to the Equipment Module level is defined, the Instruments are simply dragged into the Physical Model and become Control Modules.

See also: *Instruments*, p. 42

Note: Using the Instrument List for the creation of Control Modules is an alternative way for creating them from the Physical Model tree. In comparing both ways, using the Instrument List enhances your productivity.

Advantageous Features

With the Instrument List you can do the following:

- Create multiple objects in an easy way, for example 100 equivalent motors at one time
- Simply drag and drop the selected Instrument(s) to the target Equipment Module node in the Physical Model
- Import Instruments from the P&ID (pipework and Instruments drawing)

What you Can Define for an Instrument

Using the Instrument List you can define Control Modules with all of their properties except of the property `PLC`.

For each Instrument you can define the following properties:

- Name
- Description
- Control Module Type
- Comments...
- Documents...
- Properties specific to the Control Module Type
- Variables

Note: For an Instrument you cannot define interlocks. This is possible only after having moved the Instrument into the Physical Model.

How to...

The table explains how to work with the Instrument List:

If you want to...	Then...
Import Instruments from the P&ID	See <i>Example: Import Instruments / Physical Model Hierarchy</i> , p. 188
Open the Instrument List	Press the Instrument List button or use the Site context menu.
Create a new Instrument or Create a number of equivalent Instruments	<ol style="list-style-type: none"> 1. Right-click somewhere in the dialog box. 2. Choose New from the context menu. 3. Enter the Type, Name and Description of the Instrument(s). 4. Optional: Enter the Startnumber and Number of equivalent Instruments to create. 5. Press Apply. Result: The property tabs corresponding to the selected Control Module Type appear. In the Instrument List the new Instrument(s) appear. 6. Enter the properties by moving through the tabs. 7. Confirm with OK.
Open the properties of an Instrument	Right-click somewhere in the row and choose Properties from the context menu.
Open the variables of an Instrument	Right-click somewhere in the row and choose Open Variables from the context menu.
Copy, paste, delete an Instrument	Right-click somewhere in the row and choose the desired item from the context menu.
Move or copy one or several Instruments into the Physical Model	<ol style="list-style-type: none"> 1. Select the Instrument(s) to be moved. 2. For moving: Drag the selected Instrument(s) into the target Equipment Module node of the Physical Model. For copying: Press Ctrl key while dragging the selected Instrument(s) into the target Equipment Module node of the Physical Model. <p>Result: Before pasting Unity Application Generator checks for uniqueness of names and if the maximum number of 32 Control Module in one Equipment Module will be exceeded. If the paste is allowed the Instruments become part of the Physical Model.</p>

Restrictions

The following restrictions apply to the Instrument List:

- **Control Module Type:**
 - It is possible to create an Instrument without specifying the Control Module Type (**controlmodule_type=Not_assigned**). You can drag the type-less Instrument to the Physical Model without restriction. In the Physical Model it becomes a type-less Control Module.
 - Once you have assigned the Control Module Type of a new Instrument with **Apply** in the **New** dialog, you **cannot** change it to another type. If you want to change the Control Module Type, delete this Instrument and create a new one with the correct Control Module Type.
 - **Edit properties:** You cannot edit properties for multiple Instruments. This is only possible in the **New** dialog during the creation of multiple equivalent Instruments.
 - **Max. 32 Control Modules in an Equipment Module:** Dragging a Control Module from the Instrument List to the target Equipment Module node of the Physical Model cannot be completed, if the Equipment Module already contains 32 Control Modules. In this case you need to delete one or more Control Modules in the Equipment Module or create a new Equipment Module.
 - **Uniqueness of names:** You cannot drag Control Modules with the same name into one Equipment Module. The uniqueness of names within one Equipment Module is checked before the dragging is completed. Change the names before dragging.
-

Copy and Paste in the Physical and Topological Model

Copy and Paste Editing Feature

In many cases the designer builds up similar structures in the Physical Model and the Topological Model of the process. For an efficient process design with Unity Application Generator you can copy and paste objects of the Physical and Topological Model. Copy and paste is possible within a project and even between projects.

Copy and Paste in the Physical Model

You can copy objects of the Physical Model. For example it is possible to copy a complete Area with the complete hierarchy. Depending on the copied object types, the paste menu item is only available for those objects which can contain the copied object types. For example: If the copied object is an Equipment Module, the paste is only available for Unit objects, because an Equipment Module always belongs to a Unit. When you try to paste the copied object, Unity Application Generator opens a window with a list of elements that are contained in the object and which have to be renamed for the copy because of uniqueness of names. Only after having filled out the complete list, Unity Application Generator will paste the new object.

Note: In the copy the references to the Topological Model have been removed. This means that for example the assignment of Equipment Modules or Control Modules to a PLC or HMI are lost or that IO_PLC variables are not mapped to IO modules anymore (It is not possible to map more than one variable to the same IO point on the same IO module). These references have to be added manually after the copy-paste action. For efficiency, you should use the Analyzer.

Copy and Paste in the Topological Model

In the Topological Model you can copy and paste PLCs and Racks. When you try to paste the copied object UAG opens a window with a list of elements that have to be renamed for the copy and network addresses which have to be assigned for the copy. Only after having filled out the complete list, Unity Application Generator will paste the new object.

Note: In the copy the references to the Physical Model have been removed. This means that for example the assignment of Equipment Module or Control Modules to a PLC or HMI are lost or that IO_PLC variables are not mapped to IO modules anymore (It is not possible to map more than one variable to the same IO point on the same IO module). These references have to be added manually after the copy-paste action. For effective working, you should use the Analyzer.

**Copy and Paste
between Projects**

It is possible to copy and paste objects from one project to another if the customization is identical. For this the same principles are valid as described above. Often, this is less work, because the copied objects may keep the same names, if they are not yet existent in the target project.

Note: Unity Application Generator allows only one project to be open. If you need to open two projects at once for the copy-paste action, you can run Unity Application Generator twice on your PC.

How to Build an Interlock Definition

Overview

The following procedure explains step by step, how you build an interlock definition for a Control Module. The interlock definition field allows a restricted syntax.

See also: *Interlocks for Control Modules*, p. 43

See also: *Generated Code: Equipment Modules, Control Modules and Interlocks*, p. 297

Procedure for Building an Interlock Definition

For building an interlock definition for a Control Module follow the steps:

Step	Action
1	Right-click on the Control Module to be interlocked in the Physical Model.
2	Choose Open Interlocks from the context menu. Result: The interlock dialog for this Control Module opens.
3	In the <code>Input</code> field select the name of the interlock input pin.
4	<p>Define each interlock condition (See <i>Interlock Condition, p. 44</i>) in one line of the table.</p> <ul style="list-style-type: none"> ● Enter interlock variables and condition variables by dragging them from a variables table and dropping them in the corresponding field of the interlock condition. ● Enter literals directly in the input field (only numbers). ● To check if a Boolean variable is TRUE, enter it as <code>Interlock Variable</code> and leave <code>Condition</code> empty. ● If desired, delete interlock conditions by using the context menu. <p>Note:</p> <ul style="list-style-type: none"> ● Variables belonging to another PLC may be configured as condition variables. In order to achieve a correct generation, this variable has to be communicated via a Channel to the PLC of the interlock. ● If you modify the name of a condition variable, it is automatically converted to a literal. ● If you need to use functions like SIN, COS,... see below.
5	<p>In the field Interlock Definition describe the logical combination of the interlock conditions by using <code>\$<condition number></code> to refer to the 1st, 2nd, 3rd ... condition and combining the conditions with logical operators and parenthesis.</p> <p>Note: Below the editing text box the expanded interlock definition is displayed, where <code>\$<condition number></code> is replaced by the condition from the table above. If you change a condition in the table, the expanded interlock definition is automatically adapted.</p>
6	Add a comment to your interlock definition.
7	<p>Save your interlock definition in the database with Apply.</p> <p>Result: When applying the input, a parser checks the correct syntax of your expression.</p>

Result

In case of a syntactically correct interlock definition the following will happen:

- The Concept/Unity Pro generator later will generate the code for each interlock definition as Function Block network connected to the interlock input pin of the corresponding SCoD.

If your interlock definition is syntactically incorrect, you may decide **not** to correct the syntax. In this case the following will happen:

- The interlock definition will be marked as a comment.
- The Analyzer will report a warning, when the project is analyzed.
- The code generator will generate a comment in the corresponding Function Block instead of PLC code.

**Functions like
SIN or COS in an
Interlock
Definition**

In some cases you may need complex functions like SIN or COS in your interlock definition. You must configure such functions directly in Concept/Unity Pro because Unity Application Generator allows only the use of comparing operators and Boolean operators. The solution is to define a free variable as a placeholder for the complex function and adding the complex logic after generation directly in Concept/Unity Pro.:

Step	Action	Example
1	Define a free variable in the Equipment Module. Note: This free variable shall be the placeholder for the complex function in the interlock condition.	You want to use the variable Angle as argument of the SIN function SIN(Angle) in an interlock condition. Define a free variable with the name SIN_ANGLE which shall be the placeholder for SIN(Angle).
2	Drop this free variable to an interlock condition where you need the complex function.	Define the interlock condition number 1: SIN_ANGLE=1
3	Use the reference number of the interlock condition in your interlock definition.	\$1
4	After Concept/Unity Pro generation add your complex logic to the corresponding ConceptUnity Pro section and connect it with the free variable pin of the generated Function Block.	Integrate a SIN Function Block into the corresponding section. Define Angle as input and connect the output to the free variable SIN_ANGLE.

**Example:
Interlock
Definition**

The screen shot shows a syntactically correct interlock definition:

The screenshot displays the Unity Application Generator interface. The main window is titled 'Unity Application Generator - Testproject'. On the left, there are two tree views: 'Physical Model' and 'Topological Model'. The 'Physical Model' shows a hierarchy: Site > area1 > pc1 > unit1 > equ1 > msr1 [MSR01], msr2 [MSR01], aia1 [AIA01T], and equ2. The 'Topological Model' shows: Site > PLCs > plc1, plc2; HMIs; Data Servers; Network Nodes; Network Segments; and Routing Paths.

The main workspace shows the 'Interlock [equ1_msr2]' configuration window. The 'Input' is set to 'PINLCK'. Below this is a table for the 'Interlock Condition':

Number	Interlock Variable	Operator	Condition Variable/Literal	Description
1	equ1_msr1_RDY			
2	equ1_aia1_DAHH			
3	equ1_aia1_SP	>	50.5	
4	equ1_aia1_TH	<=	equ1_aia1_TL	
5	equ1_msr1_FTR			
6	equ1_msr1_FTS			
7				

Below the table, there is a checkbox for 'Show Description' which is unchecked. The 'Interlock Definition' text area contains the logical expression: '\$1 or \$2 or (\$5 xor \$6) or (\$3 and \$4)'. Below this, a preview shows the expanded logic: '(equ1_msr1_RDY) or (equ1_aia1_DAHH) or ((equ1_msr1_FTR) xor (equ1_msr1_FTS)) or ((equ1_aia1_SP > 50.5) and (equ1_aia1_TH <= equ1_aia1_TL))'. A 'Comment' dialog box is open, showing the text: 'Internetlocks Motor Control MSR2 with Motor Control MSR1 and Mesurement Unit AIA1'. The dialog has 'OK' and 'Cancel' buttons.

Working with the Topological Viewer

Topological Viewer

The Topological Viewer is part of Unity Application Generator. The Topological Viewer allows the graphical representation of the Topological Model in a separate screen on Unity Application Generator. The topological view is created from the contents of the Topological Model.

With the Topological Viewer you get an overview of the current status of the automation hardware configuration of your project at any time.

Features

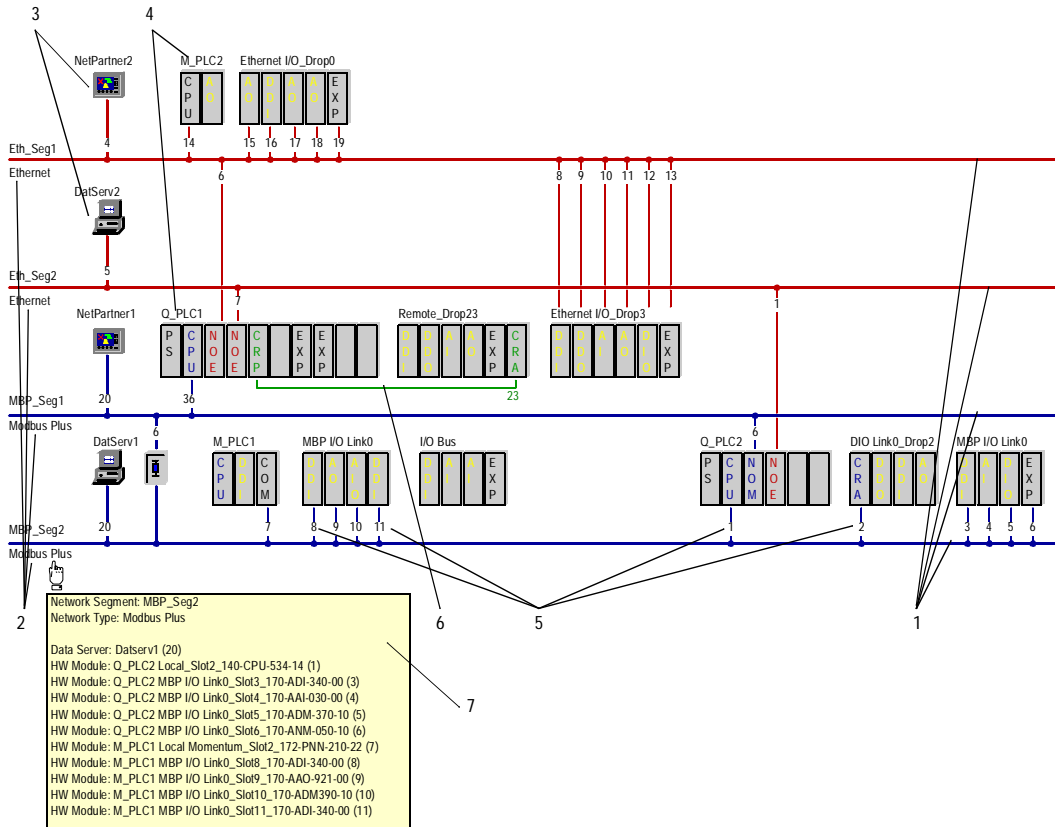
Compared to the Topological Model tree the topological view displays more information on the relations between the hardware components and more details like for example the free slots in a PLC in a graphical form.

The following features are interesting for you work:

- You can save the topological view as a bitmap and include it into a Word report file.
- The topological view is synchronized with the Topological Model.
- You can use the topological view for navigating to property dialogs.
- After having changed something in the configuration the topological view is refreshed as soon as the window of the Topological Viewer is re-activated.
- The window of the Topological Viewer can be open while other dialogs are open.

Note: The Topological Viewer is **not** a graphical editor. It is not possible to create new objects like PLCs, hardware modules, ... or deleting them. This has to be done by the menus of the Topological Model tree.

Example Topological View Topological View of an example project:



- 1 The topological view is based on the Network Segments which are represented as horizontal lines in different colors. The color indicates the protocol type (Modbus Plus or Ethernet).
- 2 The Network Segments are labeled with the name and protocol type.
- 3 The PLCs, Data Servers, Network Nodes or Routing Paths are connected to the Network Segments.
- 4 A PLC consists of racks and HW modules. The racks are represented as a box showing the slot with the HW modules and - in the display mode 'complete' the remaining empty slots. The HW modules are marked with an abbreviation of the module category. The communication modules are connected with the corresponding Network Segment.
- 5 Link number: The link number is the Modbus Plus address, the Ethernet address or the communication path (for Routing Path objects). For Ethernet addresses only the last byte is displayed, for communication paths only the first byte is displayed. Moving the mouse over the abbreviated address shows the complete address.

-
- 6 A thin line between the CRP head and the CRA drop module connects remote racks. The connection to a CRA module is labeled with the rack number.
- 7 A window opens containing information on the object when moving the mouse over the object.
-

How to ...

The table explains how to work with the Topological Viewer:

If you want to...	Then...
Change the display to <ul style="list-style-type: none"> ● Complete: Displaying all slots of PLCs ● Compact: Displaying only used slots ● Communication: Displaying only the hardware configuration for the communication 	Use the context menu by clicking somewhere in the window of the Topological Viewer.
See more information of an object	Move the mouse over the symbolized object. Result: Near to the mouse pointer a window opens containing detail information on the object. Note: If you click once on the symbolized object the object is selected in the Topological Model tree.
Move a non-visible object into the visible area of the topological view	Select the desired object in the Topological Model tree. Note: The selected object moves into the visible area of the display and its name displayed with a yellow background.
Edit an object	Right-click on the symbolized object or on it's label; the context menu opens Double-click on the symbolized object or on it's label; the objects property dialog opens Note: If you now enter new information in the property dialog, this will be automatically applied in the topological view.
Save the topological view as a bitmap	Right-click somewhere in the topological view, select Create Bitmap... and enter the target filename for the bitmap.
Include the topological view in a MS Word report file	In the Report dialog select the topological view.

How to View the Generation Status

Overview

You can view at any time, which objects have been changed after the last generation and have to be re-generated. This possibility is necessary for keeping the project consistent, for example if you have generated individual PLCs for testing purposes or other reasons.

Viewing the Generation Status

Possibilities for viewing the information about the status of generation:

If you need information about ...	Then...
How the project was generated (new or incremental?)	View the change history of the project with File → Change History
Which objects still have to be generated for the PLC	Open the list of objects to be generated with File → Properties → Objects to be generated → PLC
Which HMIs have to be generated	Open the list of HMIs to be generated with File → Properties → Objects to be generated → HMI
Which Data Servers have to be generated	Open the list of Data Servers to be generated with File → Properties → Objects to be generated → Data Server

Workflow to Build an Application

6

Overview

Introduction

The workflow described in the following is valid for projects developed at a single site. Unity Application Generator supports also multi-site development. In this case the overall project is splitted into partial projects, which can be developed in the same way as described in this chapter.

For more information concerning multi-site project development see: *Managing Distributed Project Development - Project Merge, p. 171*

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
General Workflow to Build an Application	156
Defining the Customization	158
Defining the Physical Model	159
Defining the Topological Model	160
Complete the Physical Model	163
Generate PLC and HMI Applications	164
Generate Import File for Net Partners	166
Document the Application or Individual Objects (Report)	167

General Workflow to Build an Application

Introduction

Building an application with Unity Application Generator is divided into several phases:

- Preparation phase
 - Application development
 - Test and documentation
-

Preparation Phase

The first phase is used to adapt Unity Application Generator to the requirements of the user company and the specific process. It offers a wide range of customization options.

Stage	Description
1	Customize Unity Application Generator so that it applies to the guidelines of the specific project e. g.: <ul style="list-style-type: none">● Naming convention● List of tools linked to the project● Allowed PLC types and PLC modules● ...

Application Development

The application development is a multi stage process:

Stage	Description
1	Define the Physical Model: <ul style="list-style-type: none"> ● Areas ● Process Cells ● Units ● Equipment Modules ● Control Modules ● Parametrize the Control Modules
2	Define your Topological Model: <ul style="list-style-type: none"> ● Network ● PLCs ● HMIs
3	Complete the Physical Model with control information e. g. <ul style="list-style-type: none"> ● which Equipment Module is controlled by which PLC, ● which Equipment Module belongs to which Control Domain, ● ...
4	Analyze the application: <ul style="list-style-type: none"> ● Missing information ● Consistency of information
5	Generate the PLC and HMI applications
6	Complete the PLC and HMI applications with the standard PLC and HMI programming tools e. g. Concept, Unity Pro, Monitor Pro, iFIX
7	Generate import files for Net Partners

Test and Documentation

The last phase covers the testing and bug fixing as well as the documentation of the project:

Stage	Description
1	Test your application.
2	Make all required changes to your process design in Unity Application Generator and if necessary regenerate incrementally.
3	Document your application with the report generator.

Defining the Customization

Overview

This section gives an overview of the first steps of a project.

Steps with the Customization Editor

Customization steps (for more information how to work with the Unity Application customization editor see *Customizing Unity Application Generator, p. 243*):

Step	Action
1	Define the libraries to be used in your project.
2	Define the tools which you use to document your process e.g. CAD, Word Doc, etc.
3	Define the description length of your comments.
4	Define the naming conventions for the different elements of Unity Application Generator.
5	Define the access levels.
6	Define the alarm groups.
7	Define the alarm priorities.
8	Define the archive names.
9	Define the communication frames.

Steps within Unity Application Generator

In Unity Application Generator do the following:

Step	Action
1	Open a new project and select a customization file (File → New).
2	Save the new project in the desired directory (File → Save as ...).
3	Define the path of the PLC application (View → Options → PLC → PLC Project Path). Note: The PLC project path is the path for Concept/Unity Pro projects, which will be generated each in its own subdirectory.

Defining the Physical Model

Overview

This section will give an overview of what has to be done to define the Physical Model.

Note: For efficiency use the possibility of importing data from other tools, like AutoCAD. Unity Application Generator is able to import parts of the Physical Model or the complete Physical Model as a CSV file.

See also:

- *The Physical Model*, p. 25
- *Interfaces with other Tools (Import and Export Features)*, p. 183

Steps to Define the Physical Model

To define the Physical Model take the following steps:

Step	Action
1	Enter Instruments (future Control Modules) in the Instrument List and specify the properties. Note: It is possible to import Instruments as a CSV file from the Pipe & Instruments Drawing (P&ID) into the Instrument List.
2	Define the Areas, comment each and attach the documents which are relevant to the Areas.
3	Repeat step 2 for Process Cells and Units.
4	Define the Equipment Module, describe and comment each and attach the documents which are relevant to the Equipment Module. PLC and HMI related definitions will be done after the Physical Model is built up.
5	Create Control Modules in the Equipment Module objects or drag Instruments from the Instrument List to the Equipment Module objects (by this, the Instruments become Control Modules and disappear from the Instrument List). Complete the Control Module properties. Note: It is possible to configure Control Modules without specifying a type (type-less Control Module), if you are not sure yet about the necessary Control Module implementation.

Defining the Topological Model

Overview

This section will give an overview of what has to be done to define the Topological Model. First the network should be defined. Information from this definition is needed for other elements in the Topological Model.

See also: *The Topological Model*, p. 59

Steps to Define the Network

To define the network take the following steps:

Step	Action
1	For Modbus Plus define the Network Segment (name and network type). For Ethernet define the Subnet mask and the Default Gateway.
2	For Modbus Plus define the Routing Paths (send/receive, path).

Steps to Define the PLCs

To define the PLCs take the following steps:

Step	Action
1	Define the PLCs, comment each and attach the documents which are relevant for the PLC. Then define the PLC project name (one project for each PLC). For Quantum in addition define the local rack size. For Momentum the local rack size is fixed to 2. The Momentum PLC is only available in Concept. Within Unity Pro the Quantum and Premium PLCs are available.
2	Assign the Equipment Module to the PLCs. In the properties of the PLC there is an I/O statistic which shows how many I/O points are needed.
3	Define additional racks for each PLC, defining its link type, drop number and rack size.
4	Define the hardware modules inside each rack.
5	Define address ranges for each PLC. Note: Check the addresses at the end of the process definition to optimize the PLC memory usage.
6	For communication modules and CPUs define their connection to the Network Segment and network address.

Steps to Define the HMI

To define the HMI take the following steps:

Step	Action
1	Define the HMI, comment each and attach the documents which are relevant to the HMI.
2	Define the Control Domains of each HMI and comment the Control Domain.
3	Assign the Equipment Modules to the Control Domains.

Steps to Define the Data Servers

To define the Data Servers take the following steps:

Step	Action
1	Define the Data Servers, comment each and attach the documents which are relevant for the Data Server.
2	Define the time out for the Data Server
3	Define the connection to the network (segment, address).

Steps to Define the Network Nodes

To define the Network Nodes take the following steps

Step	Action
1	Define the Network Nodes, comment each and attach the documents which are relevant for the Network Node
2	Define the connection to the network (segment, addresses)

Steps to Define the Communication Channels

To define the communication Channels of each PLC take the following steps:

Step	Action
1	Select a PLC
2	<p>Define the communication Channels and comment them. A Channel is required for each communication partner of the PLC.</p> <ul style="list-style-type: none"> ● Channel name ● Communication type (PLC, Data Server, Net Partner) and communication partner ● Communication Routing Path and timeout ● Address ranges for 0x and 4x in case of Net Partners <p>Note: For communication type Data Server the statistics (Number of available variables, number of used variable, variables used in percent) of the HMI variables can be displayed using the menu command Open HMI Communication. If more than 100 % of the a communication frame is used, with the next run of the memory mapper this communication frame will be recalculated. To prevent this, increase the number of HMI variables in the PLC properties.</p>
3	Repeat step 1 until you have defined the communication Channels and communication partners for each PLC.

Steps to generate the PLCs

To generate the PLCs take the following steps:

Step	Action in Unity Application Generator
1	Open project exclusively.
2	Generate PLCs.
3	Select a PLC in the Topological Model and press the Concept/Unity Pro icon in the tool bar to open the project in Concept/Unity Pro.
4	After each Generate PLC establish the connection between communication module and network in Concept/Unity Pro.

Steps to finalize the project

To finalize the project take the following steps:

Step	Action in Concept/Unity Pro
1	Create user logic of the project.
2	Analyse project using Build → Analyse Project (Unity Pro only).
3	Debug project if necessary.
4	Download project to PLC.

Complete the Physical Model

Overview

This section will give an overview what has to be done to complete the Physical Model following the Topological Model.

Steps to Complete the Physical Model

Steps to complete the Physical Model

Step	Action
1	Assign a PLC to each Equipment Module and define a Concept/Unity Pro section name for the generated code of the Equipment Module.
2	Assign an HMI/Control Domain to each Equipment Module.
3	Define free variables for all Equipment Modules.
4	Assign a type to type-less Control Modules; define the properties corresponding to the type.
5	Define free variables for the Control Modules.
6	To get an overview of the available and required IO points consult the IO statistics in the PLC properties. Add more IO modules if needed.
7	Assign variables of type IO_PLC to the IO modules (drag from the variable spreadsheet and drop on the IO point spreadsheet of an IO module).
8	Define the variables to be communicated e.g. between PLCs (drag from the variables spreadsheet to the PLC communication table) Note: The communication with the HMI via Data Servers is done automatically by Unity Application Generator.
9	Define the variables type PLC_NET to be communicated between PLCs and Net Partners (drag from the variables table to the Net Partners communication table). Note: Each variable of type PLC_NET may only be used once for this action, otherwise the address is ambiguous.
10	Optional: To get an overview of the calculated addresses, you can run the Memory Mapper separately.

Note: For the communication with the HMI via Data Servers you do not need to define anything. This is done automatically by Unity Application Generator.

Generate PLC and HMI Applications

Overview

This section will give an overview what has to be done to generate and test the PLC and HMI applications.

An application will neither be totally correct nor complete after the first code generation. This will require changes to the design. In the following the procedures for the first generation and update generations are described.

See also:

- *General Rules for Project Configuration and Generation, p. 126*
- *How to View the Generation Status, p. 154*
- *Overview of Generated Code and Generation Principles, p. 275*
- *Code Generation for Individual PLCs, p. 173*

Steps to Generate New PLC and HMI Applications

To generate the PLC and HMI applications take the following steps:

Step	Action
1	Run the Analyzer. It will check the entire application. The result is shown in the so called message window.
2	Check the Analyzer report and correct all problems. Each line in the report is a hyperlink to the source of the problem, which can be accessed by double clicking the line.
3	Repeat step 1 and 2 until all problems are solved.
4	Check the used memory ranges in the PLC properties and define the reserve locations.
5	Open the project with <code>Open Exclusive</code> Note: Open Exclusive excludes other users from working on the same project. This is necessary if you intend to carry out a code generation.
6	Generate the Concept/Unity Pro application.
7	Prepare and generate the HMI application. See: <i>Supported HMIs and their Setup, p. 193.</i>
8	Complete the PLC and HMI applications with the appropriate standard tools e. g. Concept/Unity Pro and Monitor Pro or iFIX.

Steps to Update the Application

To update the application take the following steps::

Step	Action
1	Identify changes to the process design which have to be made within ONESTEP Generator, for example: <ul style="list-style-type: none"> ● Definition of new variables ● Adding modules to a PLC ● Additional Equipment Modules or Control Modules ● ...
2	Make all changes to the design in Unity Application Generator. Rule: Whatever can be done within Unity Application Generator must be done by Unity Application Generator.
3	Run the Analyzer. It will check the entire application. Result: The Analyzer report is shown in a message window.
4	Check the Analyzer report and correct all problems. Note: Each line in the report is a hyperlink to the source of the problem, which can be accessed by double clicking the line.
5	Repeat steps 3 and 4 until all problems are solved.
6	Open the project with <code>Open Exclusive</code> Note: Open Exclusive excludes other users from working on the same project. This is necessary if you intend to carry out a code generation.
7	Generate the Concept/Unity Pro application. Note: By default the generation is done incrementally. In the Options dialog you can switch between incremental generation and new generation.
8	Prepare and generate the HMI application (this is done incrementally by default). See: <i>Supported HMIs and their Setup, p. 193</i> .
9	Complete the PLC and HMI applications with the appropriate standard tools e. g. Concept/Unity Pro and Monitor Pro or iFIX.

Generate Import File for Net Partners

Overview

For the communication of the Net Partners with the control application, Unity Application Generator generates a CSV file, which can be imported into the Net Partners.

See also *Generation for Net Partners*, p. 364.

Step to Generate Import File for Net Partners

Step to generate CSV-file for import in the Net Partners

Step	Action
1	For each Net Partner generate CSV-file (Net Partner → Properties → Export Variables)

Document the Application or Individual Objects (Report)

Overview

With the **Report** function you can generate different reports for the documentation of your project, for example the network configuration, the complete Physical Model or I/O variables. Additionally, you have the possibility to print a report of individual objects.

Unity Application Generator provides several document properties (DOCPROPERTY) (e.g. UAGVersion, UAGProjectName, etc.) which can be used in Microsoft Word template files (.dot). The final list of the document properties provided can be found in the Releasenotes.txt. An example template file (example.dot) can be found in "Doc" directory.

Note: The documentation is generated as a Microsoft Word document. This requires to have Microsoft Word installed on the PC on which the documentation shall be generated.

Steps to Document Individual Objects

To generate the documentation for individual objects take the following steps:

Step	Action
1	Right-click on the object to report on.
2	Select Report from the context menu. Result: The Report-dialog opens.
3	Define a Word template file and a destination for the generated report file.
4	Define if the sub-objects shall be included in the report (checkbox).
5	Run the report generator (Report button).

Steps to Document the Application

To generate the documentation the complete application take the following steps:

Step	Action
1	Open the report generator dialog (File → Report).
2	Define a Word template file and a destination for the generated report file.
3	Select the chapters to be generated.
4	Run the report generator (Report button).

Project Management



Overview

Introduction

In this part you will learn how a new project can be managed when using Unity Application Generator and what tasks the project administrator has to fulfil. The following features offer very interesting possibilities for a modern project management:

- Merge of projects
- Open interfaces (import/export via CSV files)
- HMI support
- Customization tool (Unity Application Generator Customization Editor)

What's in this Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
7	Managing Distributed Project Development - Project Merge	171
8	Interfaces with other Tools (Import and Export Features)	183
9	Supported HMIs and their Setup	193
10	Customization and Project Maintenance	241

Managing Distributed Project Development - Project Merge

7

Overview

Introduction

This chapter describes the collaborative engineering (remote, multi-site engineering efforts).

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Overview	172
Code Generation for Individual PLCs	173
Workflow of Distributed Project Development	174
General Preconditions for Project Merge	177
Merging of Physical Models	178
Merging of Topological Models	180

Overview

Managing Distributed Project Development - Project Merge

Unity Application Generator supports collaborative engineering (remote, multi-site engineering efforts) with the following functionalities:

- Project merging: An automation project is separated into partial projects which are developed by independently working engineering groups. The partial projects can be merged to a single, overall project in a safe and easy way.
 - Code generation for individual PLCs
-

Characterization of the Merge Functionality

The merge functionality has the following characteristics:

- All objects of the project to be merged will be created in the overall project exactly as they are defined in the project to be merged.
 - One task of engineering is to complete the generated PLC logic and the HMI applications. Therefore merging includes the existing Concept or Unity Pro projects, data servers and HMIs belonging to the merged project.
 - To be able to merge two separately developed projects, several pre-conditions have to be met. Before the merging takes place, both projects will be analyzed if they meet these conditions. If the Analyzer detects errors, merging is aborted and the user has to correct these errors first.
 - When the user selects to merge two projects, Unity Application Generator asks to save the current project as a version in the archive to have a backup of the existing project in case of undesired results.
-

Code Generation for Individual PLCs

Code Generation for Individual PLCs

It is possible to select an individual PLC and generate the code for this PLC via a context menu item.

This is especially useful in the case of distributed development, where different engineers or groups are responsible for different PLCs. This feature may also be useful for a control engineer elaborating the PLCs one after another.


What is Generated?

If you generate for an individual PLC only the variables of this PLC are mapped to memory addresses, all other variables are not influenced.

Rule

Ensure that the PLC projects are generated consistently and are downloaded correctly to the PLCs.

See: *How to View the Generation Status*, p. 154

	CAUTION
	<p>Risk of inconsistencies between the PLCs</p> <p>Be careful when generating individual PLCs! Generating only one PLC may cause inconsistencies between PLCs.</p> <p>This can for example be the case, if the communication between PLCs has been configured in Unity Application Generator. Then the generated PLC has already the correct communication setup while the other PLCs have not. This is comparable with the situation that the code for one PLC is already downloaded, while for the others not.</p> <p>Failure to follow this precaution can result in injury or equipment damage.</p>

Workflow of Distributed Project Development

Phase 1: Split the Work

Split the work of the process engineers into different groups as described in the following steps:

Step	Action
1	Define the customization.
2	Create a new Unity Application Generator project.
3	Define the network setup with Network Segments and Routing Paths.
4	If only one HMI application will be used for the complete project, define it now. If there will be more HMI applications, let each engineering group define its HMI application later.
5	Define the Physical Model hierarchy down to the level of responsibility of each group.
6	Define clearly which group is responsible for which part of the automation of the plant.
7	Create a copy of the project for each group, each copy (partial project) with a different name.

Guidelines for the Split

Do not split the following objects:

- Equipment Module
- PLCs
- HMIs
- Data Servers

Note: Even if the complete system is designed to work with only one Data Server, it is still necessary to define a Data Server in each partial project (because it is not possible to merge two iFIX PDB files). After the merge, the Data Servers must be merged manually in the overall project, see phase 4, step 3.

Phase 2: Independent Work of Groups

Each group now works independently to complete their part of the process design:

1	<p>Define the following for the project parts of their responsibility:</p> <ul style="list-style-type: none"> ● Equipment Module, Control Modules and variables with different names ● PLCs with different names and the PLC configurations ● The mapping of the objects of the Physical Model to the objects of the Topological Model ● Data Servers with different names ● Network Partners with different names ● HMIs (if not defined globally in phase 1) ● The communication between the PLCs (in the scope of the PLCs defined by the group)
2	<p>Generate the PLC for each partial project. Note: It is possible to generate code for individual PLCs, see <i>Code Generation for Individual PLCs</i>, p. 173.</p>
3	<p>Complete the Concept/Unity Pro applications manually as far as desired in each project group. Note: Manual changes in Concept/Unity Pro will not be lost! They will be taken over in the overall project when the project parts are merged.</p>
4	<p>Copy the Concept/Unity Pro projects belonging to the partial projects to the PLC project path of the overall project.</p>
5	<p>In the case of using iFIX HMIs:</p> <ol style="list-style-type: none"> 1. Generate for iFIX HMI for all partial projects. 2. Copy the iFIX GRF files of all partial projects into the local DYNAMICS/PIC directory. 3. Copy the iFIX PDB files of all partial projects into the local DYNAMICS/PDB directory.

Phase 3: Project Merge

After each group has completed their work the projects can be merged.

1	<p>Open one of the projects exclusively.</p>
2	<p>Start project merge with File → Merge and select the project to merge. Result: The projects will be analyzed for consistency and uniqueness of names and addresses. If there are errors reported, the projects will not be merged and the errors have to be corrected. If no errors occur, the projects will be merged. Note: The archived versions and the change history of the project to be merged will not be copied to the overall project.</p>
3	<p>Repeat Step 2 for all partial projects.</p>

**Rules after
Project Merge**

Once you have merged projects obey the following rules:

- After project merge **do not** use and modify the partial projects anymore!
- Perform all further modifications **only** in the overall project!

Note: A repeated merge is **not** possible because the overall project already contains the objects of the projects to be merged (for example Equipment Modules, PLCs) which would lead to errors during analysis of the projects to be merged.

**Phase 4:
Complete Overall
Project**

After the merge, the overall project has to be completed:

1	Define the communication between PLCs that belonged to different partial projects.
2	Define the interlocking between Control Modules that belonged to different partial projects.
3	<p>If you had created a Data Server for each partial project, but the overall project shall have only one Data Server, merge the Data Servers as follows:</p> <ol style="list-style-type: none"> 1. Delete all existing Channels to the obsolete Data Server(s). 2. Create new Channels between the PLCs and the finally desired Data Server. 3. Delete the obsolete Data Server(s). <p>Note: No additional manual work has to be done (like recreating the communication tables), and no data is lost.</p>
4	Generate the project incrementally.

General Preconditions for Project Merge

Overview	<p>When two projects are merged, Unity Application Generator checks some general conditions before the merge is possible. When merging two projects you open the main project and merge another project into the opened (current) project. If you have more projects to merge, you merge one project after the other into the current project.</p> <p>For two projects to be merged the following restrictions are valid.</p>
Software Version	<p>The projects to be merged have to be created with the same version of Unity Application Generator.</p>
HMI Brand	<p>The HMI brand has to be same in both projects.</p>
SCoD Libraries	<ul style="list-style-type: none">• The SCoD libraries used in both projects have to be the same.• The SCoD library and SCoD versions in both projects have to be the same.
Opening of Projects	<ul style="list-style-type: none">• The current project has to be opened exclusively.• The project to be merged must not currently be opened by another user.• All objects must be consistently generated. This means: the lists of objects to be generated must be empty, see <i>How to View the Generation Status, p. 154</i>.
Customization	<p>It is not required that both projects have the same customization. Thus, after a successful merge an analysis of the project customization may be necessary. This case is detected by Unity Application Generator and an analysis of the customization will be offered to the user.</p>
Network Addresses	<p>The network addresses of the network nodes in the project to be merged must be different from the addresses in the current project.</p>

Merging of Physical Models

Overview

All objects are identified by their name. Thus, if two objects are detected with the same name, they are considered as being the same, at least from the point of view of the merge function.

Note: There may be objects existing identically in both projects (identified by their name). In this case merging will keep the existing objects in the current project and will **not** check inconsistencies between the object properties in the two projects.

Merge of Elements and Rules

For the elements of the Physical Model, some rules have to be obeyed for correct merge:

Element	Rules	Merge Processing
Areas, Process Cells, Units	The hierarchy may be identical, partly overlapping or totally different in both projects. If they are partly overlapping, these objects have to be identical.	Merging will take care to combine the hierarchy of both projects appropriately.
Equipment Module	The Equipment Module names in both projects have to be different.	The Equipment Module of the project to be merged will be created identically in the overall project.
Control Modules	Control Module names must be unique within one Equipment Module. Two or more Control Modules with the same name may exist in both projects as long as they belong to different Equipment Module.	The Control Modules of the project to be merged will be created identically in the overall project.
Variables	Variable names must be unique within one Equipment Module or Control Module. Two or more variables with the same name may exist in both projects as long as they belong to different Equipment Module or Control Module.	The variables of the project to be merged will be created identically in the overall project.
Instrument List	Two or more Control Modules with the same name may exist in the Instrument Lists in both projects, because the name of the Control Modules in the Instrument List do not have to be unique.	The Control Modules in the Instrument List of the project to be merged will be created identically in the overall project.

Merging of Topological Models

Overview

All objects are identified by their name. Thus, if two objects are detected with the same name, they are considered being same, at least from the point of view of the merge function.

Note: There may be objects existing identically in both projects (identified by their name). In this case merging will keep the existing objects in the current project and will **not** check inconsistencies between the object properties in the two projects.

Merging of Elements and Rules

For the elements of the Topological Model, some rules have to be obeyed for correct merging:

Element	Rules	Merge Processing
Network Segments	The same Network Segments may exist in both projects. If Network Segments with the same name exist in both projects they have to be identical.	If Network Segments are existing in the project to be merged and not in the current project, they will be created in the overall project.
Routing Paths	The same Routing Paths may exist in both projects. The network addresses must be unique, for a new Routing Path.	If Routing Paths are existing in the project to be merged and not in the current project, they will be created in the overall project
PLCs	<ul style="list-style-type: none"> ● The PLCs in both projects have to be different. ● The Concept/Unity Pro projects have to be generated before the merging. ● The Concept/Unity Pro projects belonging to the project to be merged have to be copied manually to the PLC project path of the overall project. 	<p>The PLCs of the project to be merged will be created identically in the current project.</p> <p>Merging will take care to create all administration information to enable the user to continue to generate incrementally the PLCs in the overall project.</p>

Element	Rules	Merge Processing
HMIs	<ul style="list-style-type: none"> ● The HMIs in both projects may be the same. ● Monitor Pro / iFIX: The HMI screens (GRF files) have to be generated before the merging and the Monitor Pro / iFIX GRF files have to be copied manually to local DYNAMICS\PIC directory. 	<p>If HMIs exist in the project to be merged and not in the current project, they will be created in the overall project.</p> <p>Merging will take care to create all administration information to enable the user to continue to generate incrementally the HMIs in the overall project.</p>
Control Domains	The control domains in both projects may be the same as long as they belong to the same HMI.	
Data servers	<ul style="list-style-type: none"> ● The data servers in both projects have to be different. ● Monitor Pro / iFIX: The Data Servers (tag databases = PDB files) have to be generated (as part of the HMIs) before the merging and the iFIX PDB files have to be copied manually to local DYNAMICS\PDB directory. 	The Data Servers of the project to be merged will be created identically in the overall project.
Net Partners	The Net Partners in both projects have to be different.	The Net Partners of the project to be merged will be created identically in the overall project.
Channels	The Channels in both projects have to be different, since all Channels belong to PLCs.	The Channels of the project to be merged will be created identically in the overall project.

Interfaces with other Tools (Import and Export Features)



8

Overview

Introduction

Unity Application Generator allows for the import and export of data to and from its database by the means of CSV files. This concept of open interfaces facilitates the project development, because data can be exchanged with other tools like for example AutoCAD.

In this chapter you learn about the concept of open interfaces and read about examples for using them.

As reference information use: *Format of the CSV Files for Import and Export*, p. 367.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Concept of Open Interfaces	184
How to Import/Export CSV Files	186
Example: Import Instruments / Physical Model Hierarchy	188
Example: Import Initial Values for Variables	191

Concept of Open Interfaces

Open Interfaces Unity Application Generator offers open interfaces for the import and export of data from and to other applications. Import and export is possible for objects of both, the Physical and the Topological Model. The data is exchanged via CSV files (**C**omma **S**eparated **V**alue) for which a specific format has been defined in Unity Application Generator.

During the import process Unity Application Generator checks all data before it is included in the database, for example for compliance with the project customization. Erroneous data is ignored and a message informs the user about the errors.

Examples

Examples for import:

- Import Instruments from the pipework and instruments drawing, created for example with AutoCAD
- Import initial values of existing variables for tuning purposes

Examples for export:

- Export a PLC for the creation of wiring diagrams with AutoCAD
 - Export the variables for a Net Partner
-

Restrictions for Unity Pro

Only objects which are created in Unity Application Generator can be imported e.g.:

- Instruments
- Initial values of existing variables
- Hardware modules in Modbus Plus, Ethernet or Generic racks.

Objects which are NOT created in Unity Application Generator CANNOT be imported e.g.:

- Unity Pro PLCs
- Unity Pro racks

The export of objects is not restricted. All objects can be exported.

CSV File

A CSV (comma separated value) file is an ASCII file containing a number of lines. Each line of a CSV file contains a number of entries, separated by a separator, for example a comma or a semicolon or a tab.

Format of CSV Files

The CSV files for import and export to and from Unity Application Generator are defined as follows:

- The first entry in each line contains a keyword to specify which object is contained in the line.
- The following entries in the line contain the attributes of the object. The order of the values is fixed for each keyword.
- The separator can be selected by the user.

The format of CSV files is the same for import and export.

For more information see: *Format of the CSV Files for Import and Export, p. 367*

How to Import/Export CSV Files

Note: Import / Export between different product versions of UAG is not supported.

Procedure for Import

For importing a CSV file follow the steps:

Step	Action
1	Prepare a CSV file corresponding to the format described in <i>Format of the CSV Files for Import and Export</i> , p. 367 Note: The import data must fit the naming conventions of your target project customization.
2	Open the Unity Application Generator target project.
3	Set the separator in Unity Application Generator to fit the separator in your CSV file (View → Options).
4	In the Analyzer options (View → Options → Analyzer) set if warnings should be displayed and if the import should be stopped after a certain number of errors or warnings.
5	Start the import process with File → Import...

Result

During the import process the message window is used to display the current activity of the import. The CSV file is processed line by line. First, the line is analyzed to be syntactically and logically correct. In case of an error a message is written to the message window. Unity Application Generator ignores the erroneous line and processes the next line. If the line is correct, the line is interpreted and the different objects are created - if necessary. The **Import** function uses the settings of the Analyzer options whether to display warnings and stop after a certain number of errors or warnings.

Procedure for Export

For exporting parts of the physical or Topological Model as a CSV file follow the steps

Step	Action
1	Set the separator in Unity Application Generator as desired for the target CSV file (View → Options).
2	Open the context menu of the object to export by right-clicking on it in the model tree and select Export .
3	Follow the instructions in the dialog box.

Result

Unity Application Generator generates a CSV file containing the selected object including all objects which are contained in it. The lines in the CSV file correspond to the format described in *Format of the CSV Files for Import and Export*, p. 367.

Note: If you intend to export the complete project, thus Physical and Topological Model, use **File** → **Export...** .

Example: Import Instruments / Physical Model Hierarchy

Introduction

The import functionality provides an open interface to fill Unity Application Generator project databases from an external source in a simple but very powerful way. By means of a CSV (comma separated values) file it is possible to create Instruments or even the complete hierarchy of the Physical Model.

See also: *Instruments*, p. 42 and *Instruments: CSV File Format*, p. 385.

Import from P&ID

The import feature is especially useful for customers who start the design with the P&ID, that is with the drawing of the pipe works and instruments created with another tool like AutoCAD with Rebis extensions. Having already defined all instruments (Control Modules) with these kind of tools, you can save the time to re-enter the same information into the Unity Application Generator project by using the import functionality.

Line in CSV File

For importing Instruments or the Physical Model hierarchy each line in the CSV file has the following entries:

`INSTRUMENT;controlmodule_name;controlmodule_type;library_name;controlmodule_description;controlmodule_comment;equipmentmodule_name;unit_name;processcell_name;area_name`

Note: This simplified format is only available for import. The format for exporting the objects of the Physical Model hierarchy is described in *Physical Model Elements: CSV File Format*, p. 370.

Separator

The separator must not necessarily be a semicolon. The separator character is set in the **View** → **Options** dialog in Unity Application Generator.

Note: Do not use a separator character that may be used in the fields.

Options

You have the option to either import Instruments without the hierarchy in the Physical Model or with the complete hierarchy up to the area level:

If you want to...	Then...
Import only Instruments (without hierarchy in the Physical Model)	Omit the last four entries <i>equipmentmodule_name;</i> <i>unit_name;processcell_name;</i> <i>area_name.</i>
Import Instruments as Control Modules with complete hierarchy in the Physical Model	Specify all the four entries <i>equipmentmodule_name;</i> <i>unit_name;processcell_name;</i> <i>area_name.</i>

Description of Entries

The entries in one line of the CSV file for importing Instruments have the following meaning. Required fields are marked with a star *. If not required field are not specified, they remain empty (;;).

Entry	Description and Check during Import	Example
* INSTRUMENT	The first entry must be exactly the string INSTRUMENT to identify the contents of the line.	
* <i>controlmodule_name</i>	This entry contains the name of the Control Module to be created. The name is checked to be compliant with the naming convention used in the project customization. If the Control Module should be created in an Equipment Module, the uniqueness of the Control Module name in the scope of the parent Equipment Module is checked.	motor1
* <i>controlmodule_type</i>	This element contains the name of the Control Module Type of the Control Module to be created. The name is checked to be exactly one of the Control Module Types in the Control Module Type library specified in <i>controlmodule_type_library</i> .	MSR01
* <i>library_name</i>	This entry contains the name of the Control Module Type library, which contains the specified Control Module Type. The library is checked to be contained in the project customization.	IATBasic10
* <i>controlmodule_description</i>	If the text is longer than the allowed description length, the text is truncated to the maximum length.	Motor for conveyer No. 1

Entry	Description and Check during Import	Example
controlmodule_ comment	This entry contains the comment of the Control Module to be created. If the text is longer than the allowed comment length, the text is truncated to the maximum length. Note: The special character sequence \n can be used to specify line feeds as part of the comment. The special characters sequence \t can be used to specify tabs as part of the comment.	Motor for conveyer No.1\nThe motor can be operated in manual or auto mode
* equipmentmodule_ name	These four optional entries specify the hierarchy in the Physical Model where the Control Module should be created. If they are omitted , the Control Module is created in the Instrument List and not within an equipment in the Physical Model. If they are specified , <ul style="list-style-type: none"> ● the objects will be created if they are not yet existing and ● the Control Module will be created in the Physical Model hierarchy 	Equipment1
unit_name		Unit1
processcell_name		ProcessCell1
area_name		Area1

Errors

During the import process Unity Application Generator checks the entries for uniqueness of names and for compliance with the customization. If during these checks an error occurs the processing of the line is aborted and the rest of the line is ignored. An error message is displayed in the message window.

Example: Import Initial Values for Variables

Introduction

By simply importing a CSV file it is possible to change the initial values of existing variables. Often PID parameters have to be tuned during commissioning time to optimize the loops. To restart the process with the optimized loop parameters, these values have to be stored as the initial value of the appropriate variables. With the import functionality of the initial values this requirement can be fulfilled in an easy-to-use way.

Line in the CSV File

For importing initial values for variables each line in the CSV file has the following entries:

`INITIALVALUE;variable_full_name;initial_value`

Example: Initial Values in a CSV File

A CSV file for importing initial values for variables looks for example like this:

	1	2	3
Line 1	IN	ITIALVALUE;variable_name1;	initial_value1
Line 2	IN	ITIALVALUE;variable_name2;	initial_value2
Line 3	IN	ITIALVALUE;variable_name3;	initial_value3
Line 4	IN	ITIALVALUE;variable_name4;	initial_value4
Line 5	IN	ITIALVALUE;variable_name5;	initial_value5
Line 6	IN	ITIALVALUE;variable_name6;	initial_value6

Note: The separator must not necessarily be a semicolon. The separator character is set in the **View** → **Options** dialog in Unity Application Generator for correct reading of the CSV file.

Description of Entries

The entries in one line of the CSV file for importing initial values have the following meaning:

Entry	Description and Check during Import	Example
INITIALVALUE	The first element must be exactly the string INITIALVALUE to identify the contents of the line.	
<i>variable_full_name</i>	This entry contains the full name of the variable for which the initial value should be changed. The name is checked to be the name of an existing variable.	Boiler1_Motor1_FTR
<i>initial_value</i>	This entry contains the initial value of the variable. The initial value is checked to be valid according to the data type of the variable.	3.1415

Errors

During the import process Unity Application Generator checks the entries as described in the above table. If during these checks an error occurs the processing of the line is aborted and the rest of the line is ignored. An error message is displayed in the message window.

Supported HMIs and their Setup

9

Overview

Introduction

Unity Application Generator is able to generate tags and graphics for Monitor Pro or Intellution's iFIX or to create a XML file with tag information that may be used by any HMI (generic HMI). This chapter gives you overview information about the specific combination HMI - Unity Application Generator and some specific instructions for the correct configuration of the HMI.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
9.1	Monitor Pro and Unity Application Generator	195
9.2	iFIX and Unity Application Generator	226
9.3	Generic HMI and Unity Application Generator	239

9.1 Monitor Pro and Unity Application Generator

Overview

Introduction

Unity Application Generator supports Monitor Pro as HMI . For Monitor Pro Unity Application Generator generates all databases and pictures necessary for the process visualization.

The Monitor Pro application is generated on the engineering PC and then the generated objects must be transferred to the production PCs. This chapter describes in detail the procedures from the preparation of the Monitor Pro generation to the running of an existing application.

What's in this Section?

This section contains the following topics:

Topic	Page
Introduction	196
Monitor Pro - Default Server Application	197
Generation Modes of Monitor Pro Server Application	200
Data Conversion Monitor Pro vs. Unity Application Generator	202
Monitor Pro Drivers and Communication	206
Alarming	210
Archiving	215
Monitor Pro Client Application	221

Introduction

Monitor Pro Server Application

To be able to generate a Monitor Pro Server Application, Monitor Pro has to be installed already. There are no special settings required for UAG during Monitor Pro installation.

Terminology

The following table explains the corresponding terms in Unity Application Generator and in Monitor Pro:

Term in Unity Application Generator	Corresponds to the term in Monitor Pro
Data Server	SCADA node
HMI	View node
Data Servers and HMIs (all nodes of HMI system)	Monitor Pro nodes

Monitor Pro - Default Server Application

Introduction

The code generation for SCADA systems creates a Monitor Pro server application for each data server defined within the UAG project. A Monitor Pro server application consists of different folders and files.

The user has to specify a `Data Server Path` for the server application. UAG will create a default server application in this path. The default server application will be installed with UAG in the installation directory `\MonitorPro\Server\Standard.zip`. The default application includes a set of default mailbox tags, trigger tags, settings, etc.

Note: If the user wants to enhance the default server application, it is possible to unzip the file, modify it and save it back as a `Standard.zip`.

Setup of the Server Application Generation

A Monitor Pro server application corresponds to an UAG data server. Several data servers can be defined. Each data server will be deployed as a different Monitor Pro server application node.

Data Server Path

To be able to generate a dataserver, it is necessary to specify a `Data Server Path` in the `Options` of Unity Application Generator. UAG automatically creates a sub directory with the name of the dataserver. Thus the generated server application is located in `OPTIONS_PATH\DATASERVER_NAME`. Each dataserver has its own directory structure.

Mailboxes

Mailbox is a unique data type that specific tasks use to communicate with each other.

Tags

A tag is a data element stored in the real-time database which is assigned to a logical name. This tag name is used by Monitor Pro tasks to reference the element in the real-time database.

Default Server Application

The default application Standard includes:

- Data logging setup with 4 Mailboxes for:
 - ALLOG_HIST_MBX
Used for alarm logging
 - DBLCHISTMBX
Used for database logging
 - DPLCHISTMBX
Used for datapoint logging
 - SECURITYMBX
Used for security logging

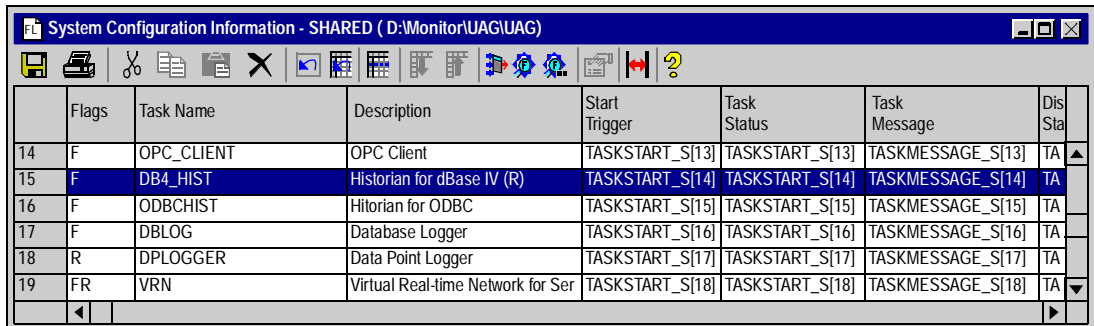
These mailboxes exist in the following configuration tables:

- Historian Mailbox Information for dBase IV
- Historian Mailbox Information for ODBC

Monitor Task

It is necessary to set the Run flag in the System Configuration table, depending on the database system the user wants to use. The UAG user can choose between two database systems, dBase IV and ODBC.

The following figure shows the System Configuration table:

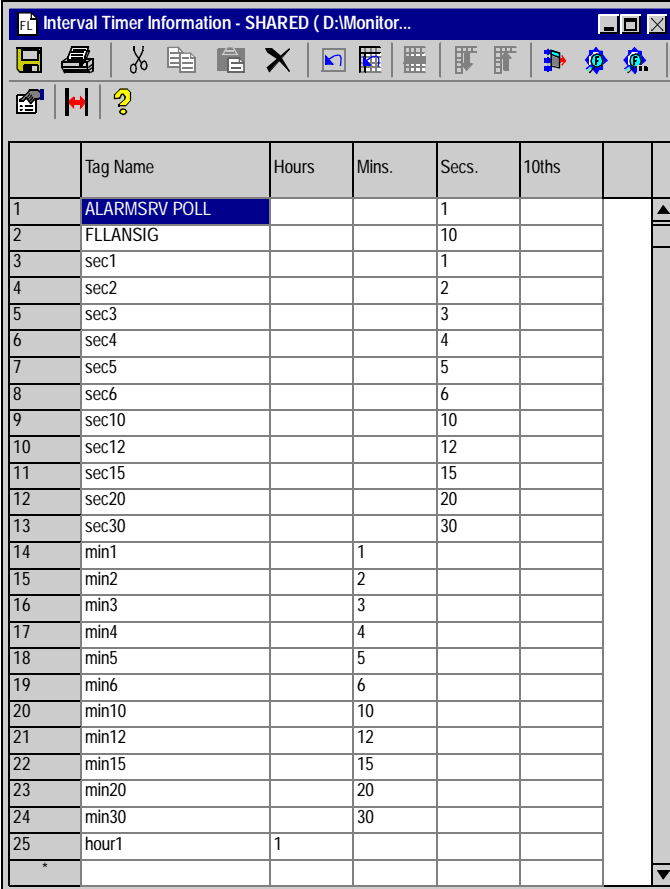


The screenshot shows a window titled "System Configuration Information - SHARED (D:\Monitor\UAG\UAG)". The window contains a table with the following data:

	Flags	Task Name	Description	Start Trigger	Task Status	Task Message	Dis Sta
14	F	OPC_CLIENT	OPC Client	TASKSTART_S[13]	TASKSTART_S[13]	TASKMESSAGE_S[13]	TA ▲
15	F	DB4_HIST	Historian for dBase IV (R)	TASKSTART_S[14]	TASKSTART_S[14]	TASKMESSAGE_S[14]	TA
16	F	ODBCHIST	Historian for ODBC	TASKSTART_S[15]	TASKSTART_S[15]	TASKMESSAGE_S[15]	TA
17	F	DBLOG	Database Logger	TASKSTART_S[16]	TASKSTART_S[16]	TASKMESSAGE_S[16]	TA
18	R	DPLOGGER	Data Point Logger	TASKSTART_S[17]	TASKSTART_S[17]	TASKMESSAGE_S[17]	TA
19	FR	VRN	Virtual Real-time Network for Ser	TASKSTART_S[18]	TASKSTART_S[18]	TASKMESSAGE_S[18]	TA ▼

Trigger Tags

Some trigger tags are created in the Default Server Application. The following figure shows the list of available interval timers:



	Tag Name	Hours	Mins.	Secs.	10ths
1	ALARMSRV POLL			1	
2	FLLANSIG			10	
3	sec1			1	
4	sec2			2	
5	sec3			3	
6	sec4			4	
7	sec5			5	
8	sec6			6	
9	sec10			10	
10	sec12			12	
11	sec15			15	
12	sec20			20	
13	sec30			30	
14	min1		1		
15	min2		2		
16	min3		3		
17	min4		4		
18	min5		5		
19	min6		6		
20	min10		10		
21	min12		12		
22	min15		15		
23	min20		20		
24	min30		30		
25	hour1	1			
*					

Communication Mailboxes

Additionally some default communication mailboxes are created in the default server application.

One mailbox is created to exchange data between the driver and the `Ioxlator` task. This mailbox is called `MODTIOXMbx`. Another mailbox is created to exchange the data with the PLC. This mailbox is called `MODTDrvmbx`. The `MODTDrvmbx` is used in the driver task and therefore it is created in the Modbus TCP/IP driver mailbox Definition table. The `MODTIOXMbx` mailbox is used in the `Ioxlator` task and therefore it is created in the I/O Translator Mailbox Definition table.

Generation Modes of Monitor Pro Server Application

Introduction

There are two modes for the generation of the Monitor Pro server application

Note: During the generation, all Monitor Pro applications have to be stopped.

Mode: New Generation

A new generation rebuilds the entire Monitor Pro applications. After confirmation, the old applications will be renamed to `DATASERVER_NAME.BAK`.

If Unity Application Generator generates a data server the first time, it makes sure that there is no existing Monitor Pro server application with the same name. If there is no subdirectory with the name of the data server, UAG creates it and copies the default server application to this directory.

Mode: Incremental Generation

Incremental or relative generation will be used to synchronize the data of UAG and the data of the Monitor Pro server application. A relative generation takes the existing Monitor Pro server applications into account. Since UAG does not generate the complete data for the Monitor Pro application, the programmer has to add data manually. UAG does not force the user to adopt a one way strategy. This strategy makes it possible to modify the process design after having modified the generated data.

**Rules for
Incremental
Generation**

Nevertheless, some rules have to be obeyed by the designer and the Monitor Pro programmer.

- Do not manually create Data Server application that will be generated by means of UAG.
- Structured variables, which can not be managed by UAG, have to be declared in the reserved memory range of the PLC in UAG. The overall memory layout belongs to the PLC. The PLC memory is segmented into different parts, HW Modules, HMI, Net Partner, PLC Communication and Reserve.
- Do not manually add or modify variables in the Ioxlator datasets generated by UAG.
- For additional logic (e.g. IML), internal variables can be created in Monitor Pro. The programmer of this variables in Monitor Pro has to ensure that the variable name is unique.

The relative generation will not alter the manual enhancements in the server application, as long as they are consistent with the process design. When starting the incremental generation, UAG checks the timestamp written to the Monitor Pro project during the last generation and compares it with the UAG timestamp for this data server. If no timestamp exists in the Monitor Pro project or the timestamps have different values, UAG will ask the user how to continue. The user can select to either start a new generation of this data server or to stop the generation of this data server.

**Generation and
Monitor Pro**

If a Monitor Pro application has been generated, this application will not be intergrated within Monitor Pro directly. It has to be added within the `Configuration Explorer` manually on the target machine.

Data Conversion Monitor Pro vs. Unity Application Generator

Data Conversion Types

The data exchange variables for the communication with the HMI system are categorized according to their communication frames. There are four different communication frame types and one combined frame (`ReadWrite`) within Monitor Pro.

The following table shows the Communication Frame Types available in UAG.

Frame Type	Monitor Pro, Generic HMI	iFix, PCVue
Read Data (RD)	x	
Exception Write (EW)	x	
Block Write (BW)	x	
Unsolicited Read(UR)	x	
ReadWrite (RD & EN)	x	x

The only valid combination of different frame types is RD & EW and has the meaning of a read / write frame.

Available UAG data types will be converted to appropriate Monitor Pro data types by the specific configuration of I/O tasks in Monitor Pro. This configuration will be generated by UAG.

The following data types can be assigned to a communication frame.

Monitor Pro data type	Equivalent data type in UAG	UAG frame data type	Monitor Pro data set definition type
Digital	BOOL	BOOL	COIL
Analog	BYTE, INT, WORD, UINT	WORD	HREG
Long	DINT, UDINT	DINT	HREG
Float	REAL	REAL	HREG

The user can select only one data type per frame in UAG, e.g. `BOOL`, `WORD`, `DINT`, `REAL`. The read or write trigger of a specific frame has to be adjusted in Monitor Pro. UAG will not modify these triggers in the communication setup. Analog, Long, Float data types will be generated as `HREG`, Digital will be generated as `COIL`.

For example, a variable of the datatype `BOOL` can be assigned only to a frame with the datatype `BOOL`. UAG creates for this communication frame a dataset within the device driver and a dataset in the I/O Translator Dataset Definition. The variable will be created as `Digital` in Monitor Pro.

**Communication
Frame Example**

The following table shows an example of communication frames.

Frame Name	UAG Frame Data	Frame Type
ReadFast	BOOL	Read Data
WritSlow	WORD	Write Data
WritFast	REAL	Write Data
...	x	

The communication frames can be defined in the Customization and the SCoD Editor. The following properties are required for frame type definition:

- Frame Name (max. 8 characters because of extension `_Number`. `_Number` is a unique number which will be created by UAG. The user must not change this number.)
- Frame Data Type
- Frame Type

If a PLC has a channel to a data server, all variables assigned to a communication frame will be generated in the Monitor Pro server application. UAG imports all defined frame types from the customization into the project, when a channel to data server is created. If there is a new frame type definition in the SCoD or the customization, UAG will update the frame definitions of the project with the new frame. Used frames can not be deleted or modified in the Customization Editor. If the user opens the HMI Communication dialog in UAG, all frame definitions are shown.

The following figure shows an example for the HMI communication.

Unity Application Generator - gbeck [Exclusive] - [HMI Communication [channel01]]

File View Generate Window Help

Data Server: moni01 Address Range (%M): 1025 - 2048, Length: 1024

PLC: plc01 Address Range (%MW): 1025 - 2048, Length: 1024

Name	Data Type	Frametype	Type	Address	Length	End Address	Used	% Used
▶ SynBOOL	BOOL	Read & Write (RD & EW)	%M	1025	259	1283	8	3
ParBOOL	BOOL	Read & Write (RD & EW)	%M	1284	251	1534	0	0
EvIBOOL	BOOL	Read & Write (RD & EW)	%M	1535	251	1785	0	0
ReadBOOL	BOOL	Read Data (RD)	%M	1786	263	2048	12	5
SynDINT	DINT	Read & Write (RD & EW)	%M	1026	86	1111	3	7
ParDINT	DINT	Read & Write (RD & EW)	%M	1112	84	1195	0	0
EvIDINT	DINT	Read & Write (RD & EW)	%M	1196	84	1279	0	0
ReadDINT	DINT	Read Data (RD)	%M	1280	86	1365	3	7
SynREAL	REAL	Read & Write (RD & EW)	%M	1366	84	1449	0	0
ParREAL	REAL	Read & Write (RD & EW)	%M	1450	84	1533	0	0
EvIREAL	REAL	Read & Write (RD & EW)	%M	1534	84	1617	0	0
ReadReal	REAL	Read Data (RD)	%M	1618	84	1701	0	0
SynWORD	WORD	Read & Write (RD & EW)	%M	1702	84	1785	0	0
ParWORD	WORD	Read & Write (RD & EW)	%M	1786	84	1869	0	0
EvWORD	WORD	Read & Write (RD & EW)	%M	1870	84	1953	0	0
ReadWord	WORD	Read Data (RD)	%M	1954	84	2037	0	0

plc01 | plc01 - Unity | | | 27.08.2004 | 10:13

The user has to enter the address ranges for each used frame. He has two different possibilities to define the ranges.

- Automatic

The **Default** button in the **HMI Communication** grid calculates the start addresses and lengths of all communication frames. This calculation is based on the available address range and the used memory. After the calculation the unused memory is evenly distributed to all communication frames. The previous values will be irreversibly overwritten.

The user has to be aware, that even small changes in the used memory (e.g. by adding a control module) can lead to a reassignment of several communication frames.

- Manual

The user has to enter the addresses manually. The address has to fit to the end address which is defined in the PLC properties and is shown in the header of the dialog.

Variable Communication Frame Assignment Each `PLC_HMI` variable has to be assigned to a `Communication Frame`. This is possible in the SCoD Editor, which means on the type definition. In UAG the communication frame of a variable can be modified. All the variables of a PLC, assigned to the same `Communication Frame` in UAG are packed for optimization without gaps in the corresponding data set in Monitor Pro.

Alarm Group and Archive Name

Note: Communication Frames, Alarm Groups and Archive Names are only deletable if they are not used.

Monitor Pro Drivers and Communication

Introduction

The adjustments of the Monitor Pro SCADA system are very complex. Unity Application Generator sets all these adjustments automatically. The following information gives an overview about the adjustments made by UAG within the Monitor Pro communication.

Communication in Monitor Pro

The Monitor Pro server application database tags are updated through Ethernet (TCP/IP) drivers with the actual values of the PLC variables. IOX/RAPD driver tasks are used between SCADAs and the PLCs.

Modbus TCP/IP Driver Device Definition

Each PLC with a channel to a data server gets the following entries in the Monitor Pro server application. UAG will generate one Modbus TCP/IP Driver definition per PLC. The driver definition includes the following items.

- PLC name (max. 16 character)
- IP network address (address of used communication module in channel path)
- Connection timeout (in seconds)

Note: All fields which are not mentioned here will not be filled or modified by the Unity Application Generator.

Frame Type Names

The frame type names consists of different parts. The constituent parts are: 8 characters + _XXXX + YYY, for example `readfast_0010002` - frame type name `readfast` with the PLC Id 10 and the frame number 2

The following table explains the different parts of the type frame name.

Part	Description
Name	Frame type name. Maximum 8 characters, specified in SCoD or Customization Editor.
XXXX	PLC identification. A number from 1 to 9999 generated by UAG.
YYY	Frame number. A number from 1 to 999 generated by UAG.

Note: The PLC Id and the frame number must not be changed by the user. They will be generated and administered by Unity Application Generator.

Modbus TCP/IP Driver Dataset Definition

For each frame type there will be at least one data set definition in the driver definition. If the number of variables assigned to one frame type exceeds the maximum frame length, UAG will automatically generate additional frames with an incremental extension number. E.g. the first frame of the frame type = `ReadFast` (PLC Id = 1) will be generated as data set `ReadFast_0001001`, the second of type `ReadFast` as `ReadFast_0001002` etc. UAG creates more than one frame at the same time in Monitor Pro when the length exceeds 100 words. The following table shows the Modbus TCP/IP driver Dataset Definition.

Field	Content generated by UAG
Mailbox Tag	MODTIOMbx
Data Set	Frame Name + <code>_FrameNumber</code>
Type	HREG (no booleans) or COIL (booleans)
Start	Frame start address
Len	Length of the frame (max. 100 for HREG and 1600 for COIL)

Note: All fields which are not mentioned here will not be filled or modified by the Unity Application Generator.

I/O Translator Protocol Driver Definition

In addition each PLC gets one I/O translator protocol driver definition. The I/O translator driver definition includes the following items.

- Protocol Driver Name (PLC name)
- Protocol Driver Mailbox Tag
- Max MSG in PD (Protocol Driver)

Note: All fields which are not mentioned here will not be filled or modified by the Unity Application Generator.

I/O Translator Dataset Definition

For each frame type there will be at least one data set definition in the I/O translator configuration, according to the data sets in the driver definition shown above. The following table shows the I/O Translator Dataset Definition.

Field	Content generated by UAG
Data Set Name	Frame Name + <code>_FrameNumber</code>
Data Set Control Tag	Frame Name + <code>_FrameNumber</code>
Read Trigger	<code>IOXRDTTrigger</code>
RD	Read Data Depends on frame communication type
EW	Exception Write Depends on frame communication type
BW	Block Write Depends on frame communication type
UR	Unsolicited Read Depends on frame communication type
Abs	<code>Yes</code>

The user can adjust the read trigger to have optional read/write rate. The parameter will not be regenerated by UAG.

Note: All fields which are not mentioned here will not be filled or modified by the Unity Application Generator.

Ioxlator Tag Definition

UAG will generate for each I/O Translator Dataset Definition the appropriate variables.

The following table shows the I/O Translator Tag Definition.

Field	Content generated by UAG
Data Tag	Variable Name
Dimension	1
Address	Absolute variable address
Conv	Conversion type, according to the following table

Note: All fields which are not mentioned here will not be filled or modified by the Unity Application Generator.

The following table shows the I/O Translator conversions for UAG data types.

UAG data type	I/O Translator conversion
Bool	Bit
BYTE, INT, WORD, UINT	Word
DINT, UDINT, RLNG	Long
REAL	RFLT

Alarming

Introduction

The alarm logger task in Monitor Pro is responsible for monitoring specified tags to check whether their values fulfill specified alarm conditions. If a tag is recognized to be in an alarm state, an alarm record is created and passed to a historian task, which writes it to a relational database. If a tag value does no longer meet the alarm criteria, the alarm logger creates an additional record. Alarm viewer controls in client applications communicate with the alarm server allowing the user to browse and acknowledge the alarms.

Generation of the Alarming Configuration

UAG configures the alarm logger, its connection to both the dBase4 historian and the ODBC historian tasks, and the alarm server. The user is responsible for running only one historian task in his application.

The Alarm Logger Setup tables will only be created when UAG creates a new Monitor Pro application, these tables (and others) are part of the UAG Standard server application, which will be copied when creating a new Monitor Pro application.

The Alarm Group Control table of the alarm logger in Monitor Pro will be filled with the names of the corresponding Alarm Group of UAG. It is possible to create Alarm Groups in the SCoD / Customization editor. These Alarm Group can be assigned to a variable on Type Level (SCoD) or instance level (UAG). If a variable is set to Alarm Yes the Alarm Group is not empty the following tables will be generated into Monitor Pro application.

The following table shows the Alarm Group Control Table.

Field	Content generated by UAG
Group Name	Alarm Group Name
Acknowledgement	YES
Audible	YES
Logging	YES

The Alarm Definition Information table of an alarm group in Monitor Pro will be filled with entries for all alarm variables belonging to the corresponding Alarm Group. The following table shows the Alarm Group Control Table.

Field	Content generated by UAG
Unique Alarm ID	Unique number corresponding to the alarm definition.
Cond.	Alarm condition property of variable.
Limit	Alarm limit property of variable (i.e. a constant value, not a tag).
Deadband	-
Message Text	Alarm text property of variable.
Priority	Alarm priority property of variable.
Area Name	Equipment name of variable (i.e. a picture with thisname will be generated).
Use Global Hide	YES

Note: In case of manual addition of the alarm entries directly in Monitor Pro the user has to check the uniqueness of the alarm IDs.

Configuration of the Alarm in UAG

UAG will allow the configuration of the alarm state for both boolean and non-boolean variables.

Unity Application Generator variable objects will expose two properties for alarms. The configuration of the alarms are the same than the configuration of the archive data.

The following list shows the two different alarming properties.

- Alarm
It can store values for `Not_Assigned`, `Yes` or `No`
- Alarm Group Name
It stores a string that will be used as the name of the database table in Monitor Pro. With this property the user is able to group the alarm variables, which will handle the same alarm and which will use the same trigger.

The following figure shows the UAG Alarm Properties for a `BOOL` variable.

The screenshot shows the 'Variable Properties' dialog box for the variable 'AHHM'. The 'Alarm' tab is selected, displaying configuration options for the variable's alarm behavior. The 'Selected Object' field contains 'AHHM'. The 'Alarm' section is expanded, showing settings for the main alarm and three specific alarm instances (Alarm 1, Alarm 2, and Alarm 3).

Variable Properties

Selected Object:

General | **Value** | **Alarm** | **Command** | **Archive** | **Display**

Alarm:

Visible Changeable

Inheritance:

Show List Show Details

Alarm Group:

Visible Changeable

Inheritance:

Select Alarm:

Alarm 1 - Priority:

Visible Changeable

Inheritance:

Alarm 1 - Text:

Visible Changeable

Inheritance:

Alarm 1 - Operator:

Visible Changeable

Inheritance:

Alarm 1 - Limit:

Visible Changeable

Inheritance:

The variable properties can be set and used in SCoD editor as with other variable properties and can also be viewed and/or modified in UAG.

The following table shows the attributes of the alarm for the datatype `BOOL`.

Number of alarms	1
Alarm Group	The alarm can be set to an Alarm Group to combined alarms.
Alarm Priority	The Alarm Priority will be preset within the SCoD Editor or the Customization. For example: 1 - low 2 - medium 3 - high 4 - urgent
Alarm Text	default or user-defined
Operator	=
Alarm Limit	Not_Assigned 0 - OFF 1 - ON

The following table shows the attributes of the alarms for other possible datatypes.

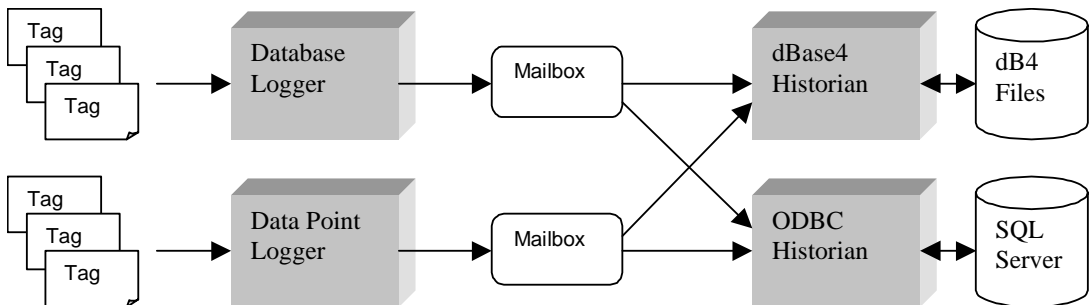
Number of alarms	1 to 8
Alarm Group	The alarm can be set to an Alarm Group to combined alarms.
Alarm Priority	The Alarm Priority will be preset within the SCoD Editor or the Customization. For example: 1 - low 2 - medium 3 - high 4 - urgent
Alarm Text	default or user-defined
Operator	Not_Assigned =, <>, >, <, >=, <=
Alarm Limit	Number for the selected alarm.

Note: All alarms of a non-boolean variable must be in the same Alarm Group.

Archiving

Introduction

In Monitor Pro there are several tasks that can perform archiving (logger tasks). They read values of specified tags from the real-time database and pass the values or derived data through to one of the several other tasks (historian tasks). The historian tasks perform saving of the values in a relational database according to specified database schemes. For archiving real-time values there are two logger tasks, the Database Logger and the Data Point Logger. For the historian tasks we consider here only the two probably most commonly used tasks, the dBase4 Historian and the ODBC Historian. Only one of the four possible combinations of logger and historian tasks should be used in one server application. The following figure shows the data structure for logging/archiving.



Database Logger vs. Data Point Logger

One of the main differences between Database Logger and Data Point Logger is the database schema for storing the values. The Database Logger uses database tables where up to 1100 tag values can be stored in a row together with a time stamp, whereas the Data Point Logger uses database tables where only one tag value can be stored in a row together with a time stamp. The Database Logger saves space and is more efficient, whereas the Data Point Logger is easier to use, especially when tags have to be added or removed from logging. With the Database Logger, adding or removing tags from logging could lead to a change of the database schema and this means to change an already existing database table manually in the database system.

The following table shows the different properties of the available Logger functions.

Logger type	Stored values
Database Logger	up to 1,024 tags in one row
Data Point Logger	1 tag in one row

Generating the Archiving Configuration

UAG delivers together with the standard server application the pre-configured mailboxes for both `DB Logger` and `DP Logger` that are also introduced into the configuration for `dBaseIV` and `ODBC` historians. However the user has to configure the chosen historian information table on his own using the database names generated by UAG for corresponding `DB` or `DP Loggers`. Both historian tasks will be entered in the system configuration table, but the user has to also here activate one of the historian tasks manually.

Note: The user must not delete or modify the pre-configured by UAG historian mailboxes (`DBLCHISTMBX` and `DPLCHISTMBX`).

Note: It is not possible to activate more than one historian tasks.

UAG configures both Database Logger and Data Point Logger to allow the user to choose one of them to run in his application. UAG creates schema information and logger control information for both logger tasks, but the user is responsible for starting one of these tasks.

Note: The user has to be aware that the Database Logger can be used with the `dBase4` Historian only if the length of the column names (i.e. tag names) in the database schema does not exceed 8 characters.

Since UAG does not know, which combination of tasks the user prefers, UAG can only issue a warning message to the user, if the schema columns are longer than 8 characters.

For Database Logger, when a tag is added or removed from logging, UAG adds or removes the corresponding column to/from the database schema and issues a warning message to the user that he must change an existing database manually before (re-)starting the application. If the database is not existing, Monitor Pro will create it, using the new schema when the first value will be stored.

UAG creates tags for triggering both Database and Data Point Loggers actions. The user if required can enter them into the Interval timer or Event timer information tables accordingly to the chosen logging strategy.

Database Logger Configuration The Schema Control table of the database logger in Monitor Pro will be filled with the archive names defined in UAG appended with the number like _001 if required.

Note: All fields which are not mentioned here will not be filled by the Unity Application Generator.

The following table shows the Schema Control Table.

Field	Content generated by UAG
Schema Name	Archive name (in upper case) if necessary appended with _001, _002 etc. This can be necessary if more than 1100 variables are assigned to a schema.

The first child table of the Schema Control table is the Index Information table. It will be filled with the same content for all schemas.

Note: All fields which are not mentioned here will not be filled by the Unity Application Generator.

The following table shows the Index information Table.

Field	Content generated by UAG
Index Number	1
Unique Index	YES
Column List	FLTIME

The second child table of the Schema Control table is the Schema Information table. It will be filled with the names of all variables that are assigned to the archive, which is represented by the current schema.

Note: All fields which are not mentioned here will not be filled by the Unity Application Generator.

The following table shows the Schema Information Table.

Field	Content generated by UAG
Column Name	FLTIME for the first column, the variable name otherwise.
Column Usage	TIME for the first column, DATA otherwise.
Column Type	" " for the first column, either INTEGER, SMALLINT, NUMBER or FLOAT according to the variable's data type otherwise.

The Database Logging Control table of the database logger in Monitor Pro will be filled with the archive names defined in UAG.

Note: All fields which are not mentioned here will not be filled by the Unity Application Generator.

The following table shows the Schema Information Table.

Field	Content generated by UAG
Log Name	Archive name, if necessary appended with _001, _002 etc.
Log Trigger	Log name appended with _TRIG
Historian Mailbox	DBLCHISTMBX
Database Alias Name	Archive location as defined within Customization Editor.
Database Table Name	Same as Log Name
Schema Name	Archive name if necessary appended with _001, _002 etc

The Database Logging Information table of the database logger in Monitor Pro will be filled with the tags defined within UAG.

The following table shows the Database Logging Information table entries generated by UAG.

Field	Content generated by UAG
Tag Name	<ul style="list-style-type: none"> ● First name has to be empty (recommendation of Monitor Pro). ● Following names will be the tag names.
Column Name	FLTIME for the first tag and variable name (schema columns are named with variable names for uniqueness purposes).
Logon Change	No
Column Usage	<ul style="list-style-type: none"> ● If the Tag Name is empty and the Column Name is FLTIME the Column Usage is TIME. ● If the Tag Name is unequal empty the Column Usage is DATA.

Data Point Logger Configuration

The Data Point Schema Control table of the Data Point Logger in Monitor Pro will be filled with only one schema definition for all archives defined in UAG.

Note: All fields which are not mentioned here will not be filled by the Unity Application Generator.

The following table shows the Data Point Schema Control Table.

Field	Content generated by UAG
Schema Name	TAGDATA
Logged Value Data Type	FLOAT

The Data Point Logger Control table will be filled with the archive names defined in UAG.

Note: All fields which are not mentioned here will not be filled by the Unity Application Generator.

The following table shows the Data Point Logger Control Table.

Field	Content generated by UAG
Table Name	Archive name, if necessary appended with _001, _002 etc.
Schema Name	TAGDATA
Database Alias Name	Archive location as defined within Customization Editor.
Historian Mailbox	DPLOGHISTMBX

The Data Point Logger Information table will be filled with the names of all variables which are assigned to the archive, which is represented by the current Data Point Logger Control table.

Note: All fields which are not mentioned here will not be filled by the Unity Application Generator.

The following table shows the Data Point Logger Information Table.

Field	Content generated by UAG
Log Tag	Variable name
Log On Change	No
Log Trigger	Log name appended with <code>_TRIG</code>

The Customization editor imports the list of archive names from the SCoD library, but the user can add additional archive names.

The user has to specify the additional Archive parameter in Customization Editor, the archive location. This attribute corresponds in the Monitor Pro application to the database name alias configured in the historian configuration table.

Archive Configuration

Unity Application Generator variable objects will expose two properties for archiving. The following list shows the two different archiving properties.

- `Archive`
It can store values for `Not Assigned`, `No` or `Historic`
- `Archive Name`
It stores a string that will be used as the name of the database table in Monitor Pro. With this property the user is able to group the variables, which will be stored within the same archive and which will use the same trigger.

If there are too much variables for the same database table, UAG will create additional table names automatically (e.g. by appending a number to the name specified by the user) rather than limiting the user when he assigns archive names to variables. Since the length of the schema name in Monitor Pro is restricted, the UAG archive name should be restricted to 8 characters.

Both variable properties can be set within the SCoD editor (if Monitor Pro is assigned to the list of HMIs of the library) and the properties can be set to be changeable or invisible for UAG. In the SCoD editor the user has to define any value of the `Archive Name` property in a list (similar to access levels) prior to its use in a variable's property. Both archive-related properties can be used in user-defined SCoD properties and with inheritance.

The Customization editor imports the list of archive names from the SCoD library, but the user can add additional archive names. The user can only modify or delete archive names if they are not used in the UAG project. This will be checked automatically by the Customization editor.

In UAG, the user can modify the values for both properties if they were set to be changeable in the SCoD editor. The value of the archive name property can be selected from the list of names defined in the customization. If a variable with the archive property is used for a HMI other than Monitor Pro, the archive settings have no effect.

Monitor Pro Client Application

Introduction The HMI generation part includes the mimic and the symbol generation for the Monitor Pro Client Builder. For this the file `Client.zip` is used by UAG.

Client.zip The file `Client.zip` will be installed within the Unity Application Generator installation. The installation path is: `..\Unity Application Generator\MonitorPro\Client\Client.zip`. This file contains every symbol for the integrated SCoDs of UAG. The user has to set the adjustment for the OPC Cluster just once. After this UAG is generating the complete Monitor Pro Client Builder directory structure for the complete project.

Steps for Client.zip The user has to do the following steps.

Step	Action
1	Expand the file <code>Client.zip</code> with the whole directory structure.
2	Open the project <code>client.fvp</code> with the Monitor Pro Client Builder application..
3	Set the adjustment for the OPC Cluster. <ul style="list-style-type: none"> ● Tools → Servers → Cluster → OPC Cluster Change the OPC Cluster name to the Data Server name which is used within the UAG project.
4	Create a new file <code>Client.zip</code> with the whole directory structure.
5	Copy the file <code>Client.zip</code> into the client directory (<code>..\Unity Application Generator\MonitorPro\Client\Client.zip</code>).

HMI Path To be able to generate the HMI mimics, it is necessary to specify a HMI path within the `Options` of UAG. UAG automatically creates a subdirectory with the name of the HMI. Thus the generated HMI application is located in the directory `HMI_PATH\HMI_NAME`.

Generation of the Monitor Pro Client Builder mimics

UAG generates all available Client Builder mimics and symbols which are used within the SCoDs or free variables. For this UAG has a task list for the elements which will be generated new or incremental. This list is available within UAG under **File → Properties → Objects → to be generated → HMI.**

The generation for HMI consists of creating a set of Client Builder mimics. The mimic files have no extensions. They are located in `HMI_PATH\HMI_NAME\Mimic Files`.

The generator creates a mimic for each UAG Equipment Module that has been assigned to a Control Domain and to a PLC. The name of the mimic is the name of the Equipment Module. For each Control Module of that Equipment Module a SCoD symbol is generated in the Client Builder mimic. The name of the HMI symbol has to be specified in the SCoD library. For each free variable a symbol can be generated. The user has to select this within the user interface of the variable.

If UAG detects a modification of a Control Module or a Control Module free variable, all corresponding Control Module properties (Name, Data Server Name) will be set with the actual values from the UAG database.

At run time, the symbol gets the real-time information from the Monitor Pro server runtime database. This is achieved by using the OPC Server.

Client Builder Symbol and Mimic Format

The symbols and mimics for Monitor Pro are generated in ASCII format. The format is explained in the document "Client Builder ASCII Mimic guide" that belongs to the Monitor Pro documentation.

The following programming code shows the ASCII format of a Control Module (symbol) generated by UAG for Monitor Pro Client Builder.

```
O,BEGIN,S," "
  B,25,25,0,0,0,0,1,0,6400,0,1
  PP,"valve_en","monitor1:gerequip_VAS01",25,25,0,0,1,0,,,,,,,,,
O,END
```

The following programming code shows the ASCII format of a free variable generated by UAG for Monitor Pro Client Builder.

```
O,BEGIN,S," "
  B,25,25,0,0,0,0,1,0,6400,0,1
  PP,"symb","monitor1:gerequip_con01",25,25,0,0,1,0
  SUB,"@DIGI","monitor1:@gerequip_con01"
O,END
```

Monitor Pro supports branching. Thus it is possible to generate the Symbol Name (e.g. Symbol6), the Branch Name (e.g. monitor1:gerequip_VAS01) and the Variable Name (e.g. monitor1:@gerequip_con01) to define the complete symbol. The symbol will substitute the variable animation, with the Branch Name + Variable Name. The declaration @ANA is for non-bool variables, @DIGI is for boolean variables.

The following programming code shows the header format which will be generated for every Monitor Pro mimic.

```
ASCII32,27,8,2004,13:51,46
W,BEGIN,"gerequip","Mimic1"
  TEMPLATE,"",1,1,1,0,1,0,1,1,1,0,0
  POSITION,0,0
  SIZE,1023,747
  BACKCOLOR,212,208,200,0,0,0
  TITLE,0,"","gerequip"
  STYLE,1,1,1,1,0,0,1,0,1,1,1,1,1,0,1,0,0
  GRID,1,1,8,8,0,0,0,0,0,0
  LAYERS,65535
  RIGHTS,0,1,0.000000,64.000000,1,65535,0,0
  INCLUDED,0,0,0,0,0
  LINK,1,"","","",""
  LINK,2,"","","",""
  LINK,3,"","","",""
  LINK,4,"","","",""
  LINK,5,"","","",""
  LINK,6,"","","",""
  LINK,7,"","","",""
  LINK,8,"","","",""
  LINK,9,"","","",""
  LINK,10,"","","",""
  BACKBMP,"",0,0
  BACKOBJECT,"",0,0
  BEFORE,"","","",""
  BINOBJ,"gerequip.binary"
W,END
```

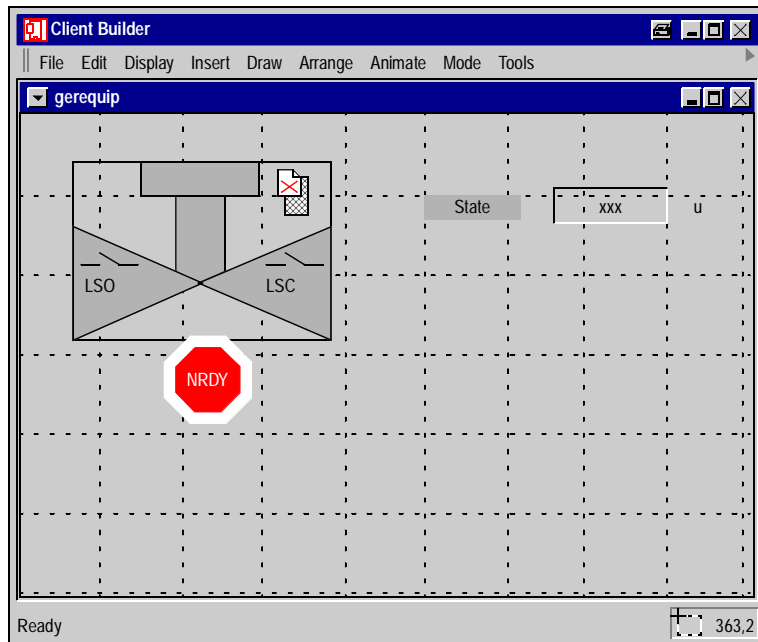
The following programming code shows the complete Monitor Pro mimic.

```

ASCII32,27,8,2004,13:51,46
W,BEGIN,"gerequip","Mimic1"
    TEMPLATE,"",1,1,1,0,1,0,1,1,1,0,0
    POSITION,0,0
    SIZE,1023,747
    BACKCOLOR,212,208,200,0,0,0
    TITLE,0,"","gerequip"
    STYLE,1,1,1,1,0,0,1,0,1,1,1,1,0,1,0,0
    GRID,1,1,8,8,0,0,0,0,0,0
    LAYERS,65535
    RIGHTS,0,1,0.000000,64.000000,1,65535,0,0
    INCLUDED,0,0,0,0,0
    LINK,1,"","","",""
    LINK,2,"","","",""
    LINK,3,"","","",""
    LINK,4,"","","",""
    LINK,5,"","","",""
    LINK,6,"","","",""
    LINK,7,"","","",""
    LINK,8,"","","",""
    LINK,9,"","","",""
    LINK,10,"","","",""
    BACKBMP,"",0,0
    BACKOBJECT,"",0,0
    BEFORE,"","","",""
    BINOBJ,"gerequip.binary"
W,END
FONTS,BEGIN
    FONT,1,-24,0,700,0,0,"Arial"
    FONT,2,-13,0,400,0,0,"System"
    FONT,3,-16,0,700,0,0,"Arial"
FONTS,END
COLORS,BEGIN
    COLOR,1,0,0,0,0,0,0
    COLOR,2,222,216,163,0,54,0
    COLOR,3,192,192,192,0,0,0
    COLOR,4,192,192,192,0,0,1
COLORS,END
O,BEGIN,S,""
    B,25,25,0,0,0,0,1,0,6400,0,1
    PP,"symb","monitor1:gerequip_con01",25,25,0,0,1,0
    SUB,"@DIGI","monitor1:gerequip_con01"
O,END
O,BEGIN,S,""
    B,25,25,0,0,0,0,1,0,6400,0,1
    PP,"valve_en","monitor1:gerequip_VAS01",25,25,0,0,1,0,,,,,,,,,
O,END

```


The following figure shows the symbol and free variable from the coding example above.



9.2 iFIX and Unity Application Generator

Overview

Introduction

Unity Application Generator supports Intellution's iFIX as HMI. For iFIX Unity Application Generator generates all databases and pictures necessary for the process visualization.

The iFIX application is generated on the engineering PC and then the generated objects must be transferred to the production PCs. This chapter describes in detail the procedures from the preparation of the iFIX generation to the running of an existing application.

What's in this Section?

This section contains the following topics:

Topic	Page
Introduction	227
How to Configure iFIX for the Use with Unity Application Generator	228
Manual Configurations Before Generation for iFIX	230
How to Generate a New iFIX Application	233
How to Generate an iFIX Application Incrementally	234
Unity Application Generator and iFIX Drivers	235
How to Deploy the Generated Application to the iFIX Nodes	236
How to Run an Existing Unity Application Generator Project	237
Configuring iFIX Redundancy	238

Introduction

Terminology

The following table explains the corresponding terms in Unity Application Generator and in iFIX:

Term in Unity Application Generator	Corresponds to the term in iFIX
Data Server	SCADA node
HMI	View node
Data Servers and HMIs (all nodes of HMI system)	iFIX nodes

Reference Information

A detailed list of the generated objects for iFIX you will find in *Generation for iFIX*, p. 344

How to Configure iFIX for the Use with Unity Application Generator

Introduction

Before you use iFIX with Unity Application Generator, you must configure iFIX in a certain way. The steps for this configuration are described in the following.

Note: Intellution recommends that the iFIX node name and the computer name are the same. The iFIX node name is specified as part of the installation of iFIX. After the installation the Modbus Plus/Ethernet Driver MBT V2.0 has to be installed.

How to get the MBT Driver?

The MBT driver is not a product of the Schneider Electric company. You have to contact Dimension Software Inc. to order the driver.

- Phone
+1 (828) 635-7189
- Fax
+1 (828) 625-5319
- Web
<http://www.caro.net/dsi>
- e-Mail
dsi@caro.net

The part number of the driver is MBT, the description is MODBUS TCP/Plus OPC Server.

Procedure for iFIX Configuration

For iFIX configuration for Unity Application Generator follow the steps:

Step	Action
1	Run the system configuration from Programs → iFIX in the task bar.
2	Select Configure → SCADA .
3	In the I/O driver name click the ? button.
4	From the list of available drivers choose Modbus Plus/Ethernet Driver MBT V2.0 .
5	Confirm with OK .
6	Click the Add button. Result: The driver appears in the configured I/O drivers list.
7	Confirm with OK . Result: You will be asked to answer the question: Database DATABASENAME does not currently exist, or is not in your database directory, use anyway?
8	Answer with Yes .
9	Save with File → Save .
10	End the configuration with Exit

Manual Configurations Before Generation for iFIX

Introduction

Unity Application Generator is not able to generate all information needed in the system configuration (SCU) of iFIX. Therefore, some configurations have to be done manually before generation. These are described in the following.

Alarm Areas Configuration

For the Alarm Area configuration the following rules apply:

- You must define an Alarm Area for each Control Domain of the HMI.
 - You must name the Alarm Area the same as the Control Domain name.
 - The maximum length of an Alarm Area name is 30 characters.
 - The number of Alarm Areas in iFIX is unlimited.
-

Result

Each generated tag belongs to an Alarm Area (if the Control Domain is defined).

Security Area Configuration

For the Security Area configuration the following rules apply:

- You must define a Security Area for each combination of Control Domain and access level.
 - Each Security Area name must be the concatenation of the Control Domain name and of the ID of the different access levels (see example below). Access levels are defined in the customization.
 - The maximum length of a Security Area name is 20 characters.
 - The maximum number of Security Areas is 254.
 - The access rights of the operators to the different Security Areas must be defined in iFIX Security.
 - An operator can have access rights for multiple Security Areas. Groups can be defined with access rights on multiple Security Areas.
 - An ActiveX variable belongs to a Security Area because the Control Module, which is represented by the ActiveX, belongs to an Equipment Module. If an operator wants to modify a variable value, the ActiveX first queries the iFIX security system if the operator currently logged in has rights on the Security Area of that variable.
-

Result

Each generated tag belongs to a Security Area (if the Control Domain and access level are defined).

Default Values for Access Levels

The default values for access levels in the default customization file IATBASIC10.OSC are:

Access Level	Default Value in IATBASIC.OSC
1	Operator
2	Production
3	Technical
4	Maintenance
5	Factory

Example

For example, for a Control Domain named CDWashing, the following Security Areas have to be defined in the iFIX system configuration:

Access Level	Example Value	Example Security Area Name
1	Operator	CDWashing_1
2	Production	CDWashing_2
3	Technical	CDWashing_3
4	Maintenance	CDWashing_4
5	Factory	CDWashing_5

Working with Remote Nodes

For working with remote nodes the following rules apply:

- The iFIX node name and the computer's name must be exactly the same.
Note: Special characters like "-" are not allowed in the iFIX node names.
- In the iFIX system configuration you have to declare the remote nodes (in the network configuration section).

Modbus TCP/IP I/O Driver

Most of the configuration for the I/O driver will be carried out by the generation in Unity Application Generator: There will be a configuration file for each Data Server and each communication driver. This file will be named DATASERVERNAME.MBT. Other modifications have to be made to complete driver configuration (see driver manual). Fields set by generation are defined in *Generated iFIX Driver Configuration from Unity Application Generator Point of View*, p. 359 and *Generated iFIX Driver Configuration from the Driver Point of View*, p. 361. If these fields are modified they will be overwritten by Unity Application Generator generation.

Duplicates of SCoD ActiveX

It is possible to create multiple copies of SCoD ActiveX in different pictures of iFIX. To do so, you can copy and paste the ActiveX, but the properties of the copies must not be changed. There is an option in Unity Application Generator to manage correctly the manually copied ActiveX.

If this option is selected in Unity Application Generator and Control Modules are modified, the generator seeks if the same object has been copied to other pictures of the ... \DYNAMICS\PIC directory and will update all copies (SAME ActiveX type and same object name).

Delete a Control Module

If a Control Module is deleted in Unity Application Generator all copies of the ActiveX will be deleted.

How to Generate a New iFIX Application

Procedure for New Generation

Steps how to generate a new iFIX application:

Step	Action
1	Generate the Concept/Unity Pro project(s).
2	Download Concept/Unity Pro project(s) into the PLC(s) or Concept/Unity Pro simulator.
3	Start iFIX.
4	Login in as a user who has all access rights.
5	Start iFIX workspace.
6	Run iFIX generation.
7	Check warnings and errors in generation.
8	Check PowerTool for driver.
9	Reload the generated configuration file of the driver (the name of the configuration file is the name of the Data Server defined in Unity Application Generator).
10	Complete the driver configuration.
11	Check if Channels, Control Modules and Data Blocks are enabled. If not: Enable them and start the driver in run mode (start button)
12	Reload the generated database in the Database Manager (the name of the database is the name of data server defined in Unity Application Generator).
13	Switch to run mode.
14	Open the pictures.

How to Generate an iFIX Application Incrementally

Procedure for Incremental Generation

Steps how to incrementally generate an iFIX application:

Step	Action
1	Generate the Concept/Unity Pro project(s).
2	Download the changes of the Concept/Unity Pro project(s) into the PLC(s) or Concept/Unity Pro simulator.
3	In iFIX switch to configure mode.
4	Generate the iFIX application incrementally.
5	Check PowerTool for the driver.
6	Reload the generated configuration file of the driver (the name of the configuration file is the name of Data Server defined in Unity Application Generator).
7	Check warnings and errors in generation.
8	Complete the driver configuration.
9	Check if Channels, Control Modules and Data Blocks are enabled. If not: Enable them and start the driver in run mode (start button)
10	Reload the database after incremental generation.
11	Switch to run mode.
12	Open the pictures.

Unity Application Generator and iFIX Drivers

Overview

At the moment, the only driver that can be used is the Schneider Automation Modbus Plus / Ethernet driver MBT V2.0.

The driver configuration file is generated before the SCADA generation and is loaded automatically into the driver, otherwise the I/O addresses of the tags would not be accepted as a valid address by the iFIX database.

The driver's Data Blocks are defined using different address ranges defined by the Unity Application Generator Memory Mapper according to data types and priority (event, parameter, synoptic).

Data Types

The data types managed by the driver must match the Unity Application Generator data types.

The following table defines the correspondences between the Unity Application Generator types, how they are grouped by the Memory Mapper, which Data Block types are used, and which auxiliary conversion has to be made at the tag level.

UAG / Concept / Unity Pro HMI, PLC_HMI Type	Memory Mapper Type	Driver's Data Block Type	Tag special conversionType
BOOL	BOOL	Digital	None
WORD	WORD	Unsigned Integer	None
UINT	WORD	Unsigned Integer	None
INT	WORD	Unsigned Integer	Signed Integer
UDINT	DINT	Unsigned Long	None
DINT	DINT	Unsigned Long	Signed Long
REAL	REAL	Float	None
BYTE	WORD	Unsigned Integer	None

Redundancy

The MBT driver supports redundancy for Quantum and Momentum. Two IP addresses can be configured (primary and backup) for two NOE modules. This protects against NOE module failure but **not** against network failure or from PC Ethernet card failure.

How to Deploy the Generated Application to the iFIX Nodes

Overview

When the generation for HMI is completed on the engineering workstation, the generated objects have to be copied manually to the corresponding PCs in the production plant. The generated objects are contained in the ... \DYNAMICS directories of the engineering workstation.

For an overview of the control topology see *The Topology of a Control System*, p. 62.

Terminology

The following table explains the corresponding terms in Unity Application Generator and in iFIX:

Term in Unity Application Generator	Corresponds to the term in iFIX
Data Server	SCADA node
HMI	View node
Data Servers and HMIs (all nodes of HMI system)	iFIX nodes

Procedure

For deploying the generated HMI application to the iFIX nodes follow the steps:

Step	Action
1	Configure each iFIX node manually: <ul style="list-style-type: none"> Each SCADA node must have the same name as the corresponding Data Server in Unity Application Generator. Note: To ensure the functioning of the distributed system it is recommended that the iFIX nodes have the same names as the PCs.
2	On each View node PC install the SCoD Library and ActiveX support DLLs needed for ActiveX access to the iFIX database.
3	Copy the database files <SCADA NODE NAME>.PDB from the engineering workstation to the corresponding SCADA node PC in the plant.
4	Copy the driver configurations <SCADA NODE NAME>.MBT from the engineering workstation to the corresponding SCADA node PC in the plant.
5	Copy the *.GRF pictures from the ... \DYNAMICS\PIC directory of the engineering workstation to the corresponding iFIX View nodes PCs.

How to Run an Existing Unity Application Generator Project

Functioning of Communication

At run time, the ActiveX get the real-time information from iFIX runtime databases. This is achieved by using DLLs developed by Unity Application Generator, which use the Intellution EDA toolkit functions. Thus, communication exists only between SCADAs and PLCs.

Two DLLs have been implemented to manage exchanges between the iFIX databases and the ActiveX: one for read/write values, another for security, alarming and alarm acknowledgment. Using the iFIX network capabilities, an ActiveX can get its information from a SCADA on a different iFIX node.

Procedure for Running an Existing Project

Steps how to run an existing Unity Application Generator project:

Step	Action
1	Run Concept/Unity Pro.
2	Open corresponding Concept/Unity Pro project(s).
3	Connect to the PLC(s), make sure that you are EQUAL.
4	Start iFIX.
5	Login in as any user with the appropriate access rights.
6	Start iFIX workspace.
7	Check warnings and errors in generation.
8	Check PowerTool for the driver.
9	Reload the generated configuration file of driver (the name of the configuration file is the name of Data Server defined in Unity Application Generator).
10	Complete the driver configuration.
11	Check if Channels, Control Modules and Data Blocks are enabled. If not: Enable them and start the driver in run mode (start button)
12	Reload the generated database into the Database Manager (the name of the database is the name of data server defined in Unity Application Generator).
13	Switch to run mode.
14	Open the pictures.

Configuring iFIX Redundancy

iFIX Redundancy	<p>iFIX provides network capabilities for SCADA node redundancy. Two different PCs can be defined as one SCADA node with different physical node name but same logical node name. This is considered as a unique SCADA node, made of a Primary node plus a Backup node.</p> <p>A View node gets the data from the logical node, and the configuration of the logical node defines which physical node is the Primary node and which one is the Backup node. During normal operation the data come from the Primary node, but if it fails, the iFIX system switches to the Backup node.</p>
Unity Application Generator and iFIX Redundancy	<p>From Unity Application Generator point of view, managing SCADA redundancy is transparent and is not configured in Unity Application Generator. The user has to define the data server as the logical SCADA node name. The logical node will be used for iFIX pictures generation.</p>
LAN Redundancy	<p>LAN redundancy between different iFIX nodes (View and SCADA nodes, but not PLCs) can be also implemented, but this is outside of the scope of Unity Application Generator and belongs to the supervisory system redundancy capabilities.</p>
Reference	<p>Detailed information about iFIX Redundancy can be found in : iFIX Electronic Books - Mastering iFIX - 2. Redundancy.</p>

9.3 Generic HMI and Unity Application Generator

Using Unity Application Generator with a Generic HMI

Working Principle

When you start a new customization with the customization editor you select the option for using a generic HMI.

If you have selected this option Unity Application Generator creates a XML file for each Data Server for import containing all tags necessary for your HMI application, see *Generation for a Generic HMI*, p. 363.

Additionally, Unity Application Generator creates a second file for each Data Server that contains data from the XML file reformatted in a way the user's HMI is able to import. How the data is reformatted is defined in a stylesheet file (XSL file).

With the customization editor the user specifies which export formats (XSL files) are available in the Unity Application Generator project.

You can create your own XSL files to adjust the format of the output file to the format of your HMI. The XSL files have to be copied to the DB directory of the Unity Application Generator directory.

Each Data Server in the Unity Application Generator project has a property to specify which export format to use for generation.

Note: The XML file for generic HMIs will always be generated completely new, even for incremental changes in Unity Application Generator.

Specifying Export Formats for Generic HMI Data

UAG provides the possibility to specify more than one transformation for the XML file which will be generated during the generation of a Generic HMI. This allows extracting different parts of the data in the XML file to different resulting files.

For each transformation the user has to specify the style sheet file and the name of the file to be created as the result of the transformation. All transformations use the same generated XML file as input, this means it is not possible to define a chain of transformations.

Customization and Project Maintenance

10

Overview

Characterization The project initialization, customization and maintenance is an important part in an application development done with Unity Application Generator. In fact it is the first step before starting a new project.

The tasks are divided into different parts:

- Adapting Unity Application Generator with the help of the Unity Application customization editor
- Setting the Analyzer options and the code generation options inside Unity Application Generator
- Version management and project documentation inside Unity Application Generator.

This chapter will explain these tasks.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
10.1	Customizing Unity Application Generator	243
10.2	Project Maintenance	258

10.1 Customizing Unity Application Generator

Overview

Introduction

This section describes how to create and use customization files.

What's in this Section?

This section contains the following topics:

Topic	Page
Introduction	244
Working with the Customization Editor	245
The Customization Options	247
Defining Naming Conventions	249
User Defined Modules - Overview	251
User Defined Modules - Properties	253
How to Define a Generic Module	255
How to Define a ModConnect Partner Module	256
How to Change the Customization	257

Introduction

Introduction

Unity Application Generator can be adapted to the customer needs with respect to libraries used, tools used, naming conventions, data format and so on. This adaptation is done with the help of the Unity Application customization editor.

Customization information is stored in files named `PROJECTNAME.OSC`. Each Unity Application Generator project requires its own customization file.

When a new project is created with Unity Application Generator, a customization file has to be selected. The selected file is then used for the customization of the project and a copy is created of the selected file with the name of the project.

In this way all new projects can be created with a standard customization, but it is also possible to change the customization of an existing project by editing the file `PROJECTNAME.OSC`.

If the customization file for an existing project is changed Unity Application Generator will detect this and offers to analyse the customization. In case of an error, e.g. an existing name does not fit to the changed naming convention, Unity Application Generator will report an error which has to be corrected manually. Unity Application Generator will give the option to analyse the customization every time the project is opened until the project conforms to the customization.

Working with the Customization Editor

What is the Customization Editor?

The customization editor of Unity Application Generator is a separate program used for the creation and modification of customization files for Unity Application Generator projects. A customization file named PROJECTNAME.OSC is necessary for each project.

The customization file contains user defined information for the configuration of the Unity Application Generator environment, like libraries and tools used, naming conventions, data formats and so on.

File Administration

For the administration of customization files you use the functions New, Open, Close, Save, Save as..., Exit in the **File**-Menu.

Display File Information

The HMI supported by the currently open customization file can be displayed with **Customization** → **Supported HMI**.

Note: The HMI used in the project is selected when a new customization is created. It cannot be changed afterwards.

The version of the currently open customization file can be displayed with **Customization** → **Version**.

Enter Customization Information

In the left part of the customization editor's starting screen you see the structure of your customization file in form of a tree.

The tree contains four principal groups of subjects that you can customize:

- General
- Naming convention
- Data
- PLC

When clicking on the + in front of these groups the sub items are displayed. For entering all customization information you go through all items given in the tree. Each time when you select another item a dialog is opened leading your input. For more detailed information on the options see *The Customization Options, p. 247* and *Defining Naming Conventions, p. 249*.

Check Customization

The entered customization can be checked with **Customization** → **Check**. The result is a report containing a validation for all customized items. Erroneous items are displayed in red. An error in customization can for example be a name definition that exceeds the maximum length of the object name. All errors have to be corrected before the customization file can be used in the project.

Handling

The handling of the Unity Application Generator customization editor is similar to most menu-driven modern software programs.

For most actions, you have different possibilities for getting the task done:

- Use the buttons of the dialogs or the menu items
 - Double-left-click on the element or in the field which is the target of your action
 - Right-click on the element or in the field which is target of your action to display a context menu
-

The Customization Options

General Customization Options

Under the general customization options the available Control Module Type libraries, the supported tools and the generic export formats are defined.

Unity Application Generator allows the user to attach documents for the process design to the different objects. Any number of Windows applications (or tools) for the different document types (for example Microsoft Word, Microsoft Excel, Autocad) can be defined in the customization.

The user defines which export formats can be used in the Unity Application Generator project by assigning names to the stylesheet files (XSL files). Only stylesheet files located in Unity Application Generator's "DB" directory can be selected.

Naming Conventions

Every industry has its own standards (e.g. S88 for batch control) and methods (e.g. GAMP in pharmaceutical). In addition, each company has its own standard for naming objects. Unity Application Generator allows the user to define rules for the naming of the objects according to the customer standards.

Separate naming rules can be defined for each object shown in this table:

Physical Model	Topological Model
Area	PLC
Process Cell	Data Server
Unit	Control Domain
Equipment Module	Channel
Control Module	HMI
Variable	Network Segment
	Network Node

You will find more detailed information about the definitions of naming conventions in *Defining Naming Conventions*, p. 249.

Data

This part of the customization defines:

- Access levels
- Alarm groups
- Alarm priority

Note: The customized options for alarm values have to fit the available options in the HMI (e.g. low, medium, and high for iFIX).

- Archive names
- Communication frames
- Display formats for numbers
- Measurement units and measurement groups
- Valid data types

Note: Valid data types are the elementary data types of Concept/Unity Pro. Structured data types which are defined and used in Concept/Unity Pro must be added by name.

PLC

This part of the customization defines which PLC configurations are allowed in the projects:

- PLC families
- Racks
- HW modules

The standard customization file IATBASIC10.OSC contains complete lists of rack types and HW modules from which the project manager can select the subset of racks and modules which are valid for his projects. Only the selected racks and modules will be offered to the control engineer when he is configuring a project. Unity Application Generator allows furthermore to integrate user defined HW modules: generic modules (with Concept and Unity Pro) or ModConnect partner modules (only with Concept).

For more information on user defined modules, see: *User Defined Modules - Overview, p. 251*

Defining Naming Conventions

Description Length

The first sub item in the naming conventions tree is called *description length*. Here, you define the maximum length of the description of each object, that can be introduced when such an object is created in Unity Application Generator.

Object Name Composition

The next sub items in the naming conventions tree are the objects, for which a naming convention can be defined: Area, Process Cell, Unit, Equipment Module, Control Module, variable, PLC, Channel, HMI, Control Domain, alarm overview, data server, network node, network segment. For each object you get a dialog for the definition of the name composition.

The naming convention for each object can be built of one to six different fields. Each field can be of the following types:

- Letter
 - Number
 - List
-

Letter Field

The following options are available for letter fields:

- Fixed size
- Variable size
- Define a fixed string
- Capital and/or small letters, with or without digits

Note: The text of a letter field with a fixed string cannot be changed later on in Unity Application Generator.

Note: The variable size string may start with a digit (but the complete field must not be numeric).

Number Field

For a number field you define the maximum number of digits. Furthermore, you choose, if leading zeros shall be added or not.

Field Defined by User's List

The third type of field allows the user of Unity Application Generator to select a part of the object name from a list. The elements of this list are defined with the customization editor using the list designer. Lists can be newly created, deleted and modified using the list designer. All lists defined during customization get a unique name and can be used for fields of different objects.

Such a list contains two columns

- **Short:** The contents of the short column is the text used for the object name.
- **Description:** The contents of the Description column serves as description for the user of Unity Application Generator when selecting list element for the name to be defined.

Note:

- An empty list cannot be used in the name convention.
- It is not possible to delete a list used by an object of the naming conventions.
- The elements of a list may have a different number of characters for the short column, e.g.
 - a, b, c, d, e, ...
 - 11, 12, 13, 14, ...
 - A, B, AB, ABC, ABCD, ...
- "_" is allowed in the short column. (Lists containing "_" are not allowed in naming convention for Equipment Modules, Control Modules and Variables.)

User Defined Modules - Overview

What are User Defined Modules?

User defined modules are hardware modules that shall be used in the automation project configured with Unity Application Generator but do not belong to the Schneider Automation product catalog.

User defined modules can be either

- ModConnect partner modules (only with Concept) or
 - Generic modules (with Concept and Unity Pro).
-

ModConnect Partner Modules

ModConnect is an initiative to integrate third party modules into Concept.

ModConnect partner modules are modules that are known by name in Concept, specified in the Concept ModConnect Tool.

The administrator adds them to the list of modules in the customization.

He must specify the following for each ModConnect partner module:

- Exact name as in the Concept ModConnect Tool
- Number of inputs and outputs
- Number of 0x,1x,3x and 4x registers consumed
- Number of status registers
- Possible rack type(s)

ModConnect partner modules can be configured in Quantum racks and are generated in the Concept I/O map.

The ModConnect partner modules have to be imported in Concept with ModConnect Tool.

Generic Modules Generic modules are user defined hardware modules which are not known by name in Concept/Unity Pro.

The administrator adds them to the list of modules in the customization.

He must specify the following for each generic module:

- Unique name within the customization file
- Number of inputs and outputs
- Number of 0x (%M), 1x (%I), 3x (%IW) and 4x (%MW) registers consumed
- Number of status registers
- Possible rack type (Modbus Plus, Ethernet, Generic)

With the restriction of the possible rack types to `Generic` the user defined module is defined as a generic module. In the project, such modules can only be configured in generic racks and they are **not** generated in Concept/Unity Pro. They **can not** be configured in Quantum/Momentum/Premium racks and are **not** entered in the Concept/Unity Pro I/O mapping.

The specifications made are only used for the reservation of state RAM addresses and for the addresses of the I/O variables.

After generation, the integration of generic modules have to be completed manually. What the tasks are depends on the type of module.

For example, if you configure Momentum I/O, the data must be communicated with Peer Cop or I/O Scanner.

Or, if you configure PROFIBUS DP, you must additionally import the configuration into Concept with the Sycon tool.

User Defined Modules - Properties

Description of Properties

The properties for user defined modules are the following:

Property	Description	Options	Comment
Module Name	Name of the user defined module	Free text	Name must be unique within the customization file! <ul style="list-style-type: none"> ● For a generic module: choose a free name. ● For a ModConnect partner module: Name must be exactly the same as in ModConnect Tool.
Module Category	Type of module	<ul style="list-style-type: none"> ● Digital I/O ● Analog I/O ● Experts 	-
Description	User defined description of the user defined module	Free text	-
Possible racks	Rack type in which the user defined module can be configured; corresponds to property <code>Link Type</code> .	<ul style="list-style-type: none"> ● Quantum ● Premium ● SY/MAX ● 800 I/O ● Momentum ● I/O Bus ● Ethernet I/O ● MBP I/O ● Generic 	Generic rack: With the assignment of the user defined module to a generic rack, it becomes a generic module. For more information on link types see <i>Racks and Modules, p. 98</i>
Number Inputs	Number of input connections on user defined module	Integer	-
Number Outputs	Number of output connections on user defined module	Integer	-

Property	Description	Options	Comment
Register 0x, 1x, 3x, 4x resp. %W, %I, %IW, %MW	Number of 0x, 1x, 3x, and 4x resp. %W, %I, %IW, %MW registers needed for user defined module	Integer	For the number of inputs, outputs and registers, the following is valid: <ul style="list-style-type: none"> • Number of outputs must be the same as number of registers 0x/%M or 4x/%MW. • Number of inputs must be the same as number of registers 1x/%I or 3x/%IW.
Status Register 3x/%IW, 4x/%MW	Number of status registers 3x/%IW and 4x/%MW needed for user defined module	Integer	-

How to Define a Generic Module

Overview

If generic modules shall be configured in the Unity Application Generator project(s), the administrator has to define them in the customization file with the Unity Application customization editor.

Procedure for Defining a Generic Module

For defining generic modules with the Unity Application customization editor follow the steps:

Step	Action
1	Open the subgroup <code>User Defined Module</code> . Result: An empty list with the columns <code>Module Category</code> , <code>Name</code> , <code>Description</code> , ... appears.
2	Right-click anywhere in the list and select Add from the context menu. Result: The dialog window <code>Add User Defined Module</code> appears.
3	Enter the information on your generic module in the fields. Please note: <ul style="list-style-type: none"> ● By restricting the possible rack type to only <code>Generic</code>, the user defined module is defined as a generic module. ● Be sure to very precisely fill out the fields, because the data cannot be checked by Unity Application Generator! For field descriptions see <i>User Defined Modules - Properties</i> , p. 253.
4	When you have filled out all the fields, confirm with OK . Result: The new generic module is entered in a line of the list.
5	If necessary, revise data with the context menu items Remove and Modify .
6	Save your customization file.

Result

When using this customization file for projects, your user defined generic modules are available in the drop down lists for entering modules. They can be configured in a project as other hardware modules. They can only be configured in a generic rack. For such generic modules **no** entries are generated in the Concept/Unity Pro I/O map. The necessary addresses will be defined in the corresponding PLC as defined for the generic module.

Enhanced Ethernet Rack

It is possible to create user-defined Ethernet racks which are not supported by default within UAG. An Ethernet rack belongs to an Ethernet communication module (like ETY 4103 or NOE 771-xx). After the creation of an Ethernet rack, the user can assign these Ethernet communication modules. See also *Enhanced Ethernet Module*, p. 107

How to Define a ModConnect Partner Module

Overview

If ModConnect partner modules shall be configured in the Unity Application Generator project(s), the administrator has to define them in the customization file with the Unity Application customization editor.

When using this customization file for projects, these ModConnect partner modules are available in the drop down lists for entering modules and can be used as any other hardware module.

Procedure for Defining a ModConnect Partner Module

For defining ModConnect partner modules with the Unity Application customization editor follow the steps:

Step	Action
1	Open the subgroup <i>User Defined Module</i> . Result: An empty list with the columns <i>Module Category</i> , <i>Name</i> , <i>Description</i> , ... appears.
2	Right-click anywhere in the list and select Add from the context menu. Result: The dialog window <i>Add User Defined Module</i> appears.
3	Enter the information on your ModConnect partner module in the fields. Please note: <ul style="list-style-type: none"> • For using a ModConnect partner module fill out the exact name, as it is defined in ModConnect Tool. • Be sure to very precisely fill out the fields, because the data cannot be checked by Unity Application Generator! For field descriptions see <i>User Defined Modules - Properties</i> , p. 253.
4	When you have filled out all the fields, confirm with OK . Result: The new ModConnect partner module is entered in a line of the list.
5	If necessary, revise data with the context menu items Remove and Modify .
6	Save your customization file.

Result

When using this customization file for projects, your user defined ModConnect partner modules are available in the drop down lists for entering modules. They can be configured in a project as other hardware modules. They can be configured in a Quantum rack.

Note: ModConnect partner modules must be imported into Concept with ModConnect Tool.

How to Change the Customization

General

The customization file is the basic information which is required by Unity Application Generator. Each time a new project is created, a customization file has to be selected. The selected customization file is copied into the project directory with the same name as the project, with the file extension .OSC.

How to Change the Customization of an Existing Project

Changes to the original customization file will not influence a project. The changes must always be made to the file PROJECTNAME.OSC or the changed customization file has to be copied as PROJECTNAME.OSC. If the customization file for an existing project is changed, Unity Application Generator will detect this and will offer to analyse the customization. In case of an error, e.g. an existing name does not fit the changed naming convention, Unity Application Generator will show an error which has to be corrected manually. Unity Application Generator will show the option to analyse the customization every time the project is opened until every error of the customization has been corrected.

10.2 Project Maintenance

Overview

- Characterization** Project management inside Unity Application Generator is divided in two major tasks:
- Setting the Analyzer and generation options Unity Application Generator
 - Version management and project documentation
-

What's in this Section?

This section contains the following topics:

Topic	Page
Setting Options for Analysis and Generation	259
Version Management and Change Tracking	260
Project Documentation (Report Generator)	262
Trouble Shooting	262

Setting Options for Analysis and Generation

Introduction

You have options for the way the project is analyzed and if a completely new generation should be carried out instead of an incremental generation (see *Overview of Generated Code and Generation Principles*, p. 275).

Under **View** → **Options** you find the tabs **General**, **PLC**, **Monitor Pro**, **Analyze Project**, **Memory Mapper**.

Options for Project Analysis


Under the tab **Analyze Project** you have the possibility to select which parts of the project shall be analyzed and how many details the report should contain.

Note: Here, you may enter, after how many warnings the Analyzer should stop. If you desire an unlimited number of warnings, enter 0.

Option for New Generation

Under the tabs **PLC** and **Monitor Pro / iFIX** you have for each the option `Generate completely new (only next generation)`. If you select this option for PLC or Monitor Pro / iFix, completely new PLC project(s) or Monitor Pro / iFIX applications(s) will be generated.

The same option exists for the Memory Mapper to optimize the PLC memory usage.

	<p>CAUTION</p>
	<p>Risk of loosing code</p> <p>If you use the option for new generation all manual changes in PLC or Monitor Pro / iFIX will be lost.</p> <p>Failure to follow this precaution can result in injury or equipment damage.</p>

Note: When you have carried out a new generation, your old code is saved in a backup, see *Version Management and Change Tracking*, p. 260.

Version Management and Change Tracking

Overview

Unity Application Generator allows the user to save different versions of a project. If necessary, it is possible to go back to an older version of the project. In addition, Unity Application Generator keeps track of all changes which have been made in a change history.

Version Management

Besides the possibility to save a project with the `Save as...` command, the user can create snapshots of the actual status in the `Versions` dialog, invoked by the menu item `Files / Versions`.

In opposite to the `Save as...` command, the version management has two significant advantages:

- Multiple development steps can be frozen in the same project.
- Not only the UAG project will be saved, but the according PLC and HMI projects.

In the `Versions` dialog the actual version can be saved, and stored versions can be restored or deleted. A comment will be attached to each version. The user can enter additional information. The `Change History` since the last save or the beginning of the project can be displayed. So it is very easy to go back to another version, if the last user modifications did not fulfill her expectations. While an old version is restored, already existing folders are copied in folders with the extension `.BAK`.

The following informations will be saved:

- Unity Application Generator project and customization file
 - Generated Concept / Unity Pro projects
 - MonitorPro projects:
 - Generated HMIs in HMI application path
 - Generated Data Servers in Data Server application path
 - iFIX projects:
 - Pictures (*.grf in dynamics\pic)
 - HMIs (*.gil in dynamics\pic)
 - DataServers ([ObjectName].* in dynamics and dynamics\pdb)
 - AlarmAreas (AlarmAreas.aad in dynamics\pdb)
-

Change Tracking Any user action which changes an object in Unity Application Generator will be recorded in the change history. The change history shows the information who has done which changes at what time. The change history can be exported as a comma separated value file (CSV) for further processing e. g. with Microsoft Excel. If a project is saved as a particular version by Unity Application Generator version management, the change history file will be stored in the saved version but cleared in the current version. With this approach the change history file always contains the changes done since the last version freeze. This is especially useful if a version had been validated and one wants to check what has been changed since the last validation. I.e changes made since validation are always recorded in the change history.

Project Documentation (Report Generator)

Overview

Unity Application Generator allows the user to document the project automatically. A report generator is built into Unity Application Generator which creates a project report as Microsoft Word file.

What Will be Documented?

The report generator offers multiple options for the generated report, for example:

- Table of contents
- Physical Model hierarchy
- Equipment Modules
- Alarm variables
- Command variables
- IO variables
- PLCs
- HMIs
- Network configuration
- Interlocks
- Instruments

The generated report can either be a new report or it can be appended to an existing report.

How to Define the Document Layout

When the report is generated, a template file (DOT) has to be selected which will be used by the report generator. Adopting this template file allows the user to design the report according to the user company requirements.

Additional Documentation Options

Other functions in Unity Application Generator can also be used to document additional information:

- The change history file can be exported as a comma separated value (CSV) file.
 - The message window content (Analyzer, code generator output) can be exported as a comma separated value (CSV) file.
-

Trouble Shooting

Repair Project Database

If your project database has been damaged, for example due to power failure or a PC crash, use the item **File** → **Repair Project Database** for repair.

Appendices



Overview

What Kind of Information is in the Appendices

The appendices contains release notes and reference information, e. g. information related to the generated output of Unity Application Generator.

What's in this Appendix?

The appendix contains the following chapters:

Chapter	Chapter Name	Page
A	Release Notes Version 2.1	265
B	Generated Code	273
C	Format of the CSV Files for Import and Export	367
D	Format of the XML File for Generic HMI	389

Release Notes Version 2.1



Overview

Introduction

Unity Application Generator Version 2.1 replaces Unity Application Generator V2.0, V1.5, V1.4 and V1.3.

Some new features and functionalities have been implemented. This chapter describes:

- The changes compared to Version 2.0,
- How you migrate your existing projects to the new versions of Unity Application Generator
- How you migrate your existing projects to the new versions of Concept.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
New Features in Unity Application Generator Version 2.1	266
Hardware Requirements	266
Software Requirements	267
Installation information for new Users	268
Upgrade of Existing Projects to UAG 2.1 and Concept V2.6	271

New Features in Unity Application Generator Version 2.1

New Features for Unity Pro

The following (new) features for Unity Pro have been implemented:

Feature	Description	See..
Support of Unity Pro version 2.0 XL	-	-
Complete integration of Unity Pro	Complete customization of the Unity Pro PLCs Quantum and Premium within UAG. Included the selection of racks and modules.	-

Note: The following hardware modules are not supported.

- TSX P57 0244M
 - TSX P57CA 0244M
 - TSX P57CD 0244M
-

UAG and Monitor Pro

Integration of Monitor Pro Version 7.2.

Hardware Requirements

Hardware Requirements

The minimum configuration of the PC to run Unity Application Generator is:

- PC Pentium 1000 MHz or higher
 - Minimum 512 MB RAM
 - 50 MB hard disk space for the installation of Unity Application Generator
 - CDROM drive
 - VGA graphics adapter and screen (minimum resolution: 1024x768)
 - Microsoft compatible mouse
-

Software Requirements

Operating System Unity Application Generator runs under Microsoft Windows 2000 and Windows XP.

Programming Software UAG is generating PLC logic and configuration for Quantum/Momentum type PLCs in Concept and Quantum/Premium type PLCs in Unity Pro. Therefore the following programming software has to be installed on your PC:

- Concept V2.5 SR2 or Concept V2.6 SR1 and/or
- Unity Pro XL V2.0

Note: During installation Unity Application Generator detects the existing Concept/Unity Pro versions on your PC and will install the compliant DLLs and Function Blocks. Setting a PC with different Concept versions or different Unity versions in parallel is **not** supported.

Monitor Pro Unity Application Generator generates tags and graphics for the Monitor Pro SCADA system. Therefore the following software has to be installed on your PC:

- Monitor Pro Version 7.2

iFIX Unity Application Generator generates tags and graphics for the Intellution iFIX SCADA system. Therefore the following software has to be installed on your PC:

- Intellution iFIX V2.6 or iFIX V3.0
- Modbus Plus / Ethernet MBT V2.0 Driver

Acrobat Reader With Unity Application Generator you have received the user manual and the documentation for Smart Control Devices. Therefore the following software needs to be installed on your PC:

- Adobe Acrobat Reader

The setup for Acrobat Reader is available on the Unity Application Generator product CD.

MS Word Unity Application Generator allows to generate reports for project documentation. This is done with Microsoft Word. Therefore the following software has to be installed on your PC:

- Microsoft Word 97 or Word 2000/2003

Installation information for new Users

Installation Overview

As a new user of Unity Application Generator you need to install the following software:

1. PLC programming software: Schneider Electric Concept V2.5 SR2 **or** V2.6 SR1 **and/or** Schneider Electric Unity Pro XL V2.0
2. HMI programming software: Schneider Electric Monitor Pro V7.2 or Intellution iFIX V2.6 **or** V3.0
3. Driver for iFIX: Schneider Electric Modbus Plus / Ethernet MBT V2.0
4. Schneider Electric Unity Application Generator V2.1

Note: If you want to generate documentation of your Unity Application Generator projects, Microsoft Word 97 or Word 2000/2003 has to be installed on your PC.

Unity Pro Installation

For the installation of Unity Pro follow the steps:

Step	Action
1	From Unity-CD run SETUP.EXE
2	Follow the instructions of the installation routine.

Concept Installation

For the installation of Concept follow the steps:

Step	Action
1	From Concept-CD run SETUP.EXE and follow the instructions.
2	From Concept Service Release CD run SETUP.EXE and follow the instructions.

Monitor Pro Installation

For the installation of Monitor Pro follow the steps:

Step	Action
1	From CD Monitor Pro V7.2 run SETUP.EXE
2	Follow the instructions of the installation routine. Note: Do not install any driver!

iFIX Installation

For the installation of iFIX follow the steps:

Step	Action
1	From CD iFIX V2.6 or V3.0 run SETUP.EXE
2	Follow the instructions of the installation routine. Note: Do not install any driver!

MBT Driver Installation

For the installation of the Modbus Plus / Ethernet Driver MBT V2.0 follow the steps:

Step	Action
1	From CD MBT V2.0 run SETUP.EXE.
2	Follow the instructions to install MBT, use default values offered by Setup.
3	Configure the MBT driver in iFIX's System Configuration. Note: Do not change the Auto Create option in the Power Tool to Off (Options → Setup → Advanced).

Unity Application Generator Installation

For the installation of Unity Application Generator follow the steps:

Step	Action
1	Introduce the CD Unity Application Generator in the CD-ROM drive of your system. Result: A startup screen for the installation process appears. Note: If the startup screen does not appear automatically do the following: From Unity Application Generator CD run SETUP.EXE.
2	Follow the instructions of the installation routine. Note: Specify the same directory for Concept as you have chosen in the Concept installation.

Specifying Directories for Application Data Files

UAG uses several files to read or write different kinds of application data. In older versions of UAG the user has to specify the directories for some of the files, but not for all. Some files types are expected to be located in subdirectories with fixed names of the UAG installation directory. These subdirectories are created during installation.

The following list specifies the predefined directories for the application data files in older versions of UAG.

- SCoD libraries (*.osl) in subdirectory \DB
- XML style sheets (*.xsl) for Generic HMI generation in subdirectory \DB
- Help files for SCoDs (user defined, e.g. *.pdf) in subdirectory \Doc

Today UAG will provide the possibility to specify the directories for all application data files, to allow archiving all application-related data files easily.

All directories for application-related data files should be specified on a single page of the options dialogue of UAG. It will be possible to specify a base directory for all application files, whereby subdirectories with appropriate default names are created. But the user is able to override those default names. It is also possible to specify those directories either with absolute pathnames or relative to the directory of the current project file. The names of the directories will be stored in the project file as in older versions of UAG.

The following list shows the names of the directories which should be specified.

- The path for PLC projects
- The default documentation path
- The data server application path (for Generic HMI only)
- The path for SCoD libraries (*.osl)
- The path for XML style sheets (*.xsl) (for Generic HMI only)
- The path for help files for SCoDs (user defined, e.g. *.pdf)
- The path for archiving application data

In UAG, a new option dialogue has to be implemented and different UAG DLLs have to take care to use the files from the user-defined directory names.

The UAG system database files (*.osy, EmptyProject.osp, EmptyCustom.osc and EmptyLibrary.osl), currently located in the DB subdirectory of UAG's installation directory, should be moved to UAG's installation directory.

Upgrade of Existing Projects to UAG 2.1 and Concept V2.6

Automatic Upgrade

Unity Application Generator V2.1 is downward compatible to the existing product Unity Application Generator V2.0. Existing Unity Application Generator projects, customizations and SCoD libraries will be automatically updated to the new structure of V2.1.

Migration to Concept V2.6 SR1

If you additionally want to upgrade existing projects Concept V2.6 SR1, the Concept projects have to be updated first with the Concept converter. Follow the steps:

Step	Action
1	Export the Concept projects with Concept V2.2 or V2.5.
2	Install Concept V2.6.
3	Import Concept projects with Concept V2.6.
4	Install Unity Application Generator V2.0.

Note

Please read in any case the `Readme.txt` file delivered with your software!

Generated Code



B

Overview

Introduction

This chapter includes an overview and detailed information about the generated code and the generation principles.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
B.1	Overview of Generated Code and Generation Principles	275
B.2	Generation for Concept	278
B.3	Generation for Unity Pro	304
B.4	Generation for Monitor Pro	332
B.5	Generation for iFIX	344
B.6	Generation for a Generic HMI	363
B.7	Generation for Net Partners	364

B.1 Overview of Generated Code and Generation Principles

Overview

Introduction This section gives an overview of the generated code of Unity Application Generator and the generation principles.

What's in this Section? This section contains the following topics:

Topic	Page
Overview of Generated Code	276
Generation Principles	277

Overview of Generated Code

Introduction

Unity Application Generator generates

- code to be downloaded for the PLCs,
 - tags and graphics for Monitor Pro / iFIX
or
XML files with all tags and their attributes and a customziable text file for usage in any HMI (generic HMI) and
 - CSV files for import into Net Partners.
-

Generation for PLCs

For each PLC Unity Application Generator generates the following:

- The PLC configuration
- Code for program initialization
- Code for analog scaling
- Code for communication Channels
- Code for Control Modules
- All variables
- PLC memory managing

Note: The generated code will be in IEC 61131-3 standard FBD language.

Generation for Monitor Pro / iFIX HMI

For the HMI Monitor Pro / iFIX Unity Application Generator generates:

- Screens for Equipment Modules
 - All tags needed for the communication between HMI and PLC
 - All HMI internal tags
-

Generation for Generic HMI

For a generic HMI Unity Application Generator generates a XML file and a customziable text file for each Data Server containing all tags and their attributes necessary for process visualization.

Generation for Net Partners

For Net Partners Unity Application Generator generates a CSV file containing all variables related to a Net Partner.

Generation Principles

Introduction

Unity Application Generator is used to generate PLC and HMI applications as well as import files for Net Partners. The generated code has to be completed by the control engineer in order to build the complete control application.

The generation can be started separately for

- address calculation with the Memory Mapper,
- PLC,
- HMI and
- Net Partners.

Note: Generation includes automatically the address calculation by the Memory Mapper.


Generation Principles for PLC and HMI

When completing the control application, the control engineer has the choice of two modes of code generation in Unity Application Generator:

- New generation (available in options)
- Incremental generation (by default)

New generation: Unity Application Generator builds the entire application code from scratch. Existing code for the PLC and the HMI will be overwritten.

Incremental generation: Only the objects modified in Unity Application Generator will be updated. Additions made by the control engineer in the PLC and HMI logic will be left unchanged.

	CAUTION
	<p>Risk of losing code!</p> <p>Do not make any changes to the part of code generated by Unity Application Generator. Even for incremental generation this part of code will be overwritten. Only additional code is maintained.</p> <p>Failure to follow this precaution can result in injury or equipment damage.</p>

B.2 Generation for Concept

Overview

Introduction

This section describes the generation of Unity Application Generator for Concept.

What's in this Section?

This section contains the following topics:

Topic	Page
What is generated?	279
Generation from General Project Settings - Overview	280
Generation from the Topological Model - Overview	281
Generation from the Physical Model - Overview	287
Generated PLC Configuration	294
Generated Variables	295
Generated Code: Equipment Modules, Control Modules and Interlocks	297
Generated Code: Communication	300
Generated Code: Initialization	302
Generated Code: Scaling of Analog Values (Quantum only)	303
Generated Code: Hot Standby	303

What is generated?

What is generated?

For each PLC on the Topological Model, Unity Application Generator creates one Concept project.

For all elements of the Physical Model, Unity Application Generator creates special objects.

Generated objects from the Physical Model:

Element	Object
Area	Group in project browser
Process Cell	Group in project browser
Unit	Group in project browser
Equipment Module	FBD section with Function Blocks for all Control Modules
Control Module	Function Block with all inputs and outputs connected correctly with variables, literal values, or links Note: For type-less Control Modules nothing is generated!
Variables	Depending on the variables type, Unity Application Generator generates the respective Concept variables and an initialization section for the variables.

Generated objects from the Topological Model:

Element	Object
Channel	Section with logic for communication, Peer Cop or I/O scanner configuration
PLC	Concept project with the PLC configuration and the IO map.

Generation from General Project Settings - Overview

Introduction

The following tables gives an overview about the general settings in Unity Application Generator and the resulting settings in Concept.

Generation from General Project Settings

Generation from General Project Settings

Attribute	Concept	Comment
View → Options → PLC → PLC project path	Directory where Concept project is generated	Concept project is generated as <code>PLCProjectPath\PLCProjectName\PLCProjectName.PRJ</code> Example: <ul style="list-style-type: none">● <code>PLCProjectPath = c:\PLCPrjs</code>● <code>PLCProjectName = PLC1</code> generated project will be <ul style="list-style-type: none">● <code>c:\PLCPrjs\PLC1\PLC1.PRJ</code>
View → Options → PLC → Generate completely new	Concept project will be generated completely new.	

Generation from the Topological Model - Overview

Introduction

The following tables gives an overview about the settings in Unity Application Generator for the Topological Model and the resulting settings in Concept.

Generation for PLC

Generation for PLC

Attribute	Concept	Comment
Concept Project Name	Directory and name of Concept project	See <i>Generation from General Project Settings - Overview, p. 280</i>
CPU Type	PLC configuration → CPU	-
Address ranges	Addresses for <ul style="list-style-type: none"> ● Special registers ● I/O modules ● Variables ● Peer Cop ● I/O Scanner 	<p>Changing the addresses for HW modules will result in recalculation of the memory layout for the modules and thus all I/O variables.</p> <p>Changing the addresses for HMI will result in recalculation of the addresses of PLC_HMI variables.</p> <p>The Memory Mapper keeps the addresses, which are not affected by the change and recalculates only those addresses which are outside the old range.</p> <p>To view the different HMI communication frames, use the menu command Open HMI Communication for communication channels from type Data Server. If more than 100 % of a HMI communication frame is used, increase the number of HMI variables in the PLC properties, to prevent a recalculation of this HMI communication frame.</p>
Greatest Address 0x	PLC configuration → Memory range Coils	-
Greatest Address 1x	PLC configuration → Memory range Discrete Inputs	-
Greatest Address 3x	PLC configuration → Memory range Input Registers	-
Greatest Address 4x	PLC configuration → Memory range Holding Registers	-

Attribute	Concept	Comment
BatteryCoil	PLC configuration → Special: Battery Coil	-
TimerRegister	PLC configuration → Special: Timer Register	-
TimeOfDayStart	PLC configuration → Special:Time of Day	-
Comment	Project Comment	-

Generation for a Rack

Generation for a rack

Object	Concept	Comment
Rack	PLC configuration → I/O map → Remote or Distributed Drops	Exception: For racks of the following link types no racks in the I/O map are generated. <ul style="list-style-type: none">● Momentum Modbus Plus I/O● Momentum Ethernet I/O● Generic

Generation for rack attributes

Attribute	Concept	Comment
Number	Number of drop	-

Generation for a HW Module

Generation for a HW module

Object	Concept	Comment
Quantum HW Module	PLC configuration → I/O map → Module	The module is entered to the I/O map.
Momentum HW Module (Local Rack)	PLC configuration → I/O map → Module	The module is entered to the I/O map.
Momentum I/O Bus Module	PLC configuration → I/O map → Module	The module is entered to the I/O map.
Momentum Ethernet I/O Module	PLC configuration → I/O Scanner	For this type of modules no entry to the I/O map is made. The module is set up in the I/O Scanner.
Momentum Modbus Plus I/O Module	PLC configuration → Peer Cop	For this type of modules no entry to the I/O map is made. The module is set up in the Peer Cop.
Generic I/O Module	-	For this type of modules no entry to the I/O map is made.

Generation for the attributes of a HW module

Attribute	Concept	Comment
Rack	PLC configuration → I/O map → Drop of HW Module	HW modules can be moved between racks by drag and drop in the Unity Application Generator HW modules table within the same PLC, except Modbus Plus I/O, Ethernet I/O and IO Bus.
Slot Number	PLC configuration → I/O map → Slot of HW Module	HW modules can be moved within a rack by drag and drop in the Unity Application Generator HW modules table
Timeout state	PLC configuration → I/O map → HW Module Parameter → Timeout state	Applies only to digital output modules. Timeout state of the HW module must be same as the timeout state of the variables mapped to the module.
Timeout value	PLC configuration → I/O map → HW Module → Parameter → Timeout value	Applies only to digital output modules with timeout state = User defined. The timeout value of the module is calculate by the timeout states of the variable mapped to that module.
Addresses	PLC configuration → I/O map → HW Module → Addresses	The State RAM addresses of the I/O modules are calculated from the information entered in PLC → Addresses in Unity Application Generator.

Note: For generic modules **no** entries are generated in the I/O map!

Generation for a PLC - PLC Channel

Generation for a PLC - PLC Channel

Object	Concept	Comment
PLC Channel	PLC configuration → Peer Cop entry/ I/O Scanner	One or two entries (depending on the number of communication paths specified in Unity Application Generator) are generated in the Peer Cop/ I/O Scanner table according to the network addresses of the PLCs involved in the communication. The 4x addresses used by the communication are calculated from the information entered in PLC → Addresses for PLC communication in Unity Application Generator.
	FBD logic for Peer Cop	A FBD section with the name of the Channel is generated in a group called <i>Communication</i> . The logic takes care to pack and unpack the communicated variables into the 4x memory range used for the Peer Cop/ I/O Scanner. It also takes care to switch to the second communication path if the first fails (if a second path is specified) and to use the failure values for the variables if the communication fails completely.
	Variables	All variables specified in the PLC communication table are generated in receiving PLC (Concept project) with the same name as in the sending PLC (the scope of the variable names is the complete plant with all PLCs). The generated FBD logic takes care to set the variables to the correct values communicated by Peer Cop/ I/O Scanner or to predefined failure values.

Generation for the attributes of a PLC - PLC Channel

Attribute	Concept	Comment
Name	Name of the section with the communication logic	With Ethernet communication for the receiving PLC 2 sections will be generated, called Channelname and Channelname_2.
Comment	Comment of the section with the communication logic	-
Communication partner	Either <ul style="list-style-type: none"> ● all (Global, broadcast - only Modbus Plus) or ● another PLC (Specific, peer-to-peer communication) 	The communication partner cannot be changed after the Channel has been created in Unity Application Generator.
Communication path(s)	Peer Cop nodes or I/O Scanner entries	The network address of the communication partner specifies the possible communication paths and therefore also the link and the node addresses in the Peer Cop/ I/O Scanner.
Timeout(s)	The duration literals connected to special send and receive Function Blocks in the Peer Cop logic	The timeout(s) specifies the supervision time of the communication before the communication logic switches to the second communication path or failure values. If there is no timeout specified the generated logic has a default timeout of 2 seconds.

Generation for a PLC-PLC Channel - Other Objects/ Attributes that influence a PLC-PLC Channel

Generation for a PLC-PLC Channel, other objects/attributes that influence a PLC-PLC Channel

Attribute	Concept	Comment
PLC communication table → Variables	Variables that will be generated in the receiving PLC(s) and communicated via Peer Cop or I/O Scanner	Variables of data type BOOL are packed into Words (max. 16), Variables of other data types are expanded in one or two words.
PLC communication table → Variables → Failure state	Specifies how the communication logic will handle communication failures	Last Value: <ul style="list-style-type: none"> ● in case of communication failure the variable will hold its last value. User defined: <ul style="list-style-type: none"> ● in case of communication failure the variable will be set to value specified as failure value. Not_Assigned: <ul style="list-style-type: none"> ● in case of communication failure the variable will be set to 0.
PLC communication table → Variables → Failure value	The value to which variables are set in case of communication failure	Applies only if failure state = User defined.
Network address of CPU, NOM, NOE, PNN	Node addresses in Peer Cop or I/O Scanner	The Modbus Plus network address of the module specifies the node addresses used in the Peer Cop. The Ethernet network address of the module specifies the node addresses used in the I/O Scanner.
PLC → Addresses → PLC communication	4x addresses used by Peer Cop/ I/O Scanner	The 4x addresses used by the communication are calculated from the information entered in PLC → Addresses for PLC communication in Unity Application Generator.

Generation from the Physical Model - Overview

Introduction

The following tables gives an overview about the settings in Unity Application Generator for the Physical Model and the resulting settings in Concept.

Generation for an Area

Generation for the Area

Object	Concept	Comment
Area	Concept Project Browser → Group	An Area group is a top level group in the Concept Project Browser. The group is generated in those Concept projects where there is an Equipment Module or a Control Module in the Physical Model hierarchy of Unity Application Generator which belongs to this PLC. The group can therefore exist in several PLCs.

Generation for the attributes of an Area

Attribute	Concept	Comment
Name	Group name	

Generation for a ProcessCell

Generation for a Process Cell

Object	Concept	Comment
Process Cell	Concept Project Browser → Group	A Process Cell group is a group within the parent Area group. The group is generated in those Concept projects where there is an Equipment Module or a Control Module in the Physical Model hierarchy of Unity Application Generator which belongs to this PLC. The group can therefore exist in several PLCs.

Generation for the attributes of a Process Cell

Attribute	Concept	Comment
Name	Group	
Area	Parent Group	A Process Cell can be moved to another Area by drag and drop in the Physical Model of Unity Application Generator.

Generation for a Unit

Generation for a Unit

Object	Concept	Comment
Unit	Group	A Unit group is a group within the parent Process Cell group. The group is generated in those Concept projects where there is an Equipment Module or a Control Module in the Physical Model hierarchy of Unity Application Generator which belongs to this PLC. The group can therefore exist in several PLCs.

Generation for the attributes of a Unit

Attribute	Concept	Comment
Name	Group	
Process Cell	Parent Group	A Unit can be moved to another Process Cell by drag and drop in the Physical Model of Unity Application Generator.

Generation for an Equipment Module

Generation for an Equipment Module

Object	Concept	Comment
Equipment Module	Concept FBD section	An Equipment Module is a section within the parent Unit group in the Concept Project Browser. Always the complete group hierarchy Unit - Process Cell - Area is generated.

Generation for the attributes of an Equipment Module

Attribute	Concept	Comment
Name	The name of the Equipment Module is part of: <ul style="list-style-type: none"> ● Function block instance names ● Equipment variable names ● Control Module variable names 	Changing the name of an Equipment Module result in changing the name of <ul style="list-style-type: none"> ● all Function Block instance names of its Control Modules ● all names of the variables of the Equipment Module ● all names of the variables of the Control Modules of the Equipment Module.
Section name	Concept section name	
Unit	Parent Unit group of section	An Equipment Module can be moved to another Unit by drag and drop in the Physical Model of Unity Application Generator.
PLC	Concept project to which the equipment belongs	The PLC to which the Equipment Module is assigned cannot be changed if the Equipment Module has been generated once.
Description, Comment	Concept section comment	The Concept section comment is the concatenation of the Equipment Module description and Equipment Module comment.
Description	The description of the Equipment Module is part of the comment of all variables of the Equipment Module	

Generation for a Control Module

Generation for a Control Module

Object	Concept	Comment
Control Module	<p>Concept Function Block instance</p> <p>Note: For type-less Control Modules nothing is generated. Analyzer prints a warning.</p>	<p>The Concept Function Block instance name is generated as 'EquipmentModuleName_ControlModuleName'. If a Control Module is assigned to a different PLC than the parent Equipment Module, the Equipment Module section will also be generated in the Concept project (PLC) to which the Control Module is assigned.</p>

Generation for the attributes of a Control Module

Attribute	Concept	Comment
Parent Equipment Module	<ul style="list-style-type: none"> Section in which the Function Block instance will be generated. Name of Function Block instance Names of the variables of the Control Module 	<p>A Control Module can be moved to another Equipment Module by drag and drop in the Physical Model of Unity Application Generator.</p>
PLC	in which project	<p>The PLC to which the Control Module is assigned cannot be changed if Control Module has been generated once.</p>
Name	Name for FB instance and ControlModuleVar	
Description, Comment	Comment of Function Block instance	<p>The Function Block instance comment is the concatenation of the Control Module description and Control Module comment.</p>
Description	The description of the Control Module is part of the comment of all variables of the Control Module	

Attribute	Concept	Comment
Interlock definition	<ul style="list-style-type: none"> Function Block network connected to the Control Module's Function Block The textual description of the interlock definition as a comment in the Control Modules's corresponding Function Block. 	<p>The interlock definition has to be syntactically and semantically correct. This means:</p> <ul style="list-style-type: none"> It has to be a valid logical expression. The variables used have to be existing locally. <p>Otherwise only a comment is generated to the Control Module's Function Block.</p> <p>Note: Existing interlock definitions from older versions of Unity Application Generator (<V1.5) will stay as comments.</p>

Generation for a Variable

Generation for a variable

Object	Concept	Comment
Variable	Concept variable	<p>The variable name is generated as 'EquipmentModuleName_ControlModuleName_VvariableName' (Control Module variable) or 'EquipmentModuleName_VariableName' (Equipment Module).</p> <p>The variable comment is generated as 'VariableDescription - ControlModuleDescription - Equipment ModuleDescription' (Control Module variable) or 'VariableDescription' - EquipmentModuleDescription' (Equipment Module variable).</p>

Generation for the attributes of a variable

Attribute	Concept	Comment
Name	Variable Name	The name can only be changed for free variables.
Description	Part of variable comment	See above
InitialValue	Initial value of variable	Initial values can be specified for: <ul style="list-style-type: none"> ● unlocated variables (PLC or PLC_NET). ● located 0x variables (PLC_HMI), since Concept does not accept initial values for 0x located variables, special sections are generate to assign a value of 1 to 0x variables as part of the startup of the PLC (Concept sections 'InitControl', 'InitCoils1', 'InitCoils2', etc in the group 'Initialization'.
Timeout state	Used for setting the Timeout state of the I/O module, to which the variable is mapped. Timeout state of the I/O module must be same as the Timeout state of the variables mapped to the module.	Applies only to IO_PLC digital output variables.
Timeout value	Used for calculating the Timeout value of the I/O module, to which the variable is mapped	Applies only to IO_PLC digital output variables with timeout state = User defined.

Attribute	Concept	Comment
Command	Defines how the variable is connected to the Function Block instance	<p>Applies only to PLC_HMI variables. Free variables are always IN/OUT, see below.</p> <p>IN variables: Always connected to input of Function Block instance</p> <ul style="list-style-type: none"> ● Operator/Parameter The variable can be changed by the HMI. ● Logic/Constant The variable cannot be changed by the HMI. ● OUT variables Always connected to output of Function Block instance always View only. <p>IN/OUT variables</p> <ul style="list-style-type: none"> ● Operator/Parameter: The variable is connected to the input and the output of the Function Block instance. ● Constant: The input of the Function Block instance is connected with the initial value of the variable, the output is connected with variable. ● Logic: The input of the Function Block instance is not connected, the output is connected with variable.
Variable used	Specifies if a variables is generated and connected to the Function Block instance.	Applies only to IO_PLC variables. Other variables and free variables are always used.
Invert	Creates invert functionality of pin or creates literal if an additional pin is used to invert this IO point.	<p>Applies only to IO_PLC variables of data type BOOL.</p> <p>Depending on the Control Module type the input/output of Function Block instance is either inverted or if the Control Module Type has a special input to specify the inverted logic, a literal 0 or 1 (inverted) is connected to this input.</p>
State RAM address	Variable addresses	<ul style="list-style-type: none"> ● IO_PLC variables: The addresses are calculated from addresses of the I/O modules to which the variables are assigned. ● PLC variables: Generated as unlocated variables. ● PLC_NET variables: The address are calculated from the information entered in PLC → Addresses for HMI in Unity Application Generator.

Generated PLC Configuration

What is Generated?

Unity Application Generator will generate all configuration information which is in the scope of Unity Application Generator and what is necessary to describe the high level process design. It will **not** generate e. g.:

- ASCII setup information
- Segment scheduler information
- Specific module parameters

PLC Configuration

In the Concept configurator, Unity Application Generator will take care for the following aspects:

- PLC type
- Memory partitions
- I/O map
- Peer Cop and I/O Scanner
- Battery coils
- Timer register
- Time of day register

I/O Map

Unity Application Generator will generate for Quantum PLCs the local rack and remote racks or distributed racks, SY/MAX and 800 I/O racks. For Momentum PLCs the Unity Application Generator will generate the local rack and the I/O Bus rack. The HW modules will be generated in the correct racks and slots and the PLC addresses will be assigned.

Note:

- For the following racks and modules no entries are generated in the I/O map:
 - Modbus Plus racks and I/O modules
 - Ethernet racks and I/O modules
 - generic racks and generic modules
- ModConnect partner modules are generated in the I/O map. They have to be imported into Concept with ModConnect Tool and created in the Unity Application customization editor as user defined module.
- Not all parameter options which are possible for hardware Modules will be set up by Unity Application Generator. These alternative options have to be defined manually in Concept. These options will not be overwritten in an incremental generation.

Generated Variables

What is Generated?

Most variables in Unity Application Generator will generate variables in Concept. Only variables of the type HMI will not generate a variable in Concept but a tag in the HMI data base.

Variables

Each variable in Unity Application Generator of the connection types listed in the following table will result in a variable in Concept.

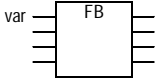
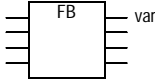
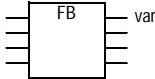
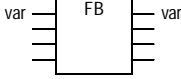
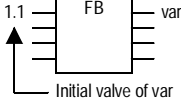
Note: IO_PLC variables which are declared `not used` will not be generated in Concept.

Connection types and related variables in Concept:

Connection Type	Variable type in Concept	Comment
IO_PLC	Located variable	The address will be the I/O point to which the variable is mapped. In the Concept logic the variable will be connected to the input/output of the FB. Free variables will not be connected.
PLC	Unlocated variable	In the Concept logic the variable will be connected to the input/output of the FB of the Control Module. Free variables will not be connected.
PLC_HMI	Located variable	In the Concept logic the variable will be connected to the input/output of the FB of the Control Module, for details see table below.
PLC_NET	Located variable	In Concept located variables will be generated.

Note: Free variables will be generated in Concept without any logic.

Relation of In/Out definition, command type and generated connection of the variable to the FB for PLC_HMI Control Module type variables (the variable is named "var"):

In/Out	Command	Generated
In	Operator Parameter	
Out	View only	
In/Out	Logic	
	Operator Parameter	
	Constant	

Generated Code: Equipment Modules, Control Modules and Interlocks

What is Generated?

For each Equipment Module Unity Application Generator generates a section in Concept. This section will contain a Function Block for each Control Module of the Equipment Module. Each Control Module has its own specific Function Block. For details on the Function Blocks refer to the Control Module Type (SCoD) documentation.

All variables required by the Concept project will also be created.

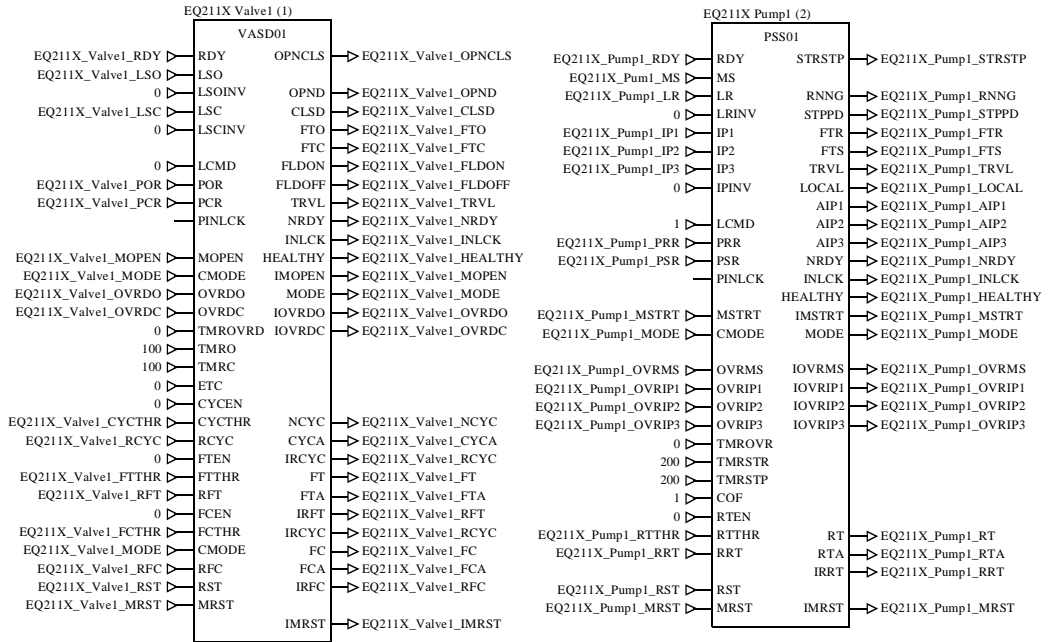
For an interlock definition the following is generated:

For ...	Unity Application Generator generates ...
A syntactically and semantically correct interlock definition	A Function Block network realizing the interlock definition connected to the corresponding SCoD
An incorrect interlock definition	A comment for the corresponding SCoD

Note: Please note the following information on interlocks:

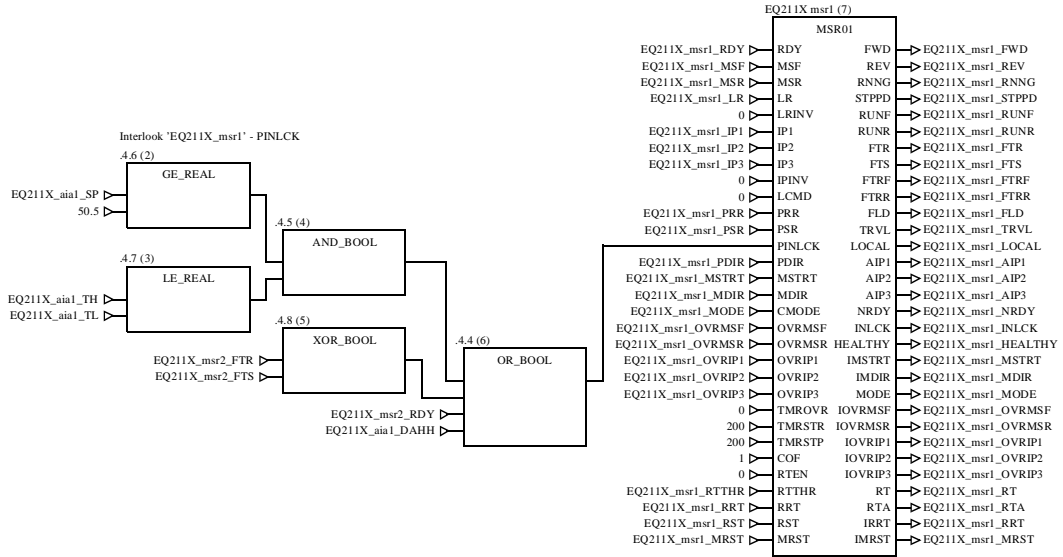
- Syntactically correct interlock definition means: It is a valid logical expression.
- Semantically correct interlock definition means: All used variables are existing on the PLC where the SCoD is generated.
- The generated Concept Function Block networks in some cases have to be rearranged manually.
- Existing interlock definitions from older versions of Unity Application Generator (<V1.5) will stay as comments.
- For interlock definitions as comments the control engineer has to add the PLC code manually.

Example Section Section generated for an Equipment Module containing 2 Control Modules:



**Example:
Interlock**

Function Block generated for a Control Module connected to a Function Block network generated for an interlock definition:



Note: See the corresponding interlock definition in Unity Application Generator in *Example: Interlock Definition, p. 150*

Generated Code: Communication

What is Generated? What is generated depends of the type of communication.
PLC <-> PLC communication is performed as Peer Cop or I/O Scanner communication.

Code for Peer Cop Communication Peer Cop communication is restricted to 32 words. Unity Application Generator uses 1 word for a watchdog counter, thus 31 words remain for the communication.
Unity Application Generator builds up logic to

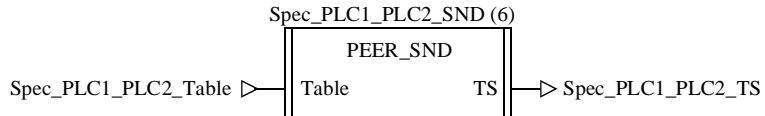
- assemble in a data table all PLC variables to be communicated into one structured variable and to
- disassemble for the receiving PLC the structured variable.

Unity Application Generator will configure the Peer Cop tables with memory and network addresses.

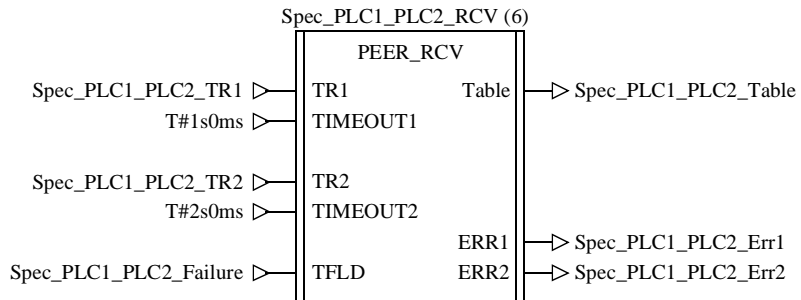
Code for I/O Scanner Communication I/O Scanner communication is very similar to Peer Cop communication.
The differences are the following:

- I/O Scanner communication is restricted to 100 words.
- I/O Scanner tables need only an entry in the sending PLC:

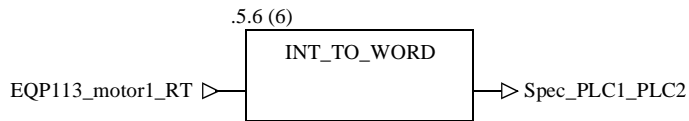
Example of Generated Function Blocks The following figures show examples of the Function Blocks generated for Peer Cop communication.
Function block to send Peer Cop table to another PLC



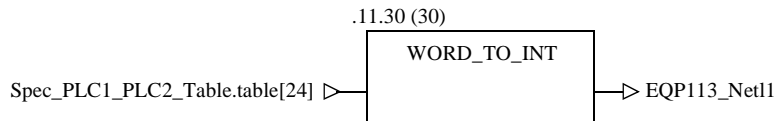
Function Block to receive Peer Cop table from another PLC



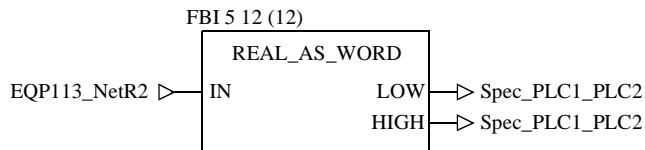
Convert datatype INT to WORD



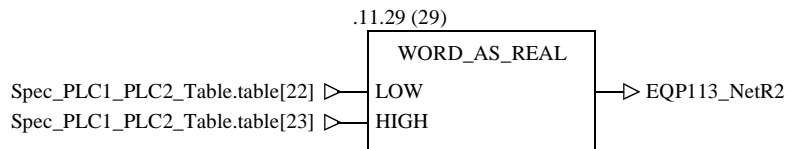
Convert datatype WORD to INT



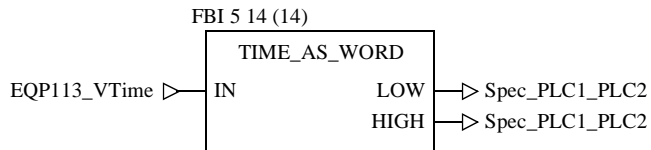
Convert REAL as WORD (2 words)



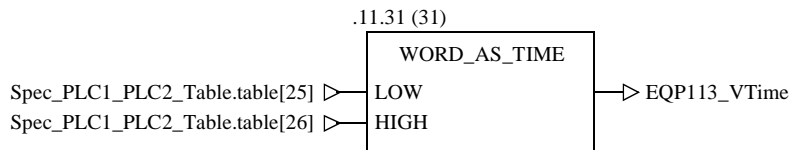
Convert WORD as REAL



Convert TIME as WORD (2 words)



Convert WORD as TIME (2 words)

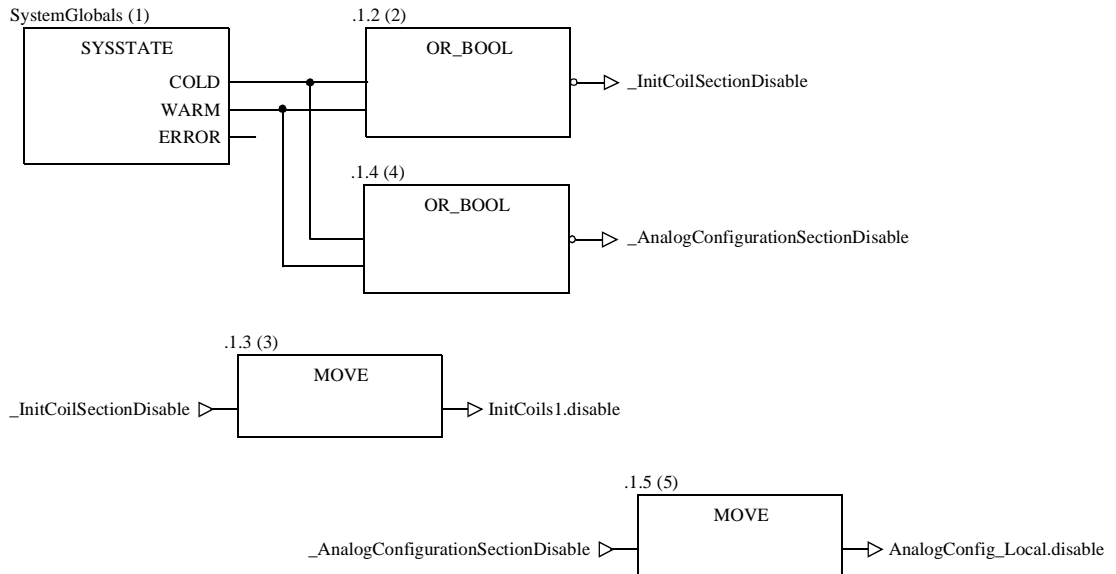


Generated Code: Initialization

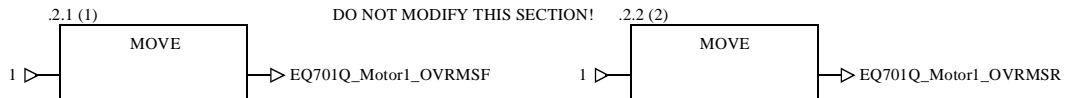
What is Generated?

The initialization sections takes care to set all the 0x variables to the initial values defined by Unity Application Generator. Additionally there is an FBD section with logic that ensures that the initialization is only executed if the PLC has to perform cold start-up or warm start-up.

Example Code Initialization control code



Initialization of Coils



Generated Code: Scaling of Analog Values (Quantum only)

What is Generated?

With Quantum PLCs for the scaling of analog values the following is generated:

- A group called `AnalogConfiguration`
- In this group further groups for the different I/O types (local, remote, distributed)
- Within these groups a section for each Rack

Examples: Section `AnalogConfig_Remote_Drop6` or
`AnalogConfig_DI01_Drop16`

Within these sections a DROP Function Block, (a XBP Function Block if a backplane expander is used) and the Function Blocks for the analog modules are generated and connected appropriately. The variables mapped to analog modules are connected to the respective EFBs (only for data types `ANL_IN` and `ANL_OUT`).

<p>Note: All generated names are independent of the installed language version of Unity Application Generator. Thus the Concept projects can be used with all Unity Application Generator language versions.</p>

Generated Code: Hot Standby

What is Generated?

For a Hot Standby system the following is generated:

- A group called `Hot Standby` (always the first group in the Concept project).
 - In this group a section called `HotStandby-Section`.
 - In this section Function Blocks for monitoring and manipulating the HSBY system. Variables are generated and connected to these Function Blocks.
-

B.3 Generation for Unity Pro

Overview

Introduction

This section describes the generation of Unity Application Generator for Unity Pro.

What's in this Section?

This section contains the following topics:

Topic	Page
What is generated?	305
Generation from General Project Settings - Overview	306
Generation from the Topological Model - Overview	307
Generation from the Physical Model - Overview	313
Generated PLC Configuration	320
Generated Variables	320
Generated Code: Equipment Modules, Control Modules and Interlocks	323
Generated Code: Communication	326
Generated Code: Initialization (Quantum only)	328
Generated Code: Scaling of Analog Values (Quantum only)	328
Generated Code: Discrete Configuration (Premium only)	329

What is generated?

What is generated?

For each PLC on the Topological Model, Unity Application Generator creates one Unity Pro project.

For all elements of the Physical Model, Unity Application Generator creates special objects.

Generated objects from the Physical Model:

Element	Object
Area	Functional Module in project browser
Process Cell	Functional Module in project browser
Unit	Functional Module in project browser
Equipment Module	FBD section with Function Blocks for all Control Modules
Control Module	Function Block with all inputs and outputs connected correctly with variables, literal values, or links Note: For type-less Control Modules nothing is generated!
Variables	Depending on the variables type, Unity Application Generator generates the respective Unity Pro variables and if necessary an initialization section for the variables.

Generated objects from the Topological Model:

Element	Object
Channel	Section with logic for communication, Peer Cop or I/O scanner configuration
PLC	Unity Pro project with the PLC configuration.

Generation from General Project Settings - Overview

Introduction

The following tables gives an overview about the general settings in Unity Application Generator and the resulting settings in Unity Pro.

Generation from General Project Settings

Generation from General Project Settings

Attribute	Unity Pro	Comment
View → Options → PLC → PLC project path	Directory where Unity Pro project is generated	Unity Pro project is generated as <code>PLCProjectPath\PLCProjectName\PLCProjectName.STU</code> Example: <ul style="list-style-type: none"> ● <code>PLCProjectPath = c:\PLCPrjs</code> ● <code>PLCProjectName = PLC1</code> generated project will be <ul style="list-style-type: none"> ● <code>c:\PLCPrjs\PLC1\PLC1.STU</code>
View → Options → PLC → Generate completely new	Unity Pro project will be generated completely new.	

Note: In the Unity Pro under **Tools** → **Project Settings...** → **Language extensions** the options **Allow leading digits** and **Allow extended character set** will be set.

Generation from the Topological Model - Overview

Introduction

The following tables gives an overview about the settings in Unity Application Generator for the Topological Model and the resulting settings in Unity Pro.

Generation for PLC

Generation for PLC

Attribute	Unity Pro	Comment
PLC Project Name	Directory and name of PLC project	See <i>Generation from General Project Settings - Overview, p. 306</i>
CPU Type	Project Browser (Structural View) → Station → Configuration	-
Address ranges	Addresses for <ul style="list-style-type: none"> ● Special registers ● I/O modules ● Variables ● Peer Cop ● I/O Scanner 	<p>IO_PLC variables are mapped in Unity Pro to topological addresses of type EBOOL. (In Unity Application Generator they are displayed with type BOOL.)</p> <p>PLC_HMI variables are mapped in Unity Pro to StateRAM addresses of type %MW.</p> <p>Changing the addresses for Quantum HW modules will result in recalculation of the memory layout for the modules and thus all I/O variables. Changing the addresses for HMI will result in recalculation of the addresses of PLC_HMI variables.</p> <p>The Memory Mapper keeps the addresses, which are not affected by the change and recalculates only those addresses which are outside the old range.</p> <p>To view the different HMI communication frames, use the menu command Open HMI Communication for communication channels from type Data Server. If more than 100 % of a HMI communication frame is used, increase the number of HMI variables in the PLC properties, to prevent a recalculation of this HMI communication frame.</p>
Greatest Address %M (0x)	Select CPU Open Module → Configuration	-

Attribute	Unity Pro	Comment
Greatest Address %I (1x)	Select CPU Open Module → Configuration	-
Greatest Address %IW (3x)	Select CPU Open Module → Configuration	-
Greatest Address %MW (4x)	Select CPU Open Module → Configuration	-
Comment	Project Browser (Structural View) → Station → Properties → Comment	-

Generation for a HW Module

Generation for a HW module

Object	Unity Pro	Comment
Quantum/ Premium HW Module	-	The modules are entered in Unity Pro.
Momentum Ethernet I/O Module	Project Browser (Structural View) → Station → Communication → Networks → Network → Open → I/O Scanner	For this type of modules no entry to the I/O map is made. The module is set up in the I/O Scanner.
Momentum Modbus Plus I/ O Module	Project Browser (Structural View) → Station → Communication → Networks → Network → Open	For this type of modules no entry to the I/O map is made. The module is set up in the Peer Cop.
Generic I/O Module	-	For this type of modules no I/O mapping is made.

Generation for the attributes of a HW module

Attribute	Unity Pro	Comment
Timeout state	Select module Open Module → Config → TIMEOUT STATE	Applies only to digital output modules. Timeout state of the HW module must be same as the timeout state of the variables mapped to the module.
Timeout value	Select module Open Module → Config → TIMEOUT STATE → VALUE	Applies only to digital output modules with timeout state = User defined. The timeout value of the module is calculated by the timeout states of the variable mapped to that module.
Addresses	Select module Open Module → Config → ADDRESSES	The State RAM addresses of the I/O modules are calculated from the information entered in PLC → Addresses in Unity Application Generator.

Note: For generic modules **no** entries are generated in the I/O map!

**Generation for a
PLC - PLC
Channel**

Generation for a PLC - PLC Channel

Object	Unity Pro	Comment
PLC Channel	Project Browser (Structural View) → Station → Communication → Networks → Network → Open → Peer Cop/ I/O Scanner	<p>One or two entries (depending on the number of communication paths specified in Unity Application Generator) are generated in the Peer Cop / I/O Scanner table according to the network addresses of the PLCs involved in the communication.</p> <p>The %MW addresses used by the communication are calculated from the information entered in PLC → Addresses for PLC communication in Unity Application Generator.</p>
	FBD logic for Peer Cop	<p>A FBD section with the name of the Channel is generated in a group called <i>Communication</i>.</p> <p>The logic takes care to pack and unpack the communicated variables into the %MW memory range used for the Peer Cop I/O Scanner. It also takes care to switch to the second communication path if the first fails (if a second path is specified) and to use the failure values for the variables if the communication fails completely.</p>
	Variables	<p>All variables specified in the PLC communication table are generated in receiving PLC (Unity Pro project) with the same name as in the sending PLC (the scope of the variable names is the complete plant with all PLCs).</p> <p>The generated FBD logic takes care to set the variables to the correct values communicated by Peer Cop / I/O Scanner or to predefined failure values.</p>

Generation for the attributes of a PLC - PLC Channel

Attribute	Unity Pro	Comment
Name	Name of the section with the communication logic	With Ethernet communication for the receiving PLC 2 sections will be generated, called Channelname and Channelname_2.
Comment	Comment of the section with the communication logic	-
Communication partner	Either <ul style="list-style-type: none"> ● all (Global, broad cast - only Quantum) or ● another PLC (Specific, peer-to-peer communication) 	The communication partner cannot be changed after the Channel has been created in Unity Application Generator.
Communication path(s)	Peer Cop nodes or I/O Scanner entries	The network address of the communication partner specifies the possible communication paths and therefore also the link and the node addresses in the Peer Cop/ I/O Scanner.
Timeout(s)	The duration literals connected to special send and receive Function Blocks in the Peer Cop logic	The timeout(s) specifies the supervision time of the communication before the communication logic switches to the second communication path or failure values. If there is no timeout specified the generated logic has a default timeout of 2 seconds.

Generation for a PLC-PLC Channel - Other Objects/ Attributes that influence a PLC-PLC Channel

Generation for a PLC-PLC Channel, other objects/attributes that influence a PLC-PLC Channel

Attribute	Unity Pro	Comment
PLC communication table → Variables	Variables that will be generated in the receiving PLC(s) and communicated via Peer Cop or I/O Scanner	Variables of data type BOOL are packed into Words (max. 16), Variables of other data types are expanded in one or two words.
PLC communication table → Variables → Failure state	Specifies how the communication logic will handle communication failures	Last Value: <ul style="list-style-type: none"> ● in case of communication failure the variable will hold its last value. User defined: <ul style="list-style-type: none"> ● in case of communication failure the variable will be set to value specified as failure value. Not_Assigned: <ul style="list-style-type: none"> ● in case of communication failure the variable will be set to 0.
PLC communication table → Variables → Failure value	The value to which variables are set in case of communication failure	Applies only if failure state = User defined.
Network address of CPU, NOM, NOE, M1, ETY, etc.	Node addresses in Peer Cop or I/O Scanner	The Modbus Plus network address of the module specifies the node addresses used in the Peer Cop. The Ethernet network address of the module specifies the node addresses used in the I/O Scanner.
PLC → Addresses → PLC communication	%MW addresses used by Peer Cop/ I/O Scanner	The %MW addresses used by the communication are calculated from the information entered in PLC → Addresses for PLC communication in Unity Application Generator.

Generation from the Physical Model - Overview

Introduction

The following tables gives an overview about the settings in Unity Application Generator for the Physical Model and the resulting settings in Unity Pro.

Generation for Unity Application Generator

Generation for the Unity Application Generator

Object	Unity Pro	Comment
UAG Site	Project Browser	UAG Site is a top level folder in the Unity Pro Project Browser in the Functional view.

Generation for an Area

Generation for the Area

Object	Unity Pro	Comment
Area	Project Browser → UAG Site → Area Functional module	An Area Functional module is a folder within the parent UAG Site folder. The folder is generated in those Unity Pro projects where there is an Equipment Module or a Control Module in the Physical Model hierarchy of Unity Application Generator which belongs to this PLC. The group can therefore exist in several PLCs.

Generation for the attributes of an Area

Attribute	Unity Pro	Comment
Name	Functional module name	-

Generation for a ProcessCell

Generation for a Process Cell

Object	Unity Pro	Comment
Process Cell	Project Browser → UAG Site → Area Functional module → Process Cell Functional module	A Process Cell Functional module is a folder within the parent Area folder. The folder is generated in those Unity Pro projects where there is an Equipment Module or a Control Module in the Physical Model hierarchy of Unity Application Generator which belongs to this PLC. The group can therefore exist in several PLCs.

Generation for the attributes of a Process Cell

Attribute	Unity Pro	Comment
Name	Functional module name	-
Area	Parent Functional module	A Process Cell can be moved to another Area by drag and drop in the Physical Model of Unity Application Generator.

Generation for a Unit

Generation for a Unit

Object	Unity Pro	Comment
Unit	Project Browser → UAG Site → Area Functional module → Process Cell Functional module → Unit Functional module	A Unit Functional module is a folder within the parent Process Cell folder. The folder is generated in those Unity Pro projects where there is an Equipment Module or a Control Module in the Physical Model hierarchy of Unity Application Generator which belongs to this PLC. The group can therefore exist in several PLCs.

Generation for the attributes of a Unit

Attribute	Unity Pro	Comment
Name	Functional module name	-
Process Cell	Parent Functional module	A Unit can be moved to another Process Cell by drag and drop in the Physical Model of Unity Application Generator.

Generation for an Equipment Module

Generation for an Equipment Module

Object	Unity Pro	Comment
Equipment Module	Unity Pro FBD section	An Equipment Module is a section within the parent Unit folder in the Unity Pro Project Browser in the functional view. Always the complete folder hierarchy Unit - Process Cell - Area is generated.

Generation for the attributes of an Equipment Module

Attribute	Unity Pro	Comment
Name	The name of the Equipment Module is part of: <ul style="list-style-type: none"> Function block instance names Equipment variable names Control Module variable names 	Changing the name of an Equipment Module result in changing the name of <ul style="list-style-type: none"> all Function Block instance names of its Control Modules all names of the variables of the Equipment Module all names of the variables of the Control Modules of the Equipment Module.
Section name	Unity Pro section name	
Unit	Parent Unit folder of section	An Equipment Module can be moved to another Unit by drag and drop in the Physical Model of Unity Application Generator.
PLC	Unity Pro project to which the equipment belongs	The PLC to which the Equipment Module is assigned cannot be changed if the Equipment Module has been generated once.
Description, Comment	Unity Pro section comment	The Unity Pro section comment is the concatenation of the Equipment Module description and Equipment Module comment.
Description	The description of the Equipment Module is part of the comment of all variables of the Equipment Module	

Generation for a Control Module

Generation for a Control Module

Object	Unity Pro	Comment
Control Module	Unity Pro Function Block instance Note: For type-less Control Modules nothing is generated. Analyzer prints a warning.	The Unity Pro Function Block instance name is generated as 'EquipmentModuleName_ControlModuleName'. If a Control Module is assigned to a different PLC than the parent Equipment Module, the Equipment Module section will also be generated in the Unity Pro project (PLC) to which the Control Module is assigned.

Generation for the attributes of a Control Module

Attribute	Unity Pro	Comment
Parent Equipment Module	<ul style="list-style-type: none"> Section in which the Function Block instance will be generated. Name of Function Block instance Names of the variables of the Control Module 	A Control Module can be moved to another Equipment Module by drag and drop in the Physical Model of Unity Application Generator.
PLC	in which project	The PLC to which the Control Module is assigned cannot be changed if Control Module has been generated once.
Name	Name for FB instance and ControlModuleVar	
Description, Comment	Comment of Function Block instance	The Function Block instance comment is the concatenation of the Control Module description and Control Module comment.
Description	The description of the Control Module is part of the comment of all variables of the Control Module	

Attribute	Unity Pro	Comment
Interlock definition	<ul style="list-style-type: none"> Function Block network connected to the Control Module's Function Block The textual description of the interlock definition as a comment in the Control Modules's corresponding Function Block. 	<p>The interlock definition has to be syntactically and semantically correct. This means:</p> <ul style="list-style-type: none"> It has to be a valid logical expression. The variables used have to be existing locally. <p>Otherwise only a comment is generated to the Control Module's Function Block.</p> <p>Note: Existing interlock definitions from older versions of Unity Application Generator (<V1.5) will stay as comments.</p>

Generation for a Variable

Generation for a variable

Object	Unity Pro	Comment
Variable	Unity Pro variable	<p>The variable name is generated as 'EquipmentModuleName_ControlModuleName_VvariableName' (Control Module variable) or 'EquipmentModuleName_VariableName' (Equipment Module).</p> <p>The variable comment is generated as 'VariableDescription - ControlModuleDescription - Equipment ModuleDescription' (Control Module variable) or 'VariableDescription' - EquipmentModuleDescription' (Equipment Module variable).</p>

Generation for the attributes of a variable

Attribute	Unity Pro	Comment
Name	Variable Name	The name can only be changed for free variables.
Description	Part of variable comment	See above
InitialValue	Initial value of variable	Initial values can be specified for: <ul style="list-style-type: none">● unlocated variables (PLC or PLC_NET).● located %MW variables (PLC_HMI or PLC_NET).
Timeout state	Used for setting the Timeout state of the I/O module, to which the variable is mapped. Timeout state of the I/O module must be same as the Timeout state of the variables mapped to the module.	Applies only to IO_PLC digital output variables.
Timeout value	Used for calculating the Timeout value of the I/O module, to which the variable is mapped	Applies only to IO_PLC digital output variables with timeout state = User defined.

Attribute	Unity Pro	Comment
Command	Defines how the variable is connected to the Function Block instance	<p>Applies only to PLC_HMI variables. Free variables are always IN/OUT, see below.</p> <p>IN variables: Always connected to input of Function Block instance</p> <ul style="list-style-type: none"> ● Operator/Parameter The variable can be changed by the HMI. ● Logic/Constant The variable cannot be changed by the HMI. ● OUT variables Always connected to output of Function Block instance always View only. <p>IN/OUT variables</p> <ul style="list-style-type: none"> ● Operator/Parameter: The variable is connected to the input and the output of the Function Block instance. ● Constant: The input of the Function Block instance is connected with the initial value of the variable, the output is connected with variable. ● Logic: The input of the Function Block instance is not connected, the output is connected with variable.
Variable used	Specifies if a variables is generated and connected to the Function Block instance.	Applies only to IO_PLC variables. Other variables and free variables are always used.
Invert	Creates invert functionality of pin or creates literal if an additional pin is used to invert this IO point.	<p>Applies only to IO_PLC variables of data type BOOL.</p> <p>Depending on the Control Module type the input/output of Function Block instance is either inverted or if the Control Module Type has a special input to specify the inverted logic, a literal 0 or 1 (inverted) is connected to this input.</p>
State RAM address	Variable addresses	<ul style="list-style-type: none"> ● IO_PLC variables: The addresses are calculated from addresses of the I/O modules to which the variables are assigned. ● PLC variables: Generated as unlocated variables. ● PLC_NET variables: The address are calculated from the information entered in PLC → Addresses for HMI in Unity Application Generator.

Generated PLC Configuration

What is Generated?

Unity Application Generator will generate all configuration information which is in the scope of Unity Application Generator and what is necessary to describe the high level process design. It will **not** generate e. g.:

- ASCII setup information
 - Specific module parameters
-

PLC Configuration

In the Unity Pro configurator, Unity Application Generator will take care for the following aspects:

- PLC type
 - Memory partitions
 - Peer Cop and I/O Scanner
-

Generated Variables

What is Generated?

Most variables in Unity Application Generator will generate variables in Unity Pro. Only variables of the type HMI will not generate a variable in Unity Pro but a tag in the HMI data base.

Variables

Each variable in Unity Application Generator of the connection types listed in the following table will result in a variable in Unity Pro.

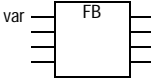
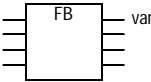
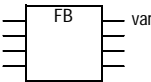
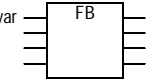
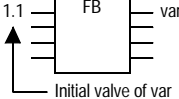
Note: IO_PLC variables which are declared `not used` will not be generated in Unity Pro.

Connection types and related variables in Unity Pro:

Connection Type	Variable type in Unity Pro	Comment
IO_PLC	Located variable	The variables will be mapped to topological addresses of type EBOOL. In the Unity Pro logic the variables will be connected to the inputs/outputs of the FB. Free variables will not be connected.
PLC	Unlocated variable	In the Unity Pro logic the variables will be connected to the inputs/outputs of the FB of the Control Module. Free variables will not be connected.
PLC_HMI	Located variable	The variables will be mapped to %M and %MW variables in the State RAM. In the Unity Pro logic the variables will be connected to the inputs/outputs of the FB of the Control Module, for details see table below.
PLC_NET	Located variable	In Unity Pro located variables will be generated.

Note: Free variables will be generated in Unity Pro without any logic.

Relation of In/Out definition, command type and generated connection of the variable to the FB for PLC_HMI Control Module type variables (the variable is named "var"):

In/Out	Command	Generated
In	Operator Parameter	
Out	View only	
In/Out	Logic	
	Operator Parameter	
	Constant	

Generated Code: Equipment Modules, Control Modules and Interlocks

What is Generated?

For each Equipment Module Unity Application Generator generates a section in Unity Pro. This section will contain a Function Block for each Control Module of the Equipment Module. Each Control Module has its own specific Function Block. For details on the Function Blocks refer to the Control Module Type (SCoD) documentation.

All variables required by the Unity Pro project will also be created.

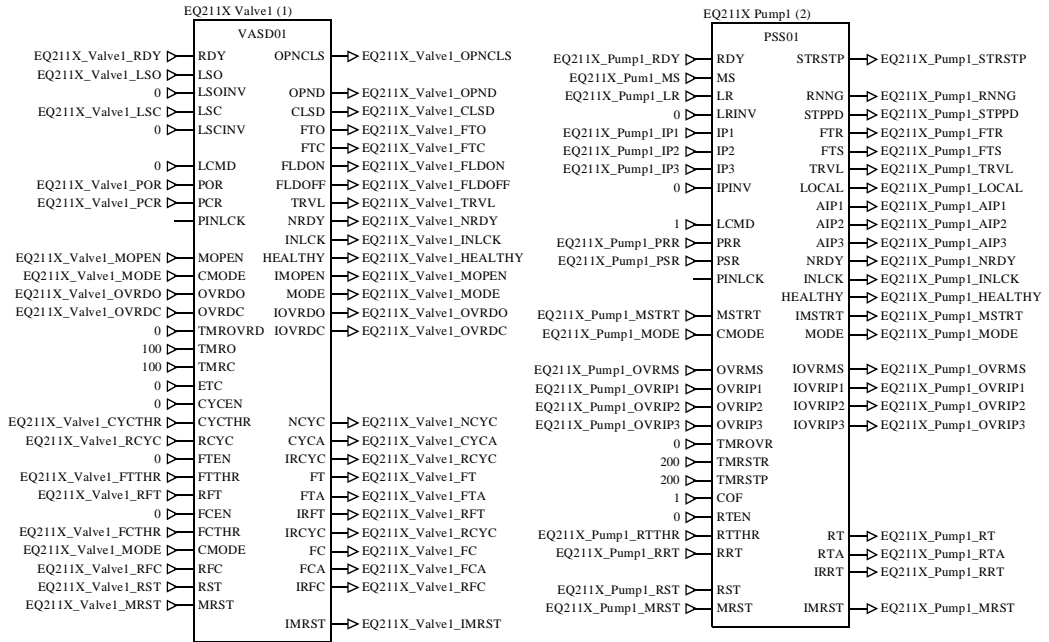
For an interlock definition the following is generated:

For ...	Unity Application Generator generates ...
A syntactically and semantically correct interlock definition	A Function Block network realizing the interlock definition connected to the corresponding SCoD
An incorrect interlock definition	A comment for the corresponding SCoD

Note: Please note the following information on interlocks:

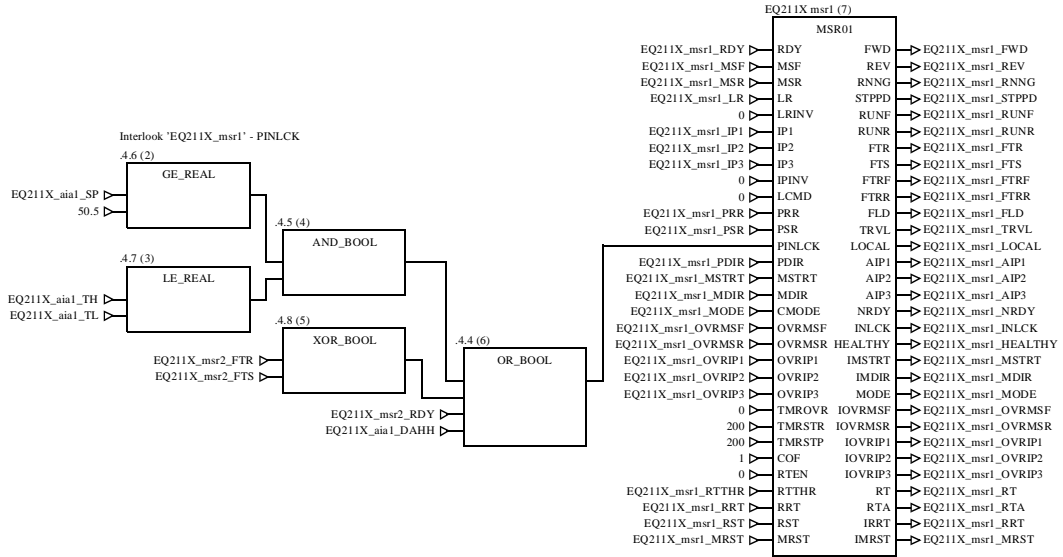
- Syntactically correct interlock definition means: It is a valid logical expression.
- Semantically correct interlock definition means: All used variables are existing on the PLC where the SCoD is generated.
- The generated Unity Pro Function Block networks in some cases have to be rearranged manually.
- Existing interlock definitions from older versions of Unity Application Generator (<V1.5) will stay as comments.
- For interlock definitions as comments the control engineer has to add the PLC code manually.

Example Section Section generated for an Equipment Module containing 2 Control Modules:



**Example:
Interlock**

Function Block generated for a Control Module connected to a Function Block network generated for an interlock definition:



Note: See the corresponding interlock definition in Unity Application Generator in *Example: Interlock Definition, p. 150*

Generated Code: Communication

What is Generated? What is generated depends of the type of communication. PLC <-> PLC communication is performed as Peer Cop or I/O Scanner communication.

Code for Peer Cop Communication Peer Cop communication is restricted to 32 words. Unity Application Generator uses 1 word for a watchdog counter, thus 31 words remain for the communication. Unity Application Generator builds up logic to

- assemble in a data table all PLC variables to be communicated into one structured variable and to
- disassemble for the receiving PLC the structured variable.

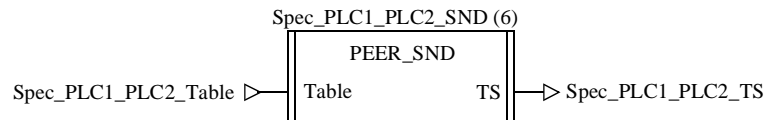
Unity Application Generator will configure the Peer Cop tables with memory and network addresses.

Code for I/O Scanner Communication I/O Scanner communication is very similar to Peer Cop communication. The differences are the following:

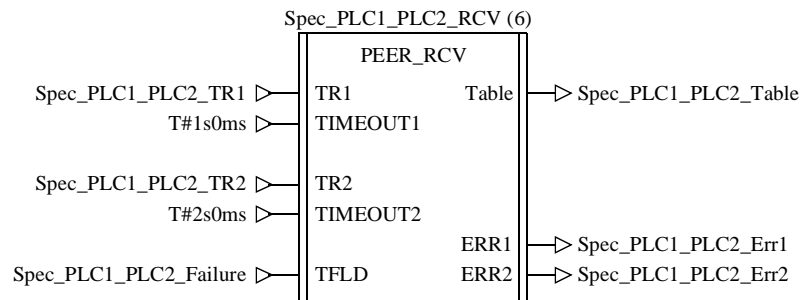
- I/O Scanner communication is restricted to 100 words.
- I/O Scanner tables need only an entry in the sending PLC:

Example of Generated Function Blocks The following figures show examples of the Function Blocks generated for Peer Cop communication.

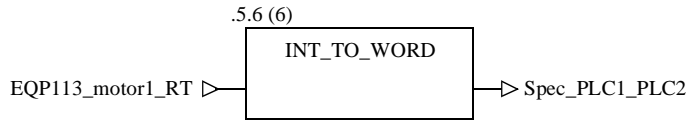
Function block to send Peer Cop table to another PLC



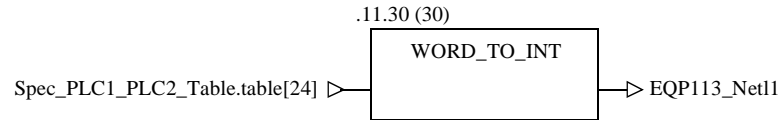
Function Block to receive Peer Cop table from another PLC



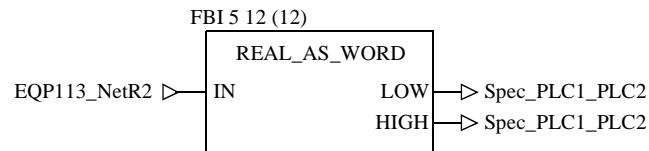
Convert datatype INT to WORD



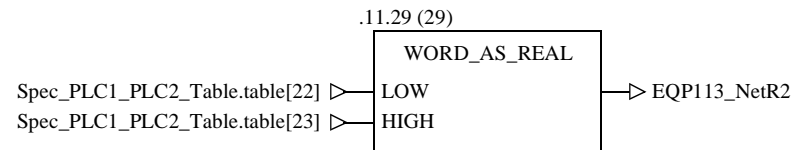
Convert datatype WORD to INT



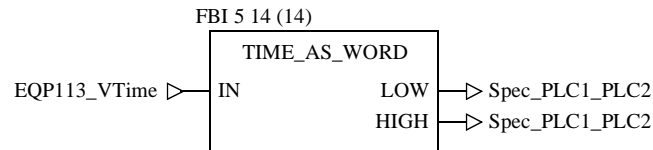
Convert REAL as WORD (2 words)



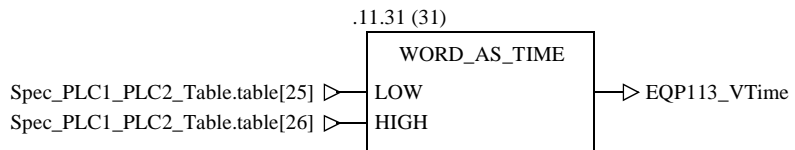
Convert WORD as REAL



Convert TIME as WORD (2 words)



Convert WORD as TIME (2 words)



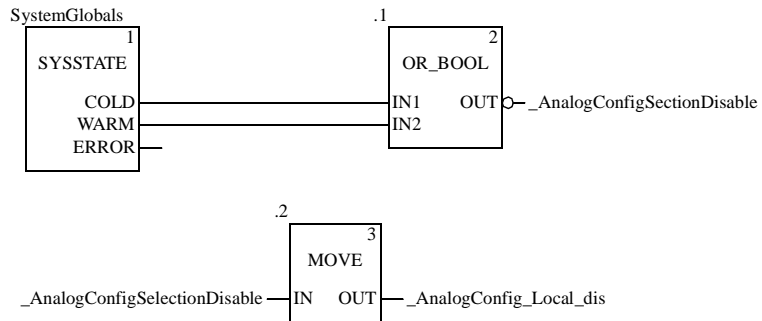
Generated Code: Initialization (Quantum only)

What is Generated?

The initialization section ensures that the initialization is only executed if the PLC has to perform cold start-up and warm start-up.

Example Code

Initialization control code



Generated Code: Scaling of Analog Values (Quantum only)

What is Generated?

With Quantum PLCs for the scaling of analog values the following is generated:

- A group called `AnalogConfiguration`
- In this group further groups for the different I/O types (local, remote, distributed)
- Within these groups a section for each Rack

Examples: Section `AnalogConfig_Remote_Drop6` or `AnalogConfig_DIO1_Drop16`

Within these sections a DROP Function Block, (a XBE Function Block if a backplane expander is used) and the Function Blocks for the analog modules are generated and connected appropriately. The variables mapped to analog modules are connected to the respective EFBs (only for data types ANL_IN and ANL_OUT).

Note: All generated names are independent of the installed language version of Unity Application Generator. Thus the Unity Pro projects can be used with all Unity Application Generator language versions.

Generated Code: Discrete Configuration (Premium only)

What is Generated?

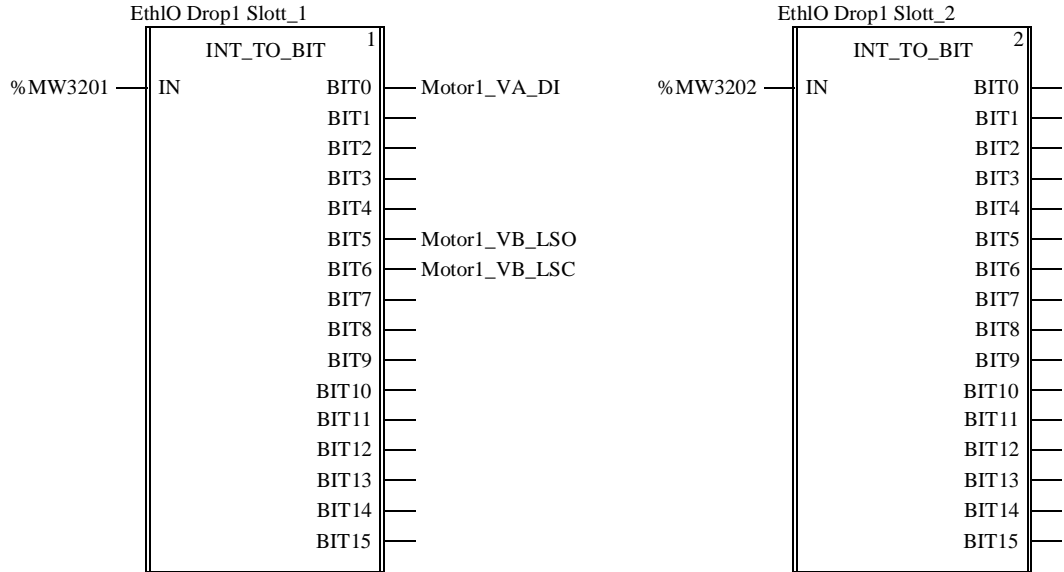
With Premium PLCs for discrete configuration the following is generated:

- A group called `DiscreteConfiguration`.
- Within this group a `Program` group.
 - Within this group a section for IO Scanner management if Ethernet I/O is used. The section contains 1 or more `INT_TO_BIT` and/or `BIT_TO_INT` function blocks for each Rack and Modul. It is needed to make the single I/O points available in Unity Application Generator.
 - Within this group a section for Peer Cop management if Modbus Plus I/O is used. The section contains 1 or more `INT_TO_BIT` and/or `BIT_TO_INT` function blocks for each Rack and Modul. It is needed to make the single I/O points available in Unity Application Generator.

Note: All generated names are independent of the installed language version of Unity Application Generator. Thus the Unity Pro projects can be used with all Unity Application Generator language versions.

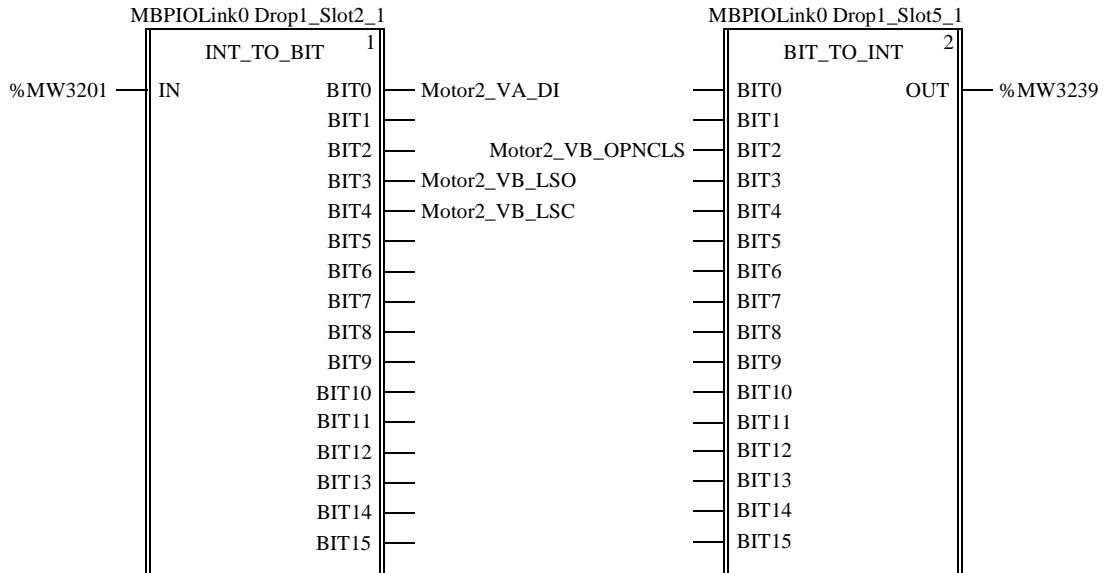
Example for IO Scanner Example for IO Scanner

DO NOT MODIFY THIS SECTION



Example for Peer Cop

DO NOT MODIFY THIS SECTION



B.4 Generation for Monitor Pro

Overview

Introduction

This section describes the generation of Unity Application Generator for Monitor Pro.

What's in this Section?

This section contains the following topics:

Topic	Page
Introduction	333
Generated Variables and their Graphical Representation in the HMI	333
Generated Screens	335
Generated Monitor Pro Database Objects	336
Generated Monitor Pro Pictures	340

Introduction

Introduction

If you are using Monitor Pro, the communication with the HMI is performed through standard Ethernet drivers. Unity Application Generator will create all required variables in the Data Servers so that they can be used within the PLC and HMI. To display the information at the HMI, Unity Application Generator generates the following:

- One screen for each Equipment Module
- One animated symbol for each Control Module
- All required variables for the Control Module
- A graphical symbol for each free variable (optional for PLC_HMI variables)

Control Module variables are part of the animated symbols of the Control Module. Free variables will be shown as a graphical symbol which shows the value of the variable and allows the user to change that value depending on the access rights of the operator and command settings. They will be displayed on the Equipment Module screen they belong to.

Generated Variables and their Graphical Representation in the HMI

What is Generated?

Unity Application Generator will generate tags for all variables of the connection type HMI and PLC_HMI.

Variables

Each variable in Unity Application Generator of the following connection types will create a variable in the HMI.

Connection types and related variables in the HMI:

Connection Type	Variable in the HMI	Comment
PLC_HMI	Variable with communication parameters	If the variable is a Control Module Type variable, it will be connected to the appropriate Control Module to animate the graphic of the Control Module. If it is a free variable, the control engineer has to plan for appropriate usage. Nevertheless a graphical symbol will be created.
HMI	Variable without communication parameters	These variables will be generated in the HMI only.

Example of Variables Database in Monitor Pro

Variables database in Monitor Pro:

Name	Description	Type
TANK11_Motor1_OVRIP1		DIGITAL
TANK11_Motor1_OVRIP2		DIGITAL
TANK11_Motor1_OVRIP3		DIGITAL
TANK11_Motor1_OVRMSF		DIGITAL
TANK11_Motor1_OVRMSR		DIGITAL
TANK11_Motor1_RNNG		DIGITAL
TANK11_Motor1_RRT		DIGITAL
TANK11_Motor1_RT		ANALOG
TANK11_Motor1_RTA		DIGITAL
TANK11_Motor1_RTTHR		ANALOG
TANK11_Motor1_RUNF		DIGITAL
TANK11_Motor1_RUNR		DIGITAL
TANK11_Motor1_STPPD		DIGITAL
TANK11_Motor1_TRVL		DIGITAL
TANK11_Motor1_marion		FLOAT
TANK11_VAS1_CLSD		DIGITAL

Graphic Representation of Free Variables

For each free variable of communication type PLC_HMI an animated graphical symbol will be generated if the `Graphical Symbol` is checked and the `Graphical Symbol Name` is entered within the variable property window.

Generated Screens

What is Generated?

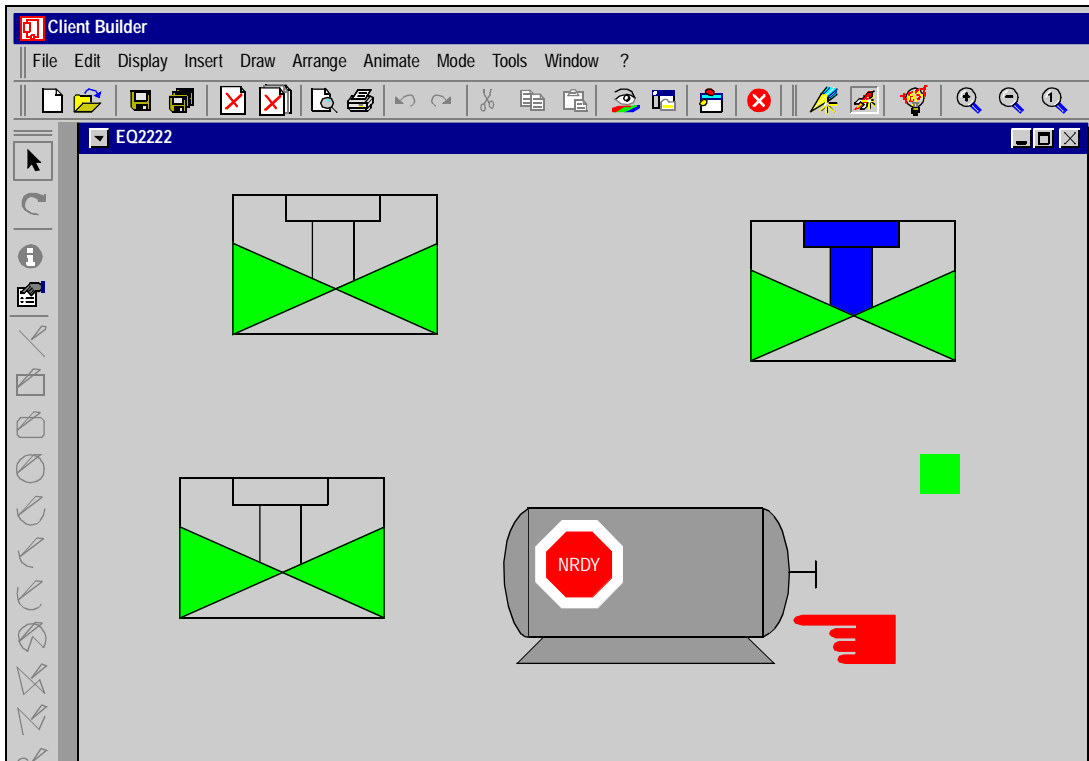
Screens are created for:

- each Equipment Module.

Control Modules are displayed on the Equipment Module screen they belong to.

Example

Configuration screen of Monitor Pro.



Generated Monitor Pro Database Objects

Introduction

A Monitor Pro Server node corresponds to a Unity Application Generator Data Server. Several Data Servers can be defined. Each Data Server must be deployed on a different Monitor Pro Server node.

There will be defined one Data Server for each Monitor Pro Server system in the Unity Application Generator project. Unity Application Generator will create all tags corresponding to the variables of type `HMI` or `PLC_HMI` defined in the project.

Modifications of Tag Names

If Unity Application Generator detects a modification of a tag during the generation, all its properties will be set with the actual values from the Unity Application Generator database. The Monitor Pro database tags are updated through drivers with the actual values of PLC variables.

Generation for a Data Server

The following table gives an overview of the Monitor Pro database objects generated by the Unity Application Generator from objects and their attributes.
Generation for a Data Server

Object	Monitor Pro Database	Comment
Data Server	Monitor Pro tag database	The user has the possibility to set the database path (View → Options → Monitor Pro).

Generation for a Channel

The path in the channel will be generated as IP network address for the TCP/IP driver
Generation for a channel

Object	Monitor Pro Database	Comment
Channel	Represents the link between Data Server and PLC	Nothing can be generated if a Channel does not exist. Deleting a Channel deletes all generated variables of the corresponding PLC in the Monitor Pro database.

Generation for a PLC

Generation for a PLC

Object	Monitor Pro Database	Comment
PLC	No corresponding object in Data Server	

Generation for an Area, Process Cell, Unit

Generation for an Area, Process Cell, Unit

Object	Monitor Pro Database	Comment
Area	Nothing done in Data Server	
Process Cell	Nothing done in Data Server	
Unit	Nothing done in Data Server	

Generation for an Equipment Module

Generation for an Equipment Module

Object	Monitor Pro Database	Comment
Equipment Module	No corresponding object in the Data Server	

Generation for the attributes of an Equipment Module

Attribute	Monitor Pro Database	Comment
Name	The name of the Equipment Module is part of Equipment Module variable names and Control Module variable names	Variables will be renamed in this Equipment Module.
PLC	A PLC is connected to one Data Server, thus the PLC defines the Data Server in which the tags will be generated.	The PLC to which the Equipment Module is assigned cannot be changed if Equipment Module has been generated once.
Description	Tag description	Changing the description of an Equipment Module results in modifying the alarm text of all variables of the Equipment Module and all variables of the Control Modules of the Equipment Module of type digital, if default alarm text is used.

Generation for a Control Module

Generation for a Control Module

Object	Monitor Pro Database	Comment
Control Module	No corresponding object in Data Server, but all PLC_HMI or HMI variables of the Control Module will be generated as tags.	

Generation for the attributes of a Control Module

Attribute	Monitor Pro Database	Comment
Parent Equipment Module	The name of the Equipment Module is part of the Control Module variable names.	
PLC	A PLC is connected to one Data Server, thus the PLC defines the Data Server in which the tags will be generated.	The PLC to which the Control Module is assigned cannot be changed if Control Module has been generated once.
Name	The name of the Control Module is part of the Control Module variable names.	Variables will be renamed in this Control Module.
Description	Used for alarm text (tag description)	Changing the description of a Control Module results in modifying the alarm text of all variables of the Control Module of type digital, if default alarm text is used.

Generation for a Variable

Generation for a variable

Object	Monitor Pro Database	Comment
Variable	Tag in Monitor Pro database	The variable name is generated as EquipmentModuleName_ControlModuleName_VariableName (Control Module variable) or EquipmentModuleName_VariableName (Equipment Module variable).

Generation for attributes of a variable

Attribute	Monitor Pro Database	Comment
Name	The name of the variable is part of tag name.	The name can only be changed for free variables.
Description	Tag description	Max. 80 characters
Communication Frame	I/O Translator Dataset Definition	Name of the communication frame the variable is assigned to. Corresponds to the name of one of the <code>hmi_communication</code> elements in the channel element. The value can be empty.
Alarm	Alarm Options /Enable Alarming	
Alarm Priority 1..8	Alarm Priority	1 to 8 alarm priorities can be possible for non-boolean variables. For boolean variables just 1 alarm can be possible. Possible value can be set up in customization (1..9999). Example values are: <ul style="list-style-type: none"> ● 1 - low ● 2 - medium ● 3 - high ● 4 - urgent
Alarm Limit 1..8	Alarm Limit: Depending on data type	1 to 8 alarm limits can be possible for non-boolean variables. For boolean variables just 1 alarm can be possible.
Alarm Text 1..8	Alarm Text	Alarm text for the specific alarm limit. 1 to 8 alarm texts can be possible for non-boolean variables. For boolean variables just 1 alarm can be possible.
Alarm Group	Alarm	For grouping the alarm variables. All alarms of a variable can be in the same alarm group.
Archive	Archive	The variable can be in three different states for archiving: <code>Not_Assigned</code> , <code>No</code> or <code>Historic</code> .
Archive Name	Archive	If the variable is marked for archiving, the user has to define the archive name. The user can choose between <code>Not_Assigned</code> or the name of the archive.

Generated Monitor Pro Pictures

Introduction

The following table gives an overview of the Monitor Pro pictures generated by the Unity Application Generator from objects and their attributes.

Generation for an HMI

Generation for an HMI

Object	Monitor Pro Pictures	Comment
HMI	Set of Monitor Pro pictures	HMI pictures are located in the path which the user has entered (View → Options → Monitor Pro). Under this path the user will find a <code>mimic</code> files directory where the pictures will be stored.

Generation for attributes of an HMI

Attribute	Monitor Pro Pictures	Comment
Name	Name of generation file	When the Monitor Pro HMI project is generated a file is created: The name of the file is <code>HMINAME.UAG</code> . The user will find the file within the <code>mimics</code> directory. Changing the path will result in renaming generation file.

Generation for a Control Domain

Generation for a Control Domain

Object	Monitor Pro Pictures	Comment
Control Domain	Represents the link between HMI and Equipment Module and Control Modules	Nothing can be generated for HMI if the Control Domain does not exist. Deleting of a Control Domain, deletes all generated Control Module symbols in Monitor Pro pictures.

Generation for a PLC

Generation for a PLC

Object	Monitor Pro Pictures	Comment
PLC	No corresponding object in HMI	

Generation for Area, Process Cell and Unit

Generation for Areas, Process Cells and Units

Object	Monitor Pro Pictures	Comment
Area	Nothing done in HMI	
Process Cell	Nothing done in HMI	
Unit	Nothing done in HMI	

Generation for an Equipment Module

Generation for an Equipment Module

Object	Monitor Pro Pictures	Comment
Equipment Module	An Monitor Pro picture for each Equipment Module	

Generation for attributes of an Equipment Module

Attribute	Monitor Pro Pictures	Comment
Name	Name of the associated Monitor Pro picture is part of Equipment Module variable Monitor Pro symbol names, Control Module Monitor Pro symbol names and Control Module variable Monitor Pro symbol names.	Changing the name of an Equipment Module results in renaming the Monitor Pro picture and all contained Monitor Pro symbols.
PLC		The PLC to which the Equipment Module is assigned cannot be changed if the Equipment Module has been generated once.
Control Domain	Define HMI in which Equipment Module picture is generated.	If the HMI is different, the Monitor Pro symbol's controls will be removed from the Equipment Module picture in the current HMI and a new picture will be created in the new HMI.

Generation for a Control Module

Generation for a Control Module

Object	Monitor Pro Pictures	Comment
Control Module	Monitor Pro symbol in the Equipment Module Monitor Pro picture	

Generation for attributes of a Control Module

Attribute	Monitor Pro Pictures	Comment
Parent Equipment Module	Defines Monitor Pro pictures	A Control Module can be moved to another Equipment Module by drag and drop in the Physical Model of Unity Application Generator. Changing Control Module Parent Equipment Module results in moving all free variable Monitor Pro symbol controls and Control Module Monitor Pro symbol's from old picture to new one.
PLC		No picture generation if no PLC is assigned to the equipment
Name	The name of the Control Module is part of Control Module Monitor Pro symbol name and free Control Module variable Monitor Pro symbol names.	Changing the name of a Control Module results in renaming all free variables Monitor Pro symbol of the Control Module, and Control Module Monitor Pro symbol.

Generation for a Variable

Generation for a variable

Object	Monitor Pro Pictures	Comment
Variable	Property of Control Module or Free variable Monitor Pro symbol.	

Generation for attributes of a variable

Attribute	Monitor Pro Pictures	Comment
Graphical symbol	Graphical symbol	An animated graphical symbol will be generated if Graphical symbol is checked in the variable property window (in the General tab). Changing the check box for Graphical symbol results in creating or deleting a free variable Monitor Pro symbol on corresponding Monitor Pro picture.
Graphical symbol name	Graphical symbol name	Name of the symbol for the free variable.
Name	Variable name	The name can only be changed for free variables.

B.5 Generation for iFIX

Overview

Introduction

This section describes the generation of Unity Application Generator for iFIX.

What's in this Section?

This section contains the following topics:

Topic	Page
Characterization	345
Generated Variables and their Graphical Representation in the HMI	346
Generated Screens	347
Generated iFIX Database Objects	348
Generated iFIX Pictures	354
Generated iFIX Driver Configuration from Unity Application Generator Point of View	359
Generated iFIX Driver Configuration from the Driver Point of View	361

Characterization

Characterization If you are using iFIX the communication with the HMI is performed through iFIX drivers. Unity Application Generator will create all required variables in the Data Servers so that they can be communicated between PLC and HMI.

To display the information at the HMI, Unity Application Generator generates the following:

- One screen for each Equipment Module
- One animated symbol for each Control Module
- All required variables for the Control Module
- A graphical symbol for each free variable (optional for PLC_HMI variables)

Control Module variables are part of the animated symbols of the Control Module. Free variables will be shown as a graphical symbol which shows the value of the variable and allows the user to change that value depending on the access rights of the operator and command settings. They will be displayed on the Equipment Module screen they belong to.

Generated Variables and their Graphical Representation in the HMI

What is Generated?

Unity Application Generator will generate tags for all variables of the connection type HMI and PLC_HMI.

Variables

Each variable in Unity Application Generator of the following connection types will create a variable in the HMI.

Connection types and related variables in the HMI:

Connection Type	Variable in the HMI	Comment
PLC_HMI	Variable with communication parameters	If the variable is a Control Module Type variable, it will be connected to the appropriate Control Module to animate the graphic of the Control Module. If it is a free variable, the control engineer has to plan for appropriate usage. Nevertheless a graphical symbol will be created.
HMI	Variable without communication parameters	These variables will be generated in the HMI only.

Example of Variables Database in iFIX

Variables database in iFIX:

	Tag Name	Type	Description	Scan Time	I/O Dev	I/O Addr
1	EQ001A_AIA1_DB	AI	Deadband	1	MBT	CH10:401746
2	EQ001A_AIA1_ERR	AI	Error	1	MBT	CH10:401750
3	EQ001A_AIA1_PVR	AI	Process var. real value	1	MBT	CH10:401770
4	EQ001A_AIA1_SP	AI	Setpoint	1	MBT	CH10:401748
5	EQ001A_AIA1_TH	AI	Thresh. High alarm	1	MBT	CH10:401732
6	EQ001A_AIA1_THD	AI	Thresh. H & L deviation alarm	1	MBT	CH10:401738
7	EQ001A_AIA1_THH	AI	Thresh. HH alarm	1	MBT	CH10:401740
8	EQ001A_AIA1_THH	AI	Threshold HH & LL deviation	1	MBT	CH10:401736
9	EQ001A_AIA1_TL	AI	Threshold low alarm	1	MBT	CH10:401734
10	EQ001A_AIA1_TLL	AI	Thresh. LL alarm	1	MBT	CH10:401742
11	EQ001A_AIA1_TRC	AI	Thresh. Rate of change alarm	1	MBT	CH10:401744
12	EQ001A_AIA1_Y	AI	Analog Output Y	1	MBT	CH10:401754
13	EQ001A_AIA1_YMA	AI	Manual Value	1	MBT	CH10:401730
14	EQ001A_AIA1_DF	AI	Register of active faults #1	1	MBT	CH10:401752
15	EQ001A_AIA1_DF	AI	Register of active faults #2	1	MBT	CH10:401753
16	EQ001A_AIA1_FR	AI	Drive Frequency Reference	1	MBT	CH10:401744

Graphic Representation of Free Variables

For each free variable of communication type PLC_HMI an animated graphical symbol will be generated if **Graphical symbol** is checked in the variable property window (in the **General** tab).

Generated Screens

What is Generated?

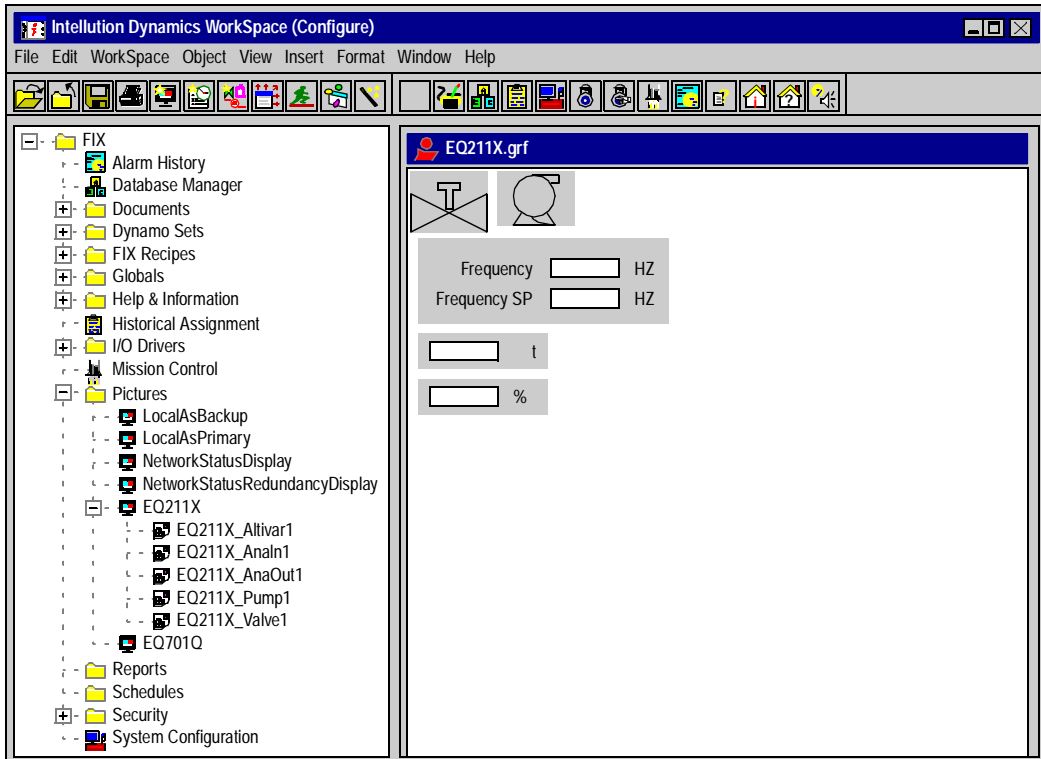
Screens are created for:

- each Equipment Module.

Control Modules are displayed on the Equipment Module screen they belong to.

Example

Configuration screen of Intellution's iFIX



Generated iFIX Database Objects

Introduction

A SCADA node corresponds to a Unity Application Generator Data Server. Several Data Servers can be defined. Each Data Server must be deployed on a different iFIX SCADA node.

The generation for SCADAs consists of creating an iFIX database (PDB file in the ... \DYNAMICS\PDB directory) for each Data Server defined in the Unity Application Generator project. Unity Application Generator will create all tags corresponding to the variables of type HMI or PLC_HMI defined in the project.

Modifications of Tag Names

iFIX does not allow to change tag names. Thus, changing a tag name will be handled by Unity Application Generator as deleting the old tag and creating a new one with a new name.

If Unity Application Generator detects a modification of a tag during the generation, all its properties will be set with the actual values from the Unity Application Generator database. The iFIX database tags are updated through drivers with the actual values of PLC variables.

Generation for a Data Server

The following table gives an overview of the iFIX database objects generated by the Unity Application Generator from objects and their attributes.

Generation for a Data Server

Object	iFIX Database	Comment
Data Server	iFIX tag database	The database is located in <iFIX path>\PDB\DataServerName.pdb

Generation for attributes of a Data Server

Attribute	iFIX Database	Comment
Name	Name of iFIX database Name of generation file	Changing the name of a Data Server will results in renaming the PDB file and the generation file.

Generation for a Channel

Generation for a channel

Object	iFIX Database	Comment
Channel	Represents the link between Data Server and PLC	Nothing can be generated if a Channel does not exist. Deleting a Channel deletes all generated variables of the corresponding PLC in the iFIX database.

Generation for a PLC

Generation for a PLC

Object	iFIX Database	Comment
PLC	No corresponding object in Data Server	

Generation for a Control Domain

Generation for a Control Domain

Object	iFIX Database	Comment
Control Domain	No corresponding object in the Data Server	

Generation for the attributes of a Control Domain

Attribute	iFIX Database	Comment
Name	Name of iFIX Alarm Area. Part of Security Area name of iFIX tag	Changing the Control Domain name results in changing the Alarm Area and Security Area for all tags of that Control Domain. The names of the Alarm Area and Security Areas have to be changed manually in iFIX SCU.

Generation for an Area, Process Cell, Unit

Generation for an Area, Process Cell, Unit

Object	iFIX Database	Comment
Area	Nothing done in Data Server	
Process Cell	Nothing done in Data Server	
Unit	Nothing done in Data Server	

Generation for an Equipment Module

Generation for an Equipment Module

Object	iFIX Database	Comment
Equipment Module		No corresponding object in the Data Server

Generation for the attributes of an Equipment Module

Attribute	iFIX Database	Comment
Name	The name of the Equipment Module is part of Equipment Module variable names and Control Module variable names	Since iFIX tag names cannot be changed, changing the name of an Equipment Module results in deleting all variables of the Equipment Module and all variables of the Control Modules of the Equipment Module, and re-creating them with new names.
PLC	A PLC is connected to one Data Server, thus the PLC defines the Data Server in which the tags will be generated.	The PLC to which the Equipment Module is assigned cannot be changed if Equipment Module has been generated once.
Control Domain	Used for Alarm Areas and Security Areas of Variables	Changing the Control Domain of an Equipment Module results in modifying Alarm Area and Security Area of all variables of the Equipment Module and all variables of the Control Modules of the Equipment Module. Note: If you change the name of a Control Domain you have to change the name of the Alarm Area (=ControlDomainName) and the names of all Security Areas (=ControlDomainName_AccessLevel) in iFIX SCU.
Description	Used for alarm text (tag description)	Changing the description of an Equipment Module results in modifying the alarm text of all variables of the Equipment Module and all variables of the Control Modules of the Equipment Module of type digital, if default alarm text is used.

Generation for a Control Module

Generation for a Control Module

Object	iFIX Database	Comment
Control Module	No corresponding object in Data Server, but all PLC_HMI or HMI variables of the Control Module will be generated as tags.	

Generation for the attributes of a Control Module

Attribute	iFIX Database	Comment
Parent Equipment Module	The name of the Equipment Module is part of the Control Module variable names.	A Control Module can be moved to another Equipment Module by drag and drop in the physical Model of Unity Application Generator. Since iFIX tag names cannot be changed, changing Control Module Parent Equipment Module results in deleting all variables of the Control Module in the Data Server where they have been generated.
PLC	A PLC is connected to one Data Server, thus the PLC defines the Data Server in which the tags will be generated.	The PLC to which the Control Module is assigned cannot be changed if Control Module has been generated once.
Name	The name of the Control Module is part of the Control Module variable names.	Since iFIX tag names cannot be changed, changing the name of a Control Module results in deleting all variables of the Control Module, and recreating them with new names.
Description	Used for alarm text (tag description)	Changing the description of a Control Module results in modifying the alarm text of all variables of the Control Module of type digital, if default alarm text is used.

Generation for an Interlock

Interlocks have no counter part in the HMI application's logic. But the ActiveX control which represents the Control Module in the HMI displays the current interlock state of the Control Module. This is outside of the scope the Unity Application Generator implementation, but is implemented for each individual SCoD.

Generation for a Variable

Generation for a variable

Object	iFIX Database	Comment
Variable	Tag in iFIX database	The variable name is generated as EquipmentModuleName_ControlModuleName_VariableName (Control Module variable) or EquipmentModuleName_VariableName (Equipment Module variable).

Generation for attributes of a variable

Attribute	iFIX Database	Comment
Name	The name of the variable is part of tag name.	The name can only be changed for free variables.
Description	Tag description	Max. 40 characters
Command	Defines how the tags can be controlled by the HMI	Applies only to PLC_HMI variables. Free variables are always IN/OUT. IN or IN/OUT variables: <ul style="list-style-type: none"> ● Operator/Parameter: The variable can be changed by the HMI: Advanced Options → Enable Output = Yes. ● Logic/Constant: The variable cannot be changed by the HMI: Advanced Options → Enable Output = No.
Alarm	Alarms/Alarm Options/Enable Alarming	Only BOOL variables can be alarms, generated as DI alarm tags. In Advanced → Alarm Extension Fields → Alarm Field1 Unity Application Generator generates the name of the picture. With a small script one can implement a jump to the picture which contains the variable with the alarm.
State0Text	Basic/Labels Open	Only DI tags. Label associated with 0 State.
State1Text	Basic/Labels Close	Only DI tags. Label associated with 1 State.
Alarm Priority	Alarms/Alarm Priority	Only DI alarm tags. Possible values: L as Low, M as Medium, H as High
Alarm State	Alarms/Alarm Type: 1 = Close, 0 = Open	Only DI alarm tags. Value 0 or 1 for which Alarm occurs
Alarm Text	Advanced/Alarm Extension Fields/ Alarm Field 1	Only DI alarm tags. Replace Tag Description for Alarm Tags
Scaling Min	Basic/Engineering Units/Low Limit	Only AI tags.

Attribute	iFIX Database	Comment
Scaling Max	Basic/Engineering Units/High Limit	Only AI tags.
Measurement Unit	Basic/Engineering Units/Units	Only AI tags.
Access Level	Advanced/Security Areas	The access level is used with the Control Domain to define the Security Area of this variable. See above.

Generated iFIX Pictures

Introduction

The following table gives an overview of the iFIX pictures generated by the Unity Application Generator from objects and their attributes.

Generation for an HMI

Generation for an HMI

Object	iFIX Pictures	Comment
HMI	Set of iFIX pictures	HMI pictures are located in IFIX PATH\PIC\PICTURENAME.GRF

Generation for attributes of an HMI

Attribute	iFIX Pictures	Comment
Name	Name of generation file	When iFIX HMI project is generated a file is created: <IFIX PATH>\PIC\HMINAME.GILChanging the path will result in renaming generation File.

Generation for a Control Domain

Generation for a Control Domain

Object	iFIX Pictures	Comment
Control Domain	Represents the link between HMI and Equipment Module and Control Modules	Nothing can be generated for HMI if the Control Domain does not exist.Deleting a Control Domain, delete all generated Control Module ActiveX in iFIX pictures.

Generation for attributes of a Control Domain

Attribute	iFIX Pictures	Comment
Name	Property of each Control Module or Free Variable ActiveX	Changing the Control Domain name results in changing property for all ActiveX of that Control Domain.If the new Control Domain belongs to another HMI, all generated Control Module ActiveX in iFIX pictures are deleted from the old HMI and created in the new HMI.

Generation for a PLC

Generation for a PLC

Object	iFIX Pictures	Comment
PLC		No corresponding object in HMI

Generation for attributes of a PLC

Attribute	iFIX Pictures	Comment
Name		Used in alias for each ActiveX

Generation for Area, Process Cell and Unit

Generation for Areas, Process Cells and Units

Object	iFIX Pictures	Comment
Area		Nothing done in HMI
Process Cell		Nothing done in HMI
Unit		Nothing done in HMI

Generation for an Equipment Module

Generation for an Equipment Module

Object	iFIX Pictures	Comment
Equipment Module	An iFIX picture for each Equipment Module	Note: iFIX picture names must not start with a digit!

Generation for attributes of an Equipment Module

Attribute	iFIX Pictures	Comment
Name	Name of the associated iFIX picture is part of Equipment Module variable ActiveX names, Control Module ActiveX names and Control Module variable ActiveX names.	Changing the name of an Equipment Module results in renaming the iFIX picture and all contained ActiveX's.
PLC		The PLC to which the Equipment Module is assigned cannot be changed if the Equipment Module has been generated once.
Control Domain	Define HMI in which Equipment Module picture is generated.	Changing the Control Domain of an Equipment Module results in modifying the Control Domain property for all ActiveX's of the iFIX Equipment Module picture. If the HMI is different, the ActiveX controls will be removed from the Equipment Module picture in the current HMI and a new picture will be created in the new HMI.
Description	Description property of iFIX picture	Changing the description of an Equipment Module results in modifying description property of iFIX Equipment Module picture.

Generation for a Control Module

Generation for a Control Module

Object	iFIX Pictures	Comment
Control Module	ActiveX in the Equipment Module iFIX picture	

Generation for attributes of a Control Module

Attribute	iFIX Pictures	Comment
Parent Equipment Module	Defines iFIX pictures	A Control Module can be moved to another Equipment Module by drag and drop in the Physical Model of Unity Application Generator. Changing Control Module Parent Equipment Module results in moving all free variable ActiveX controls and Control Module ActiveX's from old picture to new one.
PLC		The PLC to which the Control Module is assigned cannot be changed if Control Module has been generated once.
Name	The name of the Control Module is part of Control Module ActiveX name and free Control Module variable ActiveX names.	Changing the name of a Control Module results in renaming all free variables ActiveX of the Control Module, and Control Module ActiveX.
Description	Description of the Control Module	Changing the description of a Control Module results in changing that property of Control Module ActiveX.

Generation for an Interlock

Interlocks have no counter part in the HMI applications logic. But the ActiveX control which represents the Control Module in the HMI displays the current interlock state of the Control Module. This is outside of the scope the Unity Application Generator implementation, but is implemented for each individual SCoD.

Generation for a Variable

Generation for a variable

Object	iFIX Pictures	Comment
Variable	Property of Control Module or Free variable ActiveX.	The property name is: IAT_VariableName_VariableProperty (Control Module variable) or IAT_VariableProperty (Free variable).

Generation for attributes of a variable

Attribute	iFIX Pictures	Comment
Graphical symbol	Graphical symbol	An animated graphical symbol will be generated if Graphical symbol is checked in the variable property window (in the General tab). Changing the check box for Graphical symbol results in creating or deleting a free variable ActiveX on corresponding iFIX picture.
Name	Variable name	The name can only be changed for free variables.
Description	Description in ActiveX	Only for free variables
Command	Defines how the variable is connected to the Function Block instance	Applies only to PLC_HMI variables. Other variables and free variables are always IN/OUT. IN or IN/OUT variables: <ul style="list-style-type: none"> ● Operator/Parameter: The variable can be changed by the ActiveX. ● Logic/Constant: The variable cannot be changed by the ActiveX
Alarm	Alarm True or False	Only BOOL variables can be alarms, generated as DI alarm tags
State0Text	Label associated with 0 State	Only DI tags.
State1Text	For DI variable	Only DI tags
Alarm State	Value 0 or 1 for which Alarm occurs	Only DI alarm tags
Alarm Text	Alarm text	Only DI alarm tags
Boundary Min	Lower boundary to which the operator can set the value	Only AI tags
Boundary Max	Upper boundary to which the operator can set the value	Only AI tags
Display Format	Format how to display the value	Only AI tags
Measurement Unit	Engineering unit	Only AI tags
Access Level	Security area	The access level is used with the Control Domain to define the Security Area of this variable.

Generated iFIX Driver Configuration from Unity Application Generator Point of View

Introduction The following tables give an overview of the iFIX driver configuration generated by Unity Application Generator from the point of view of Unity Application Generator.

Driver Configuration for a Data Server

Generated driver configuration for a Data Server

Object	iFIX Driver	Comment
Data Server	A driver configuration file for each Data Server	

Generated driver configuration for the attributes of a Data Server

Attribute	iFIX Driver	Comment
Name	Used to define configuration file name	Changing Data Server name results in changing the configuration file name.

Driver Configuration for a Channel

Generated driver configuration for a Channel

Object	iFIX Driver	Comment
Channel	A channel and a Control Module for each Channel between PLC and Data Server	Deleting a Channel results in deleting channel, Control Module and corresponding DataBlocks in the driver configuration.

Generated driver configuration for the attributes of a Channel

Attribute	iFIX Driver	Comment
ID	Used to define a driver channel as Channel_<ID> Used to define a driver device as CH<ID>	In iFIX a channel name is limited to maximum 12 characters. Therefore the Channel name specified in Unity Application Generator is not used. A name composed of the word CHANNEL and a serial number is created instead.
HMI Communication	Used to define Datablocks driver configuration	I/O Addresses are updated by Memory Mapper. Datablock Name: <ID>_<BlockType>_<n> <ID> Channel ID <n> Block index: Number of DataBlocks depends on maximum length for each data type

Generated iFIX Driver Configuration from the Driver Point of View

Introduction

The following tables give an overview of the generated values for the iFIX driver objects from the point of view of the driver.

Values for a Channel

Values generated for the attributes of a Channel:

Attribute	Value	Comment
Name	Channel_<ID>	<ID> : Channel ID in Unity Application Generator
Description	Channel_<ID>	<ID> : Channel ID in Unity Application Generator

Values for a Device

Values generated for the attributes of a Channel

Attribute	Value	Comment
Name	Channel_<ID>	<ID> : Channel ID in Unity Application Generator
Description	Channel_<ID>	<ID> : Channel ID in Unity Application Generator
Addressing type	6 Digit	
Bit base	0-15	

Values for a DataBlock

Values generated for the attributes of a DataBlock

Attribute	Value	Comment
Name	Corresponding names will be: <ul style="list-style-type: none"> ● For Synoptic: <ul style="list-style-type: none"> ● <ID>_SBOOL_<n> ● <ID>_SWORD_<n> ● <ID>_SDINT_<n> ● <ID>_SREAL_<n> ● For Event: <ul style="list-style-type: none"> ● <ID>_EBOOL_<n> ● <ID>_EWORD_<n> ● <ID>_EDINT_<n> ● <ID>_EREAL_<n> ● For Parameter: <ul style="list-style-type: none"> ● <ID>_PBOOL_<n> ● <ID>_PWORD_<n> ● <ID>_PDINT_<n> ● <ID>_PREAL_<n> 	<ID> Channel ID in Unity Application Generator <n> Block index: Number of DataBlocks depends on maximum length for each data type
Description	Same as Name	
Starting address	Unity Application Generator value	
Ending address	Unity Application Generator value	Block length = 512 for BOOL and 125 for Other DataType
Data type	Driver types	

B.6 Generation for a Generic HMI

Generation for a Generic HMI

Introduction	<p>If you are using an HMI other than iFIX (generic HMI), Unity Application Generator provides all tag data necessary for a further importing into your HMI.</p>
What is Generated?	<p>For the use of a generic HMI Unity Application Generator generates a XML file and a customizable text file for each DataServer containing all tag information necessary for the process visualisation.</p> <p>Customization of the text file is done with a user provided stylesheet file (XSL file). XML (Extensible Markup Language) and XSL (Extensible Stylesheet Language) are specifications by the W3C (World Wide Web Consortium), see http://www.W3.org for details.</p>
The XML and XSL File	<p>The XML files are generated in the directories specified for each Data Server as application path with the name DATASERVERNAME.XML</p> <p>The XML file generated for the use of a generic HMI contains all data necessary for process visualization like PLC names, network addresses , variable names, addresses, data types, comments as configured with Unity Application Generator. The XML file is a text file that contains the data in a hierachical structure and can be viewed with tools like text editors or Microsoft Internet Explorer.</p> <p>The XSL (rather XSLT file, Extensible Stylesheet Language Transformation) file is specified in Customization editor and defines how the content of the XML file has to be reformatted so that the individual HMI can read the data. This reformatting is done automatically during generation and results in one or more files as specified in Customization editor.</p> <p>Unity Application Generator installs the stylesheet file GenericHMICSV.xls with which the generated XML file is transformed to a CSV file as it was generated in previous versions of Unity Application Generator. Other stylesheets have to be provided by the user.</p> <p>The XSL file is a text file that can be created with simple text editors. Links to tutorials and specialized tools can be found in the internet on www.w3.org.</p>

B.7 Generation for Net Partners

Overview

What is Generated?

This section describes the generation of Unity Application Generator for Net Partners.

What's in this Section?

This section contains the following topics:

Topic	Page
Generation for Net Partners	365
Net Partner Variables: CSV File Format	366

Generation for Net Partners

Introduction Net Partners are devices which cannot be programmed directly by Unity Application Generator. Nevertheless the variables which are related to a Net Partner can be exported in a CSV file to be imported into the Net Partner device.

Note: The generation of the CSV file for the Net Partners is done by the menu path **Net Partner** → **Properties** → **Export**.

What is Generated? Unity Application Generator generates a CSV file which contains all variables which have been put into the communication table of the Channel for the Net Partner by drag and drop.

The CSV File The CSV file contains all of the parameters of the variable necessary for the Net Partner to access the variables in the PLC, for example name, PLC addresses, data type, and so on.
The CSV file can for example be opened by Microsoft Excel and adjusted in a simple way to fit order and format which the programming software of the Net Partner can import.

Net Partner Variables: CSV File Format

Note The numbers in the following tables are only for your overview, they are **not** contained in the CSV file!

CSV File Format For each variable of a Net Partner the CSV file contains a line. The line contains the following elements, separated by a separator (comma or semicolon).

1	<i>plc_name</i>	2	<i>network_type</i>	3	<i>network_address</i>
4	<i>variable_name</i>	5	<i>data_type</i>	6	<i>variable_address</i>
7	<i>initial_value</i>	8	<i>description</i>		

- ***plc_name***: Name of PLC to which variable belongs.
 - ***network_type***: Valid strings are: ModbusPlus and Ethernet.
 - ***network_address***: Network address of PLC, either a ModbusPlus or Ethernet address depending on *network_type*.
 - ***variable_name***: Full variable name, for example
EquipmentModuleName_ControlModuleName_VariableName.
 - ***data_type***: Data type of the variable (for example BOOL, WORD, REAL and so on).
 - ***variable_address***: The state ram address of the variable. The format is Sxxxxxx, that is the segment 0,1,3 or 4 and the offset with five digits and leading zeros, for example 000010 or 400001.
 - ***initial_value***: Initial value of the variable. The format corresponds to the data type of the variable (for example for BOOL it is 0 or 1, for REAL 1.23 and so on).
 - ***description***: The description of the variable. It is composed as: *description of variable - description of Control Module - description of Equipment Module*.
-

Format of the CSV Files for Import and Export



Overview

Introduction

The CSV files are used for importing and exporting data to and from other tools outside Unity Application Generator. This chapter provides the exact format of the CSV files for all possible import and export processes.

Note: Import / Export between different product versions of UAG is not supported.

Restrictions for Unity Pro

Only objects which are created in Unity Application Generator can be imported e.g.:

- Instruments
- Initial values of existing variables
- Hardware modules in Modbus Plus, Ethernet or Generic racks.

Objects which are NOT created in Unity Application Generator CANNOT be imported e.g.:

- Unity Pro PLCs
- Unity Pro racks

The export of objects is not restricted. All objects can be exported.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
General Format	368
Physical Model Elements: CSV File Format	370
Topological Model Elements: CSV File Format	376
Instruments: CSV File Format	385
Initial Values: CSV File Format	386

General Format

General Structure of CSV File

The following figure shows the general structure of a CSV file used by Unity Application Generator for import or export:

	1	2	3	4	5
Line 1	KEYWORD	; value	; value	; value	; value
Line 2	KEYWORD	; value	; value	; value	; value
Line 3	KEYWORD	; value	; value	; value	; value
Line 4	KEYWORD	; value	; value	; value	; value
Line 5	KEYWORD	; value	; value	; value	; value
Line 6	KEYWORD	; value	; value	; value	; value

Separator

The separator character can be a tab, or a comma or semicolon or any other character. The separator character is set in the **View** → **Options** dialog in Unity Application Generator.

Note: Do not use a separator character that may be used in the fields.

Analyzer Options for Import

The options for analyzing the project are also used for import to show warnings or stop after a certain number of errors or warnings:

- Show warnings
 - Stop after n errors
 - Stop after n warnings
-

Keyword

The keyword is always the first entry in a line. It defines what kind of object will be described in the rest of the line. It must be spelled exactly as given in the following table:

Keyword	Meaning
INSTRUMENT	Import of Instruments and/or objects of the Physical Model
INITIALVALUE	Import of initial values for variables (for tuning purposes)
AREA	Import/Export an Area
PROCESSCELL	Import/Export a Process Cell
UNIT	Import/Export a Unit
EQUIPMENTMODULE	Import/Export an Equipment Module
CONTROLMODULE	Import/Export a Control Module
VARIABLE	Import/Export a variable
PLC	Import/Export a PLC
RACK	Import/Export a Rack
HWMODULE	Import/Export a hardware Module
IOPOINT	Import/Export an IO point
HMI	Import/Export an HMI
CONTROLDOMAIN	Import/Export a Control Domain
DATASERVER	Import/Export a Data Server
NETWORKNODE	Import/Export a Net Partner or Other Node
NETWORKSEGMENT	Import/Export a Network Segment
ROUTINGPATH	Import/Export a Routing Path

Line Feeds or Tabs in Free Text

Fields that contain free text, for example the comment of a Control Module may contain line feeds or tabs.

- A **line feed** has to be specified as `\n` in the CSV file.
- A **tab** has to be specified as `\t` in the CSV file.

Comment Line

Comment lines can be inserted in the CSV file. A **comment line** has to be specified with # as the first character in the line.

Order in Export Files

The order of elements in the export file corresponds to the trees of the Physical and Topological Models. The lines are generated subsequently, that is after the line for the parent object the lines for all contained child objects will follow.

Physical Model Elements: CSV File Format

General Remarks

Note the following:

- The **first element** is the keyword for the identification of the object. It must be exactly the string given in the table written in uppercase letters. The keywords are the same for all national languages of Unity Application Generator.
 - The **field names** are in English for all language versions of Unity Application Generator.
 - **Document references**, if they exist in the object, lie at the end of the line. Each document reference consists of two fields: the specific file and parameters of the specific tool. The number of document references is not restricted.
 - Fields with **free text**, for example a comment, may contain **line feeds** and **tabs**. A line feed has to be specified by `\n` a tab by `\t`.
 - The numbers in this table are only for your overview, they are **not** contained in the CSV file!
 - *: The **required fields** are marked with a star. The optional fields, when not filled, must remain empty (`;`).
 - The **separator** can be chosen in the Options. Do not use the separator character anywhere, for example in free text, otherwise the fields will not be identified correctly.
-

Area

The line in a CSV file for importing/exporting an area contains the following elements, separated by a separator (* = required fields).

1	* AREA	2	* <i>name</i>	3	<i>comment</i>
4	<i>document_ref_1</i>	5	<i>document_parameters_1</i>	...	<i>document_ref_i</i>
...	<i>document_parameters_i</i>	...	<i>document_ref_n</i>	...	<i>document_parameters_n</i>

Process Cell

The line in a CSV file for importing/exporting a Process Cell contains the following elements, separated by a separator (* = required fields).

1	* PROCESSCELL	2	* <i>name</i>	3	* <i>area_name</i>
4	<i>comment</i>	5	<i>document_ref_1</i>	6	<i>document_parameters_1</i>
...	<i>document_ref_i</i>	...	<i>document_parameters_i</i>	...	<i>document_ref_n</i>
...	<i>document_parameters_n</i>				

area_name: Name of the parent Area.

Unit

The line in a CSV file for importing/exporting a Unit contains the following elements, separated by a separator (* = required fields).

1	* UNIT	2	* <i>name</i>	3	* <i>processcell_name</i>
4	<i>comment</i>	5	<i>document_ref_1</i>	6	<i>document_parameters_1</i>
...	<i>document_ref_i</i>	...	<i>document_parameters_i</i>	...	<i>document_ref_n</i>
...	<i>document_parameters_n</i>				

processcell_name: Name of the parent Process Cell.

Equipment Module

The line in a CSV file for importing/exporting Equipment Modules contains the following elements, separated by a separator (* = required fields).

1	* EQUIPMENTMODULE	2	* <i>name</i>	3	* <i>unit_name</i>
4	* <i>description</i>	5	<i>plc_name</i>	6	<i>control_domain_name</i>
7	* <i>section_name</i>	8	<i>comment</i>	9	<i>document_ref_1</i>
10	<i>parameters_1</i>	...	<i>document_ref_i</i>	...	<i>parameters_i</i>
...	<i>document_ref_n</i>	...	<i>parameters_n</i>		

- **unit_name:** Name of the parent Unit.
- **plc_name** and **controldomain_name** fields can be left empty. In this case they will be set to Not_assigned by Unity Application Generator.
- **section_name:** Name of the programming section of the Equipment Module

Control Module

The line in a CSV file for importing/exporting a Control Module contains the following elements, separated by a separator (* = required fields).

1	* CONTROLMODULE	2	* <i>name</i>	3	* <i>equipmentmodule_name</i>
4	* <i>controlmodule_type</i>	5	<i>library_name</i>	6	* <i>description</i>
7	<i>plc_name</i>	8	<i>controlmodule_master</i>	9	<i>comment</i>
10	<i>document_ref_1</i>	11	<i>document_parameters_1</i>	...	<i>document_ref_i</i>
...	<i>document_parameters_i</i>	...	<i>document_ref_n</i>	...	<i>document_parameters_n</i>

- ***equipmentmodule_name***: Name of the parent Equipment Module.
- ***controlmodule_type***: Control Module Type defined in the Control Module library. In the case of a type-less Control Module this field is empty.
- ***library_name***: name of the library containing the Control Module Type. In the case of a type-less Control Module this field is empty.
- ***plc_name***: This field can be empty. Then it will be set as Not_assigned within the Unity Application Generator project.
- ***controlmodule_master***: Name of the Control Module defined as a Control Module master as full name, for example
EquipmentModuleName_ControlModuleName. The field can be empty. (This field is only used if the SCoD library defines master Control Modules.)

Variable

The line in a CSV file for importing/exporting a variable contains the following elements, separated by a separator (* = required fields).

1	<i>* VARIABLE</i>	2	<i>* name</i>	3	<i>* parent_type</i>
4	<i>* equipmentmodule_name</i>	5	<i>* controlmodule_name</i>	6	<i>* description</i>
7	<i>* data_type</i>	8	<i>* connection_type</i>	9	<i>* in_out_type</i>
10	<i>address_segment</i>	11	<i>address_offset</i>	12	<i>initial_value</i>
13	<i>timeout_behavior</i>	14	<i>timeout_value</i>	15	<i>command</i>
16	<i>inverse</i>	17	<i>variable_used</i>	18	<i>graphical_symbol</i>
19	<i>graphical_symbol_name</i>	20	<i>measurement_unit</i>	21	<i>scaling_min</i>
22	<i>scaling_max</i>	23	<i>value_boundary_min</i>	24	<i>value_boundary_max</i>
25	<i>access_level</i>	26	<i>state0_text</i>	27	<i>state1_text</i>
28	<i>alarm</i>	29	<i>alarm_1_priority</i>	30	<i>alarm_1_operator</i>
31	<i>alarm_1_limit</i>	32	<i>alarm_1_text</i>	33	<i>alarm_2_priority</i>
34	<i>alarm_2_operator</i>	35	<i>alarm_2_limit</i>	36	<i>alarm_2_text</i>
37	<i>alarm_3_priority</i>	38	<i>alarm_3_operator</i>	39	<i>alarm_3_limit</i>
40	<i>alarm_3_text</i>	41	<i>alarm_4_priority</i>	42	<i>alarm_4_operator</i>
43	<i>alarm_4_limit</i>	44	<i>alarm_4_text</i>	45	<i>alarm_5_priority</i>
46	<i>alarm_5_operator</i>	47	<i>alarm_5_limit</i>	48	<i>alarm_5_text</i>
49	<i>alarm_6_priority</i>	50	<i>alarm_6_operator</i>	51	<i>alarm_6_limit</i>
52	<i>alarm_6_text</i>	53	<i>alarm_7_priority</i>	54	<i>alarm_7_operator</i>
55	<i>alarm_7_limit</i>	56	<i>alarm_7_text</i>	57	<i>alarm_8_priority</i>
58	<i>alarm_8_operator</i>	59	<i>alarm_8_limit</i>	60	<i>alarm_8_text</i>
61	<i>archive</i>	62	<i>archive_name</i>	63	<i>communication_frame</i>
64	<i>free_property_1</i>	65	<i>free_property_2</i>	66	<i>free_property_3</i>
67	<i>free_property_4</i>	68	<i>free_property_5</i>	69	<i>free_property_6</i>
70	<i>free_property_7</i>	71	<i>free_property_8</i>	72	<i>free_property_9</i>
73	<i>free_property_10</i>	74	<i>free_property_11</i>	75	<i>free_property_12</i>
76	<i>free_property_13</i>	77	<i>free_property_14</i>	78	<i>free_property_15</i>
79	<i>free_property_16</i>	80	<i>free_property_17</i>	81	<i>free_property_18</i>
82	<i>free_property_19</i>	83	<i>free_property_20</i>		

Description of the entries:

- **parent_type**: Two keywords are allowed: `EQUIPMENTMODULE` or `CONTROLMODULE`. They describe the type of parent object of the variable.
- **equipmentmodule_name**: In the case of a free Equipment Module variable it is the name of the parent object. In the case of a Control Module variable it is the parent Equipment Module of the Control Module.
- **controlmodule_name**: In the case of a Control Module variable it is the name of the parent Control Module. In the case of a free Equipment Module variable the field remains empty.
- **data_type**: Data type of the variable defined in the customization (for example `BOOL`, `ANL_IN`, `ANL_OUT` and so on)
- **connection**: Connection type of the variable; one of the following strings `IO_PLC`, `PLC`, `PLC_HMI`, `PLC_NET`, `HMI`
- **in_out_type**: One of the following strings `IN`, `INOUT`, `OUT`
- **address_segment**: Address segment, a digit 0,1,3 or 4. The field can be empty.
- **address_offset**: Address in the scope of the specific address segment. The field can be empty.

Note: The address of a variable is exported, but ignored during import (will be recalculated by Unity Application Generator).

- **initial_value**: The format has to correspond to the data type of the variable (for example for `BOOL` it is 0 or 1, for `REAL` 1.23 and so on).
- **timeout_behavior**: One of the following strings: `Disabled`, `LastValue`, `UserDefined`. The field can be empty.
- **timeout_value**: Only 0 or 1 are allowed. The field can be empty.
- **command**: One of the following strings: `ViewOnly`, `Logic`, `Operator`, `Parameter`, `SetValue`. The field can be empty.
- **inverse**: Two strings are valid: `True` and `False`. If **inverse** is `True`, the variable value will be inverted in Concept. Valid only for Boolean variables of type `IO_PLC`.
- **variable_used**: Two strings are valid: `True` and `False`. If **variable_used** is `True`, the variable will be connected to the Function Block in Concept. Valid only for `IO_PLC` variables.
- **graphical_symbol**: Two strings are valid: `True` and `False`. If **graphical_symbol** is `True`, a graphical symbol for the variable will be generated for the HMI. Valid only for `PLC_HMI` type variables.
- **display_format**: A string defined in the customization for example `9999.9` or `99.9`. The field can be empty.
- **measurement_unit**: A string defined in the customization for example `%`, `m`, `Hz` or `1/min`. The field can be empty.
- **scaling_min**: The field value must correspond to the variable data type. Valid for `HMI` and `PLC_HMI` variables. The field can be empty.
- **scaling_max**: The field value must correspond to the variable data type. Valid for `HMI` and `PLC_HMI` variables. The field can be empty.
- **value_boundary_min**: The field value must correspond to the variable data type. Valid for `HMI` and `PLC_HMI` variables. The field can be empty.

- **value_boundary_max**: The field value must correspond to the variable data type. Valid for HMI and PLC_HMI variables. The field can be empty.
- **access_level**: One of the access levels defined in the customization editor for example Operator, Production, Technical and so on. The field can be empty.
- **state0_text**: Text to be displayed in HMI if the value of the variable is 0 - only for Boolean variables. The field can be empty.
- **state1_text**: Text to be displayed in HMI if the value of the variable is 1 - only for Boolean variables. The field can be empty.
- **alarm**: One of the following strings: True or False.
- **alarm_1_priority ... alarm_8_priority**: Each time one of the strings defined in the customization, for example low, medium, high, and so on. The fields can be empty.
- **alarm_1_operator ... alarm_8_operator**: Comparison operator for comparing the variable value against the respective alarm limit. The fields can be empty.
- **alarm_1_limit ... alarm_8_limit**: Values to be compared with the variable value. The fields can be empty.
- **alarm_1_text ... alarm_8_text**: Texts shown to the HMI operator in case of the respective alarms. The fields can be empty. A default alarm text is specified by \$.
- **archive**: One of the following strings: Historic or No. The field can be empty.
- **archive_name**: A string specifying the name of a database table in Monitor Pro. The field can be empty.
- **communication_frame**: A string specifying the name of the communication frame used to exchange the variable data with the HMI. The field can be empty.
- **free_property_1 ... free_property_20**: Values of the user definable free properties. The field can be empty.

Note:

- In case of a Control Module Type variable, the variable object will not be created in the database; only some of the properties will be set. When a property cannot be set because of the variable type, it will be ignored.
- If a variable is not defined in the library Control Module Type, it is treated as a free variable. All properties that are valid for a specific type of the variable are set, all others are ignored.

Topological Model Elements: CSV File Format

General Remarks

Note the following:

- The **first element** is the keyword for the identification of the object. It must be exactly the string given in the table written in uppercase letters. The keywords are the same for all national languages of Unity Application Generator.
 - The **field names** are in English for all language versions of Unity Application Generator.
 - **Document references**, if they exist in the object, lie at the end of the line. Each document reference consists of two fields: the specific file and parameters of the specific tool. The number of document references is not restricted.
 - Fields with **free text**, for example a comment, may contain **line feeds** and **tabs**. A line feed has to be specified by `\n` a tab by `\t`.
 - The numbers in this table are only for your overview, they are **not** contained in the CSV file!
 - *: The **required fields** are marked with a star. The optional fields, when not filled, must remain empty (`;`).
 - The **separator** can be chosen in the Options. Do not use the separator character anywhere, for example in free text, otherwise the fields will not be identified correctly.
-

PLC

The line in a CSV file for importing/exporting a PLC contains the following elements, separated by a separator (* = required field).

1	<i>* PLC</i>	2	<i>* plc_name</i>	3	<i>description</i>
4	<i>* plc_project_name</i>	5	<i>* plc_family</i>	6	<i>* local_rack_part_number</i>
7	<i>comment</i>	8	<i>battery_coil</i>	9	<i>timer_register</i>
10	<i>time_of_day</i>	11	<i>hwmodules_0x_start</i>	12	<i>hwmodules_0x_end</i>
13	<i>hwmodules_1x_start</i>	14	<i>hwmodules_1x_end</i>	15	<i>hwmodules_3x_start</i>
16	<i>hwmodules_3x_end</i>	17	<i>hwmodules_4x_start</i>	18	<i>hwmodules_4x_end</i>
19	<i>plc_communication_4x_start</i>	20	<i>plc_communication_4x_end</i>	21	<i>hmi_0x_start</i>
22	<i>hmi_0x_end</i>	23	<i>hmi_4x_start</i>	24	<i>hmi_4x_end</i>
25	<i>netpartner_0x_start</i>	26	<i>netpartner_0x_end</i>	27	<i>netpartner_4x_start</i>
28	<i>netpartner_4x_end</i>	29	<i>reserve_0x_start</i>	30	<i>reserve_0x_end</i>
31	<i>reserve_1x_start</i>	32	<i>reserve_1x_end</i>	33	<i>reserve_3x_start</i>
34	<i>reserve_3x_end</i>	35	<i>reserve_4x_start</i>	36	<i>reserve_4x_end</i>
37	<i>document_ref_1</i>	38	<i>document_parameters_1t</i>	39	<i>document_ref_i</i>
40	<i>document_parameters_i</i>	41	<i>document_ref_nt</i>	42	<i>document_parameters_n</i>

Description of entries:

- **plc_family**: Concept/Quantum, Concept/Momentum, Unity/Quantum Or Unity/Premium.
- **local_rack_part_number**: refer to table below.
- **battery_coil**: State ram address of battery coil. (Empty for Unity Pro)
- **timer_register**: State ram address of timer register. (Empty for Unity Pro)
- **time_of_day**: State ram address of time of day clock (first register). (Empty for Unity Pro)
- **hwmodule_nx_start ...hwmodule_nx_end**:
Quantum, Momentum: State ram address ranges (0x=%M, 1x=%I, 3x=%IW, 4x=%MW) for Modules.
Premium: Empty, because topological addresses are used.
- **plc_communication_4x_start ... plc_communication_4x_end**: State ram address range (4x=%MW) for PLC-PLC communication.
- **hmi_nx_start ... hmi_nx_end**: State ram address range (0x=%M, 4x=%MW) for HMI variables.
- **netpartner_nx_start ... netpartner_nx_end**: State ram address range (0x=%M, 4x=%MW) for netpartner variables.
- **reserve_nx_start ... reserve_nx_end**:
Reserved state ram address ranges (0x=%M, 1x=%I, 3x=%IW, 4x=%MW).
Premium: **reserve_1x_start**, **reserve_1x_end**, **reserve_3x_start** and **reserve_4x_end** are empty.

Table of **local_rack_part_number**:

local_rack_part_number	Size	Description
140 XBP 002 00	2	2 Slot Quantum Rack
140 XBP 003 00	3	3 Slot Quantum Rack
140 XBP 004 00	4	4 Slot Quantum Rack
140 XBP 006 00	6	6 Slot Quantum Rack
140 XBP 010 00	10	10 Slot Quantum Rack
140 XBP 016 00	16	16 Slot Quantum Rack
TSX RKY 6	6	Nonextendable 6 Position Rack
TSX RKY 8	8	Nonextendable 8 Position Rack
TSX RKY 12	12	Nonextendable 12 Position Rack
TSX RKY 6E	6	Extendable 6 Position Rack
TSX RKY 8E	8	Extendable 8 Position Rack
TSX RKY 12E	12	Extendable 12 Position Rack
TSX RKY 4EX	4	Extendable 4 Position Rack
TSX RKY 6EX	6	Extendable 6 Position Rack
TSX RKY 8EX	8	Extendable 8 Position Rack
TSX RKY 12EX	12	Extendable 12 Position Rack

Rack

The line in a CSV file for importing/exporting a Rack contains the following elements, separated by a separator (* = required field).

1	* RACK	2	* <i>part_number</i>	3	<i>description</i>
4	* <i>drop_number</i>	5	* <i>plc_name</i>	6	* <i>link_type</i>
7	* <i>extension_rack_number</i>				

Description of entries:

- **part_number**: refer to table below
- **drop_number**: Number of the rack (integer). For a local rack it is 1.
 Ranges: Remote 2 . . . 32, SYMAX: 2 . . . 32, 800I/O: 2 . . . 32, Distributed: 2 . . . 64 (=Modbus Plus address), Generic: 1 . . . 9999, IOBus: always 2, Modbus Plus I/O: no drop number necessary, Ethernet I/O: 1 . . . 16 (= slot number of the NOE or M1E (in case of M1E always 0)).
- **plc_name**: Name of the parent PLC.
- **link_type**: Type of the connection with the PLC. Valid values are: Local, Local Momentum, XBus, Remote, DIO0, DIO1, DIO2, SYMAX, 800IO, Generic, IOBUS, MBPIOLink0, MBPIOLink1, MBPIOLink2, ETHIO.
- **extension_rack_number**: Number of the extension rack. Empty string for Local Rack and number for Extension Racks.

Table of **part_number**:

part_number	Size	Description
140 XBP 002 00	2	2 Slot Quantum Rack
140 XBP 003 00	3	3 Slot Quantum Rack
140 XBP 004 00	4	4 Slot Quantum Rack
140 XBP 006 00	6	6 Slot Quantum Rack
140 XBP 010 00	10	10 Slot Quantum Rack
140 XBP 016 00	16	16 Slot Quantum Rack
TSX RKY 6	6	Nonextendable 6 Position Rack
TSX RKY 8	8	Nonextendable 8 Position Rack
TSX RKY 12	12	Nonextendable 12 Position Rack
TSX RKY 6E	6	Extendable 6 Position Rack
TSX RKY 8E	8	Extendable 8 Position Rack
TSX RKY 12E	12	Extendable 12 Position Rack
TSX RKY 4EX	4	Extendable 4 Position Rack
TSX RKY 6EX	6	Extendable 6 Position Rack
TSX RKY 8EX	8	Extendable 8 Position Rack
TSX RKY 12EX	12	Extendable 12 Position Rack

part_number	Size	Description
8030-RRK-100	5	5 Slot SY/MAX Rack
8030-RRK-200	9	9 Slot SY/MAX Rack
8030-RRK-300	18	18 Slot SY/MAX Rack
AS-H810-100	4	4 Slot 800 I/O Rack
AS-H810-208	4	4 Slot 800 I/O Rack
AS-H810-209	4	4 Slot 800 I/O Rack
AS-H819-100	7	7 Slot 800 I/O Rack
AS-H819-103	7	7 Slot 800 I/O Rack
AS-H819-209	7	7 Slot 800 I/O Rack
AS-H827-100	11	11 Slot 800 I/O Rack
AS-H827-103	11	11 Slot 800 I/O Rack
AS-H827-209	11	11 Slot 800 I/O Rack
800 I/O (55)	55	55 Slot 800 I/O Rack
Local Momentum	2	Momentum Local Rack
I/O Bus (128)	128	128 Slot I/O Bus Rack
I/O Bus (44)	44	44 Slot I/O Bus Rack
Ethernet I/O (128)	128	Ethernet I/O Rack (Quantum)
Ethernet I/O (64)	64	Ethernet I/O Rack (Momentum)
MBP I/O	64	ModbusPlus I/O Rack
Generic	255	Generic Rack

HW Module

The line in a CSV file for importing/exporting a HW module contains the following elements, separated by a separator (* = required field).

1	* HWMODULE	2	* hwmodule_name	3	* plc_name
4	* link_type	5	* drop_number	6	extension_rack_number
7	* slot_number	8	mbp_link	9	network_segment_1
10	network_address_1	11	network_segment_2	12	network_address_2
10	0x_address	11	1x_address	12	3x_address
13	4x_address	14	timeout_behavior	15	bus_children

Description of entries:

- **hwmodule_name**: Name of a valid Quantum HW module for example CPS-111-00 (power supply module) or CPU-113-03 and so on.
- **plc_name**: PLC in which the HW module exists.
- **link_type**: Type of the Rack. Valid field values are: Local, Local Momentum, XBus, Remote, DIO0, DIO1, DIO2, SYMAX, 800IO, Generic, IOBUS, MBPIOLink0, MBPIOLink1, MBPIOLink2, ETHIO.
- **drop_number**: Drop number of the corresponding Rack.
- **slot_number**: Number of the slot of the HW module.
- **mbp_link**: Link type of the Modbus Plus module. Only used for NOM-2xx-00, else empty. Valid values are 1 and 2.
- **network_segment_1/_2**: Name of the Network Segment. The field is valid only for CPU or Communication modules, otherwise it must be empty
- **network_address_1/_2**: Network address of the HW module. The field is valid only for CPU or Communication modules, otherwise it must be empty
- **0x_address**: 0x start state RAM address of the Module. The field can be empty. The address is exported, but ignored during import (recalculated by Unity Application Generator).
- **1x_address**: 1x start state RAM address of the Module. The field can be empty. The address is exported, but ignored during import (recalculated by Unity Application Generator).
- **3x_address**: 3x start state RAM address of the Module. The field can be empty. The address is exported, but ignored during import (recalculated by Unity Application Generator).
- **4x_address**: 4x start state RAM address of the Module. The field can be empty. The address is exported, but ignored during import (recalculated by Unity Application Generator).
- **timeout_behavior**: One of three strings: Disabled, LastValue, UserDefined. The field is valid only for digital output modules, otherwise it must be empty.
- **bus_children**: only for Momentum module 170-BNO-6x1-00, number of Local/Remote Bus Children; otherwise it must be empty.

I/O Point

The line in a CSV file for importing/exporting an I/O point contains the following elements, separated by a separator (* = required field).

1	* IOPOINT	2	* <i>variable_full_name</i>	3	* <i>plc_name</i>
4	* <i>link_type</i>	5	* <i>drop_number</i>	6	<i>extension_rack_number</i>
7	* <i>slot_number</i>		* <i>io_point</i>		

Description of entries:

- ***variable_full_name***: Full (generated) name of the variable mapped to the IO point; for example `EquipmentName_DeviceName_VariableName`.
- ***plc_name***: PLC containing the HW module.
- ***link_type***: Type of the Rack. Valid values are: `Local`, `Remote`, `DIO0`, `DIO1`, `DIO2`, `SYMAX`, `800IO` und `Generic`. Corresponds to the rack containing the I/O Point.
- ***drop_number***: Drop number of the Rack; integer value.
- ***slot_number***: Number of the slot of the HW module; integer value.
- ***io_point***: Position in the I/O module; integer value. In a mixed module with IN and OUT connections, the IN connections are counted first, for example in a module with 4 IN connections, the first OUT connection is 5.

HMI

The line in a CSV file for importing/exporting an HMI contains the following elements, separated by a separator (* = required field).

1	* HMI	2	<i>name</i>	3	* <i>hmi_type</i>
4	<i>comment</i>	5	<i>document_ref_1</i>	6	<i>document_parameters_1</i>
...	<i>document_ref_i</i>	...	<i>document_parameters_i</i>	...	<i>document_ref_n</i>
...	<i>document_parameters_n</i>				

type: Type of HMI. The valid values are `iFIX` or `Generic HMI`.

Control Domain

The line in a CSV file for importing/exporting a Control Domain contains the following elements, separated by a separator (* = required field).

1	* CONTROLDOMAIN	2	* <i>name</i>	3	* <i>hmi_name</i>
4	* <i>description</i>	5	<i>comment</i>		

Description of entries:

- ***hmi_name***: Name of the parent HMI.

Data Server

The line in a CSV file for importing/exporting a Data Server contains the following elements, separated by a separator (* = required field).

1	* DATASERVER	2	* <i>name</i>	3	<i>network_segment</i>
4	<i>network_address</i>	5	<i>timeout_min</i>	6	<i>timeout_sec</i>
7	<i>timeout_ms</i>	8	<i>redundant_network_segment</i>	9	<i>redundant_network_address</i>
10	<i>comment</i>	11	<i>export_format_name</i>	12	<i>document_ref_1</i>
13	<i>document_parameters_1</i>	...	<i>document_ref_i</i>	...	<i>document_parameters_i</i>
...	<i>document_ref_n</i>	...	<i>document_parameters_n</i>		

Description of entries:

- ***redundant_network_segment***: segment for optional redundant network card in data server PC.
- ***redundant_network_address***: address for optional redundant network card in data server PC.
- ***export_format_name***: Name of export format. Only for generic HMI.

Network Node

The line in a CSV file for importing/exporting a Network Node contains the following elements, separated by a separator (* = required field).

1	* NETWORKNODE	2	* <i>name</i>	3	* <i>network_node_type</i>
4	<i>description</i>	5	<i>network_segment</i>	6	<i>network_address</i>
7	<i>timeout_min</i>	8	<i>timeout_sec</i>	9	<i>timeout_ms</i>
10	<i>export_file</i>	11	<i>bitmap_file</i>	12	<i>comment</i>
13	<i>document_ref_1</i>	14	<i>document_parameters_1</i>	...	<i>document_ref_i</i>
...	<i>document_parameters_i</i>	...	<i>document_ref_n</i>	...	<i>document_parameters_n</i>

Description of entries:

- ***network_node_type***: Valid strings are NetPartner and OtherNode.

Network Segment

The line in a CSV file for importing/exporting a Network Segment contains the following elements, separated by a separator (* = required field).

1	* NETWORKSEGMENT	2	* <i>name</i>	3	* <i>network_type</i>
4	<i>subnetmask</i>	5	<i>default_gateway</i>		

Description of entries:

- **network_type**: Valid strings are ModbusPlus and Ethernet.
- **subnetmask**: Correct IP address. Only for Ethernet network type, otherwise this entry is empty.
- **default_gateway**: Correct IP address. Only for Ethernet network type, otherwise this entry is empty.

Routing Path

The line in a CSV file for importing/exporting a Routing Path contains the following elements, separated by a separator (* = required field).

1	* ROUTINGPATH	2	* <i>senders_segment_name</i>	3	* <i>receivers_segment_name</i>
4	* <i>network_path</i>				

network_path: One to four Modbus+ addresses separated by a dot, for example: 2.12.11.3 or 3.4 or 1.

Note: The line starting with ROUTINGPATH is only valid for Modbus Plus Network Segments.

Instruments: CSV File Format

General Remarks

Note the following:

- The **first element** is the keyword for the identification of the object. It must be exactly the string given in the table written in uppercase letters. The keywords are the same for all national languages of Unity Application Generator.
- The **field names** are in English for all language versions of Unity Application Generator.
- **Document references**, if they exist in the object, lie at the end of the line. Each document reference consists of two fields: the specific file and parameters of the specific tool. The number of document references is not restricted.
- Fields with **free text**, for example a comment, may contain **line feeds** and **tabs**. A line feed has to be specified by `\n` a tab by `\t`.
- The numbers in this table are only for your overview, they are **not** contained in the CSV file!
- *: The **required fields** are marked with a star. The optional fields, when not filled, must remain empty (`;`).
- The **separator** can be chosen in the Options. Do not use the separator character anywhere, for example in free text, otherwise the fields will not be identified correctly.

Order of Data in a Line

Each line in the CSV file for importing Instruments represents one Instrument. The line contains the following elements, separated by a separator (* = required field).

1	* INSTRUMENT	2	* <i>controlmodule_name</i>	3	* <i>controlmodule_type</i>
4	* <i>library_name</i>	5	* <i>controlmodule_description</i>	6	<i>controlmodule_comment</i>
7	<i>equipmentmodule_name</i>	8	<i>unit_name</i>	9	<i>processcell_name</i>
10	<i>area_name</i>				

The star * marks required fields. The not required fields when not filled must remain empty (`;`).

Note:

- The first element must be exactly the string INSTRUMENT written in uppercase letters to identify the contents of the line. This keyword is the same for all languages of Unity Application Generator.
- The numbers in this table are only for your overview, they are **not** contained in the CSV file!

Options

You have the options to either import Instruments without the hierarchy in the Physical Model or with the complete hierarchy up to the Area level:

If you want to...	Then...
Import only Instruments (without hierarchy in the physical model)	Omit the last four entries <i>equipmentmodule_name;</i> <i>unit_name;processcell_name;</i> <i>area_name.</i>
Import Instruments as Control Modules with complete hierarchy in the Physical Model	Specify all the four entries <i>equipmentmodule_name;</i> <i>unit_name;processcell_name;</i> <i>area_name.</i>

Example for a Line

```
INSTRUMENT;motor1;MSR01;IATBasic10;Motor for conveyer No 1;The motor can be operated in manual or auto mode;EquipmentModule1;Unit1;ProcessCell1;Area1
```

Initial Values: CSV File Format

Order of Data in a Line

Each line in the CSV file for importing initial values represents one initial value for one variable.
The line contains the following elements, separated by a separator (* = required fields).

1	* INITIALVALUE	2	* <i>variable_full_name</i>	3	* <i>initial_value</i>
---	----------------	---	------------------------------------	---	-------------------------------

Description of the entries.

- ***variable_full_name***: This entry contains the full name of the variable for which the initial value shall be changed, for example `EquipmentModuleName_ControlModuleName_VariableName`. The name is checked to be the name of an existing variable in the Unity Application Generator project.
- ***initial_value***: This entry contains the initial value for the variable. The value is checked to be valid according to the data type of the variable.

Note:

- The first element must be exactly the string `INITIALVALUE` written in uppercase letters to identify the contents of the line. This keyword is the same for all languages of Unity Application Generator.
- The numbers in this table are only for your overview, they are **not** contained in the CSV file!

**Example for a
Line**

```
INITIALVALUE;Boiler1_Motor1_FTR;0
```

Format of the XML File for Generic HMI



D

XML File Format for Generic HMI

Introduction

The data in the XML file is contained in a hierarchical structure formed by named elements. Elements may have named attributes and child elements as shown in the following table.

Note: A plus sign (+) appended to an element name in the child elements column means there is one or more element, an asterisk sign (*) means there are zero or more elements, otherwise there is exactly one element.

Elements XML file format

Element Name	Attribute Name	Description of Attribute Value	Child Elements
generic_hmi_file	version	Version of the file format	file_header, content_header, physical_model, topological_model, control_module_type_libraries
file_header	company	The string "Schneider Electric"	
	product	The string "Unity Application Generator" followed by a version string	
	date_time	Date and time of generation.	
	content	The string "Generic HMI data".	
content_header	uag_project_file	Path and file name of the UAG project.	
physical_model			areas
areas			area*
area	name	Name of the Area.	process_cells
process_cells			process_cell*
process_cell	name	Name of the ProcessCell.	units
units			unit*
unit	name	Name of the Unit.	equipment_modules
equipment_modules			equipment_module*

Element Name	Attribute Name	Description of Attribute Value	Child Elements
equipment_module	name	Name of the Equipment Module.	variables, control_modules
	description	Description of the Equipment Module.	
	plc	Name of the associated PLC.	
	section	Name of the PLC program section.	
	hmi	Name of the associated HMI.	
	controldomain	Name of the HMI's Control Domain.	
	picture	Name of the HMI picture for the Equipment Module.	
variables			variable *

Element Name	Attribute Name	Description of Attribute Value	Child Elements
variable	name	Full variable name, for example EquipmentModuleName_ ControlModuleName_ VariableName.	
	data_type	Data type of the variable (for example BOOL, ANL_IN, ANL_OUT and so on).	
	address_segment	The segment part of the state ram address of the variable, that is, 0, 1, 3 or 4.	
	address_offset	The offset part of the state ram address of the variable.	
	initial_value	Initial value of the variable. The format corresponds to the data type of the variable (for example for BOOL it is 0 or 1, for REAL 1.23 and so on).	
	description	The description of the variable is composed as follows: description of variable - description of Control Module - description of equipment Module.	
	graphical_symbol	One of the following strings: True or False	
	graphical_symbol_name	Name of the graphical symbol to be used for the variable, if the attribute graphical_symbol is True. The value can be empty.	

Element Name	Attribute Name	Description of Attribute Value	Child Elements
variable	communication_frame	Name of the communication frame the variable is assigned to. Corresponds to the name of one of the hmi_communication elements in the channel element. The value can be empty.	
	parent_type	The type of the variable's parent, that is, one of the following strings: equipment_module or control_module.	
	command	One of the following strings: ViewOnly, Logic, Operator, Parameter, SetValue. The value can be empty.	
	access_level	One of the strings defined in the customization, for example 1, 2, 3 and so on. The value can be empty.	
	scaling_min	The value corresponds to the variable's data type. The value can be empty.	
	scaling_max	The value corresponds to the variable's data type. The value can be empty.	
	value_boundary_min	The value corresponds to the variable's data type. The value can be empty.	
	value_boundary_max	The value corresponds to the variable's data type. The value can be empty.	
	measurement_unit	A string defined in the customization for example %, m, Hz or 1/min. The value can be empty.	

Element Name	Attribute Name	Description of Attribute Value	Child Elements
variable	display_format	A string defined in the customization for example 9999.9 or 99.9 and so on. The value can be empty.	
	state0_text	Text to be displayed in HMI if the value of the variable is 0 - only for Boolean variables. The value can be empty.	
	state1_text	Text to be displayed in HMI if the value of the variable is 1 - only for Boolean variables. The value can be empty.	
	archive	One of the following strings: No or Historic. The value can be empty.	
	archive_name	One of the strings defined in the customization for archives. The value can be empty.	
	archive_location	One of the strings defined in the customization for archive locations. The value can be empty.	
	alarm	One of the following strings: True or False.	
	alarm_group	One of the strings defined in the customization for alarm groups. The value can be empty.	
	alarm_operator1 ... alarm_operator8	For Boolean variables the string EQ, otherwise one of the following strings: EQ, NE, LT, GT, LE, GE. The value can be empty.	

Element Name	Attribute Name	Description of Attribute Value	Child Elements
variable	alarm_limit1 ... alarm_limit8	For Boolean variables one of the following integer values: 0 or 1; otherwise a number according to the variable's data type. The value can be empty.	
	alarm_priority1 ... alarm_priority8	One of the strings defined in the customization, for example low, medium, high and so on. The value can be empty.	
	alarm_text1 ... alarm_text8	Text shown to the HMI operator in case of alarm. The value can be empty.	
	userdefined_value1 ... userdefined_value18	Any string specified by the user. The value can be empty.	
control_modules			control_module*
control_module	name	Full name of the Control Module, for example EquipmentModuleName_ ControlModuleName.	variables
	description	Description of the Control Module.	
	type	Type of the Control Module as defined in the Control Module Type library.	
	library_name	Control Module Type library in which the Control Module is defined.	
	plc	Name of the associated PLC.	
topological_model			network_segments, data_servers, hmis, plcs
network_segments			network_segment*

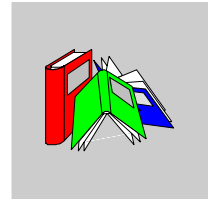
Element Name	Attribute Name	Description of Attribute Value	Child Elements
network_segment	name	Name of the network segment	
	network_type	Network type. Valid strings are ModbusPlus and Ethernet.	
	subnetmask	Subnet mask of the network segment.	
	default_gateway	Default gateway of the network segment.	
data_servers			data_server*
data_server	name	Name of the Data Server	
hmis			hmi*
hmi	name	Name of the HMI.	controldomains
	type	The string "Generic HMI"	
controldomains			controldomain*
controldomain	name	The name of the Controldomain.	
	description	The description of the Controldomain.	
plcs			plc*
plc	name	Name of the PLC.	channels, plc_variables
	description	Description of the PLC.	
	project_name	Name of the generated PLC project.	
	project_path	Path of the generated PLC project.	
	plc_family	One of the following strings: Concept/Quantum, Concept/Momentum, Unity/Quantum or Unity/Premium.	
channels			channel*

Element Name	Attribute Name	Description of Attribute Value	Child Elements
channel	name	Name of the Channel.	communication_paths
	communication_type	One of the following strings: data_server, plc, net_partner.	
	communication_partner	Name of the communication partner.	
communication_paths			communication_path+
communication_path	ordinal_number	The number 1 or 2.	
	plc_network_segment	The network segment to which the PLC is connected.	
	plc_network_address	The network address of the PLC.	
	partner_network_segment	The network segment to which the communication partner is connected.	
	partner_network_address	The network address of the communication partner.	
	global_path	Bridge path between communication partner and PLC. Only for ModbusPlus. The value can be empty.	
hmi_communication_frames			hmi_communication_frames*

Element Name	Attribute Name	Description of Attribute Value	Child Elements
hmi_communication_frames	Name	Name of the communication frame as specified in customization	
	data_type	One of the following strings: BOOL, WORD, DINT, REAL.	
	frame_type	One of the following strings: RD, EW, BW, UR, RD+EW.	
	address_segment	The segment part of the state ram addresses of the variables, that is, 0, 1, 3 or 4.	
	start_address	The offset part of the start address.	
	length	Length of the communication frame, specified in bits for data type BOOL and in words otherwise.	
plc_variables			plc_variable*
plc_variable	name	Full variable name, for example EquipmentModuleName_ ControlModuleName_ VariableName.	
control_module_type_libraries			control_module_type_library*
control_module_type_library	name	The name of the library	control_module_types
control_module_types			control_module_type*

Element Name	Attribute Name	Description of Attribute Value	Child Elements
control_module_type	name	The name of the Control Module Type	
	hmi_symbol_type	One of the following strings: ActiveX, Dynamo. The value can be empty.	
	hmi_symbol	The name of an HMI symbol for the Control Module Type. The value can be empty.	
	hmi_symbol_classid	The class ID of an ActiveX HMI symbol. The value can be empty.	

Glossary



A

- ActiveX** ActiveX is an open specification designed to govern how software components interact. ActiveX objects are pre-built, reusable software components that have interactive capabilities. An ActiveX is a programming construct that combines data (properties) with functions (methods) for using the data. Graphic objects in the HMI representing the Control Modules of Unity Application Generator are built as ActiveX controls.
- Administrator** Person or persons responsible to customize Unity Application Generator according to the companies requirements, e. g. naming conventions, documentation tools. The administrator is responsible for the user/user group administration.
- Alarm View** Overview mimic, which can be accessed via all alarm messages assigned to the alarm view.
- Analyzer** Part of Unity Application Generator that checks the formal validity of the total design.
- Area** A component of manufacturing site that is identified by physical, geographical, or logical segmentation within the site.
-

B

Batch

- The material that is being produced or that has been produced by a single execution of a batch process.
- An entity that represents the production of a material at any point in the process.

Batch means both the material made by and during the process and also an entity that represents the production of that material. Batch is used as an abstract contraction of the words "the production of a batch"

C

Channel

A Channel describes the following:

- To which objects of the control system the PLC communicates
- Via which communication path the communication takes place
- Via which alternative path the communication takes place
- Which variables have to be exchanged

The communication is always done through a PLC. This means, one communication partner in a Channel is always a PLC.

Control

(Process control) The control activity that includes the control functions needed to provide sequential, regulatory, and discrete control and to gather and display data (defined in S88).

Control Domain

Logical group of process elements which are controlled by an operator.

Assigned to a Control Domain are

- Operator screens
- Access rights
- Alarms

Control Engineer

Specialist in all aspects of a control system like PLC, HMI and networking rather than the process itself.

Control Module

Individual or collections of sensors and actuators and associated processing equipment that from a control view point is operated as a single entity, e.g. motor, valve, level switch, alarm light.

A Control Module is derived from a Control Module Type (SCoD).

Control Module Type	Control Module Types are generic types of Control Modules, which are tested and validated and can be re-used as often as needed.
Control Module Type Variable	Control Module Type variables are related to a Control Module Type. They are automatically generated for each instance of a Control Module Type.
CSV File	A CSV (comma separated value) file is an ASCII file containing a number of lines. Each line of a CSV file contains a number of entries, separated by a separator, for example a comma or a semicolon or a tab. CSV files can be used for exchanging data between different software programmes. Unity Application Generator allow import and export of CSV files.

D

Data Server	Server for all for all HMI related data (e.g. variables).
--------------------	---

E

Equipment Module	<p>A functional group of Control Modules that can carry out a finite number of specific minor processing activities.</p> <p>An Equipment Module is typically centered around a piece of process equipment (a weigh tank, a process heater, a scrubber, etc.) or process activities e. g. dosing and weighing.</p>
-------------------------	---

F

FBD	<p>Function Block diagram (FBD)</p> <p>One or several sections containing graphically represented networks consisting of functions, Function Blocks, and links.</p>
Free Control Module	<p>A Free Control Module is a Control Module which is not a generic Control Module Type. It does not define any logic for the PLC nor any predefined functionality for the HMI. Free Control Modules can have free variables assigned to it. They are typically used for process hardware for which no appropriate Control Module Type is available.</p>

Free Variable

Free variables are variables which are manually defined for

- Equipment Modules
- Control Modules

Function Block (instance) (FB)

A Function Block is a program organization unit which provides values for its outputs and internal variable(s) according to the algorithms defined in its Function Block type description, when executed as a specific instance. All values of the outputs and internal variables of a specific Function Block instance are maintained from one invocation of the Function Block to the next. Therefore, multiple calls of the same Function Block instance with the same arguments (values of input parameters) do not necessarily yield the same output value(s).

Each Function Block instance is graphically displayed by a rectangular block symbol. The name of the Function Block type is centered on top, inside the rectangle. The name of the Function Block instance is also on top, but outside the rectangle. It is automatically generated when building an instance, and should never be changed by the user (if the FB was generated by Unity Application Generator). Inputs are shown to the left, outputs to the right of the block. The names of the formal input/output parameters are shown inside the rectangle at the corresponding input/output points.

G

Generic Module

Generic modules are user defined hardware modules which are not known by name in Concept. The administrator adds them to the list of modules in the customization. He must specify the number of inputs and outputs, the number of 0x (%M), 1x (%I), 3x (%IW) and 4x (%MW) registers consumed and the possible rack type = Generic. Generic modules can only be configured in generic racks and are NOT generated. They can not be configured in Quantum/Premium racks and are not entered in the Concept/Unity Pro I/O mapping. The specifications made are only used for the reservation of state RAM addresses and for the addresses of the I/O variables.

Global Network Paths

Earlier term for Routing Paths (See , p. 407)

H

HMI

HMI = Human Machine Interface. Also called supervisory system. Through the HMI the operator controls the production process.

I

- Instrument List** List of all Instruments used in a control project.
- Instruments** The sensors and actuators used by the PLC to measure and modify the physical and chemical characteristics of the process.
-

M

- Memory Mapper** Part of Unity Application Generator. Generates all addresses of variables, IO definitions, communication parameters etc. for both, PLC and HMI consistently. The Memory Mapper can be run during the design procedure for checking the addressing. It is run automatically, when Unity Application Generator generates for Concept.
- ModConnect** ModConnect is an initiative to integrate third party modules into Concept. ModConnect partner modules are modules that are known by name in Concept, specified in the Concept ModConnect Tool. The administrator adds them to the list of modules in the customization. They have to be imported in Concept with ModConnect Tool. He must specify the exact name as in the Concept ModConnect Tool, the number of inputs and outputs, the number of 0x, 1x, 3x and 4x registers consumed and the possible rack type(s). ModConnect partner modules can be configured in Quantum racks and are generated in the Concept I/O map.
-

N

- Net Partner** A Net Partner is a node on the network that communicates with the PLC but is neither a Schneider PLC, an HMI or a Data Server. Examples are Magelis hand held panels, PLCs of different make, an intelligent sensor or actuator. It can communicate via Modbus Plus or Ethernet with other devices therefore the communication with the Net Partner is defined in Unity Application Generator.
- Network** A network is the interconnection of units along a shared data path that are communicating with each other via a common protocol.
-

Network Node A Network Node is an object in the Topological Model and consists of Net Partners and Other Nodes.

Network Segment Networks are divided in segments. This reduces the data transfer load per segment because only the data relevant for devices of the respective segment are transmitted on a Network Segment.

O

OLE Object linking and embedding

Operator Person or persons responsible for using the run-time process control system in order to control the process.

Other Node An Other Node is a node on the network that communicates with the PLC but is neither a Schneider PLC, an HMI, a Data Server nor a Net Partner. Examples are Operator PCs or intelligent drivers. It can communicate via Modbus Plus or Ethernet with other devices, but the communication with the node is not defined in Unity Application Generator. It is used for reserving a network address and displaying the node in the Topological Viewer, only.

P

P&ID Pipework and Instrument Drawing

Physical Model Defines the hierarchy of the Equipment Modules used in the process.

PLC Programmable Logic Controller is used to automate and control a process

Procedure The strategy for carrying out a process.

Process A sequence of chemical, physical, or biological activities for the conversion, transport, or storage of material or energy (defined in S88).

Process Cell A logical grouping of Equipment Modules that includes the Equipment Modules required for production of one or more batches. It defines the span of logical control of one set of process Equipment Modules within an Area.

Process control	The control activity that includes the control functions needed to provide sequential, regulatory, and discrete control and to gather and display data (defined in S88).
Process Engineer	Specialist in all aspects of process which has to be controlled rather than the control system.
Process Objects	An element of processing equipment which is represented in the PLC and supervisory control (HMI) with all of its attributes.

R

Routing Path	Routing Paths define the bridges and their addresses which interconnect different Network Segments. (In earlier versions a Routing Path was called global network path.)
---------------------	--

S

SCADA	S upervisory C ontrol and D ata A cquisition. In iFIX an HMI system is called SCADA system, a Data Server is called SCADA host.
Site	Logical unit to structure the process. Highest level element of the Physical Model of Unity Application Generator.
Supervisory Control (HMI)	Allows an operator to manage the running of the process via the process control system
System Administrator	Person or persons responsible for customizing Unity Application Generator according to the project needs.

T

Topological Model	In Unity Application Generator the Topological Model defines the architecture of the control system. It comprises the definition of the PLC hardware, HMIs and networks.
--------------------------	--

Topological Viewer	Graphical representation of the Topological Model in a separate screen on Unity Application Generator.
Type-less Control Module	<p>A type-less Control Module is a Control Module with no type assigned. It is neither a Control Module Type nor a Free Control Module.</p> <p>With type-less Control Modules the process engineer has the possibility to configure a Control Module in the Physical Model tree even, if he is not sure yet about the specific implementation.</p> <p>The control engineer can assign a type-less Control Module to a PLC. As long as the type stays undefined, nothing is generated, but the Analyzer prints a warning message. A type can be assigned at any time throughout the development process of the project. Once the type is defined, the properties and variables of this Control Module can be defined. It is not possible to change back to a type-less Control Module once the type has been assigned.</p>

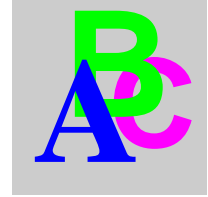
U

Unit	A collection of associated Control Modules and/or Equipment Modules and other process equipment in which one or more major processing activities can be conducted.
-------------	--

V

Variable	<p>In general a variable is a free definable parameter.</p> <p>Unity Application Generator distinguishes two major variable types</p> <ul style="list-style-type: none">● Control Module Type variables● Free variables
View Node	The term View node is used in iFIX for an HMI. It corresponds to the term HMI in Unity Application Generator.

Index



Numerics

21CFR11, 19

A

access level
 default values for access levels, 231
Acrobat Reader
 Software requirements, 267
ActiveX
 duplicates of SCoD ActiveX, 232
Additional Racks, 97
Address ranges, 88
Addressing
 Addressing in a hot standby system, 76
 Ethernet addressing rules, 71
Administrator
 Role, 22
Alarm
 Alarm related variable properties, 54
Alarm Area
 Alarm Areas configuration, 230
Alarm Group, 205
Alarming, 210
Analysis options
 Setting of, 259
Analyzer options for import, 368
And/Or
 In a search condition, Find feature, 140
Application
 Building an application, 22
 Steps to deploy the generated

 application to the iFIX nodes, 236
 Steps to document the application, 167
 Workflow to build an application, 155

Archive

 Configuration, 216
Archive Configuration, 220
Archive Name, 205
Archiving, 215

Area

 CSV file format, 370
 Description and properties, 29
 Generation for an Area, 287, 313
 Position in the Physical Model hierarchy,
 19
 Short definition, 28

Attribute

 In a search condition, Find feature, 140

B

Backup node, 238
Benefits of Unity Application Generator, 16
Block Write, 202
Branching, 223
Bridge, 69
 Defining a bridge between Ethernet
 segments, 70
 Defining a bridge between Modbus
 segments, 70
BW, 202

C

- CANOpen, 66
- Change tracking, 260
- Channel
 - Definition, 70, 90
 - Global Channel, 93
 - Specific Channel, 93
 - Steps to define the communication Channels, 162
- Client Builder
 - Mimics, 222
- Client.zip, 221
 - Steps for Client.zip, 221
- Coil, 202
- Command
 - Command related variable properties, 55
- Command register
 - Hot Standby command register, 77
- Comment line
 - Specify a comment line in a CSV file, 369
- Communication
 - Communication definition, 90
 - Communication failure (Ethernet), 96
 - Communication failure (Modbus Plus), 94
 - Communication methods, 91
 - Communication via Modbus Plus and Ethernet, 69
 - Generated code, 300, 326
 - How to define the communication, 70
 - PLC-PLC communication via Ethernet, 95
 - PLC-PLC communication via Modbus Plus, 93
- Communication Channel
 - Steps to define the communication channels, 162
- Communication Frame, 205
- Communication Parameters
 - Additional racks, 98
- Concept
 - Generation, 278
 - Rules concerning the PLC and Concept, 128
 - Software requirements, 267
 - Steps to generate PLC and HMI applications, 164
- Concept PLC configuration
 - Generated code, 294
- Concept variables
 - Generated code, 295
- Condition variable, 44
- Configuration
 - Rules for project configuration and generation, 126
- Connection type
 - List of connection types of a variable, 51
 - Of variables, 49
- Context menu, 137
- Control Domain, 211
 - Definition, 111
 - Properties, 113
- Control engineer, 16
 - Role, 22
- Control Module, 126
 - CSV file format, 372
 - Definition, 36
 - Features, 37
 - Free Control Module, definition, 40
 - Free variables, 37
 - Generated code, 297, 323
 - Generation for a Control Module, 290, 316
 - Interlocks for Control Modules, 43
 - Position in the Physical Model hierarchy, 19
 - Properties, 41
 - Representation in an HMI, 37
 - Representation in Concept, 37
 - Representation in Unity, 37
 - Selected Control Module, 232
 - Short definition, 28
- Control Module Type, 38
 - Definition, 36
 - Libraries, 36
 - Properties of Control Module Type variables, 51
 - What is defined in a Control Module Type, 39

- Control system, 62
 - Elements of a control system supported by UAG, 64
 - Control topology, 61
 - Figure showing a typical control topology, 63
 - Controller
 - Primary and standby controller, 75
 - Copy and paste
 - Copy and paste between projects, 145
 - Copy and paste in the Physical and Topological Model, 145
 - Copy and Paste of PLCs, 92
 - Criterion
 - In a search condition, Find feature, 140
 - CSV file
 - Comment line, 369
 - Definition, 184
 - For import into a Net Partner, 91
 - Format, 185
 - Format of CSV file for a Control Module, 372
 - Format of CSV file for a Process Cell, 371
 - Format of CSV file for a Unit, 371
 - Format of CSV file for a variable, 373
 - Format of CSV file for an Area, 370
 - Format of CSV file for an Equipment Module, 371
 - Format of CSV file for initial values, 386
 - Format of CSV file for Instruments, 385
 - Format of CSV file for Physical Model elements, 370
 - Format of the CSV file for the topological model, 376
 - General format, 368
 - Line feed in a free text field, 369
 - List of keywords, 369
 - Net Partner variables, 366
 - Order of objects in an export CSV file, 369
 - Steps to import/export CSV files, 186
 - Tab in a free text field, 369
 - Customization, 241
 - Access levels, 248
 - Alarm priority, 248
 - Changing of, 257
 - Defining the customization, 158
 - Generic HMI, 239
 - Display formats, 248
 - General options, 247
 - Measurement units and groups, 248
 - Naming conventions, 247, 249
 - Of Unity Application Generator, 243
 - PLC, 248
 - Properties of user defined modules, field descriptions, 253
 - Task of the administrator, 22
 - User defined modules - overview, 251
 - Valid data types, 248
 - Customization Editor, 127
 - Read-only, 127
 - Working with, 245
- ## D
- Data Files, 270
 - Data Point Logger, 215
 - Configuration, 219
 - Data Server, 114
 - Description and properties, 114
 - Relation to other elements, 114
 - Steps to define the Data Servers, 161
 - The data server group in the Topological Model, 68
 - Data type
 - Data type conversion for iFIX driver, 235
 - Data types for PLC-HMI communication, 92
 - Data types for PLC-PLC communication, 92
 - List of data types of a variable, 51
 - Of variables, 49
 - Database
 - Single process database, 20
 - Database Logger, 215
 - Configuration, 217
 - Database Logging Information table, 218
 - DBLCHISTMBX, 218

- Default gateway, 79
- Default Server Application, 197
- Default values, 135
- Directories
 - Specific, 270
- Display
 - Display related variable properties, 57
 - Refresh by user, 136
- Distributed project development, 171
 - Workflow of distributed project development, 174
- Document
 - Steps to document the application or individual objects, 167
- DPLOGHISTMBX, 219
- Driver
 - Unity Application Generator and iFIX drivers, 235

E

- Enhanced Ethernet Module, 97, 107
- Enhanced Ethernet Rack, 97
- Equipment Modul, 126
- Equipment Module
 - CSV file format, 371
 - Definition, 30
 - Features, 32
 - Generation for an Equipment Module, 289, 315
 - In relation to PLC, HMI, variables, 32
 - Position in the Physical Model hierarchy, 19
 - Properties, 33
 - Short definition, 28
- Equipment Modules
 - Generated code, 297, 323
- Ethernet, 69, 71
 - Communication method, 91
 - PLC-PLC communication via Ethernet, 95
 - Properties for Ethernet Network
 - Segments, 79
- Event, 91
- EW, 202
- Exception Write, 202

- Export
 - Examples for export, 184
 - Interfaces with other tools (import and export features), 183
- Export Format
 - Generic HMI, 239
- Export/import
 - Steps to import/export CSV files, 186

F

- Finalization
 - Steps to finalize the project, 162
- Find
 - Find objects with search criteria, 140
- Fipio, 66
- Format of CSV files, 368
- Frame Type Name, 206
- Free Control Module
 - Definition, 40
- Free text field in a CSV file
 - Line feeds and tabs, 369
- Free variable
 - Create free variables, 138
 - Of a Control Module, 37
 - Properties of free variables, 51
 - Symbol, 222
- Function
 - Complex functions like SIN, COS in an interlock definition, 149

G

- Generate
 - Steps to generate a new iFIX application, 233
 - Steps to generate an iFIX application incrementally, 234
 - Steps to generate import files for Net Partners, 166
 - Steps to generate PLC and HMI applications, 164
- Discrete Configuration (Premium only)
 - , 329

- Generated code
 - Communication, 300, 326
 - Concept PLC configuration, 294
 - Concept variables, 295
 - Control Modules, 297, 323
 - Equipment Modules, 297, 323
 - Generation principles, 277
 - Hot Standby, 303
 - iFIX database objects, 348
 - iFIX driver configuration from driver point of view, 361
 - iFIX driver configuration from Unity Application Generator point of view, 359
 - iFIX pictures, 354
 - iFIX screens, 347
 - Initialization, 302
 - Initialization (Quantum only), 328
 - Interlocks, 297, 323
 - Monitor Pro database objects, 336
 - Monitor Pro pictures, 340
 - Monitor Pro screens, 335
 - Net Partner variables, 366
 - Overview, 276
 - Scaling of analog values (Quantum only), 328
 - Scaling of analog values, 303
 - Unity Pro PLC configuration, 320
 - Unity Pro variables, 320
 - Variables for Monitor Pro, 333, 329
 - Variables for iFIX, 346
- Generated logic
 - Rules concerted generated logic, 128, 130
- Generated objects
 - Generated I/O scanner configuration, 96
- Generation
 - code generation for individual PLCs, 173
 - Concept, 278
 - iFIX, 344
 - Generic HMI, 363
 - manual configurations before generation for iFIX, 230
 - Monitor Pro, 332
 - net partner, 365
 - New or incremental generation, 126
 - Rules for project configuration and generation, 126
 - Unity Pro, 304
- Generation for Concept
 - Results from general project settings, 280
 - Results from settings for Physical Model, 287
 - Results from settings for Topological Model, 281
- Generation for Unity Pro
 - Results from general project settings, 306
 - Results from settings for Physical Model, 313
 - Results from settings for Topological Model, 307
- Generation options
 - Setting of, 259
- Generation principles
 - Generated code, 277
- Generation status
 - Viewing the generation status, 154
- Generic HMI
 - Customization, 239
 - Export Format, 239
 - Generation, 363
 - Using Unity Application Generator with a generic HMI, 239
 - XML file format, 389
- Generic module
 - Definition, 251
 - How to define a generic module with the Unity Application customization editor, 255
- Global Channel, 93
 - Set up a Global Channel, 94
- Guideline
 - Rules for working with Unity Application Generator, 125

H

- Handling
 - Handling of Unity Application Generator, 133
- Hardware Module
 - Moving hardware Modules, 139
- Hardware requirements], 266
- Health check
 - Of PLC-PLC communication via Ethernet, 95
 - Of PLC-PLC communication via Modbus Plus, 93
- Hierarchy
 - Of the Physical Model, 19
- HMI
 - Communication method, 91
 - Definition, 111
 - HMI related variable properties, 53
 - Properties, 112
 - Rules concerning HMI, 131
 - Steps to define the HMI(s), 161
 - Steps to generate PLC and HMI applications, 164
 - Supported HMIs and their setup, 193
 - The HMI Group, 110
 - The HMI group in the Topological Model, 68
 - Using Unity Application Generator with a generic HMI, 239
- HMI related variable properties, 53
- Hot Standby, 74
 - Allocation of registers in a Hot Standby system, 76
 - Command register, 77
 - Definition and example, 74
 - Functioning, 75
 - Generated code, 303
 - Hot Standby and addressing, 76
 - Limitations, 75
- HREG, 202
- HSBY, 74

I

- I/O map, 294
- I/O scanner
 - Generated I/O scanner configuration, 96
- I/O Scanner table, 109
- I/O Translator Dataset Definition, 208
- I/O Translator Protocol Driver Definition, 207
- iFIX
 - Configuring iFIX redundancy, 238
 - Generation, 344
 - iFIX and Unity Application Generator, 226
 - Manual configurations before generation for iFIX, 230
 - Rules for iFIX, 131
 - Software requirements, 267
 - Steps to configure iFIX for the use with Unity Application Generator, 228
 - Steps to deploy the generated application to the iFIX nodes, 236
 - Steps to generate a new iFIX application, 233
 - Steps to generate an iFIX application incrementally, 234
 - Steps to run an existing Unity Application Generator project, 237
 - Unity Application Generator and iFIX drivers, 235
- iFIX database objects
 - Generated code for iFIX database objects, 348
- iFIX driver configuration
 - Generated values for a channel, 361
 - Generated values for a datablock, 362
 - Generated values for a device, 361
 - Generation for a Channel, 360
 - Generation for a Data Server, 359
- iFIX driver configuration from driver point of view
 - Generated code, 361
- iFIX driver configuration from Unity Application Generator point of view
 - Generated code, 359
- iFIX node
 - Terminology in iFIX, 236

-
- iFIX pictures
 - Generated code for iFIX pictures, 354
 - Generation for a control domain, 354
 - Generation for a Control Module, 357
 - Generation for a PLC, 355
 - Generation for a process cell, 355
 - Generation for a unit, 355
 - Generation for a variable, 357
 - Generation for an area, 355
 - Generation for an Equipment Module, 356
 - Generation for an HMI, 354
 - Generation for an Interlock, 357
 - iFIX screens
 - Generated code for iFIX screens, 347
 - iFIX variables
 - Generated code for iFIX variables, 346
 - Import
 - Analyzer options for import, 368
 - Example
 - Import Instruments/Physical Model
 - Import initial values for variables, 191
 - Hierarchy, 188
 - Examples for import, 184
 - Interfaces with other tools (import and export features), 183
 - Steps to generate import files for Net Partners, 166
 - Import/export
 - Steps to import/export CSV files, 186
 - Incremental generation
 - New or incremental generation, 126
 - initial value
 - example
 - import initial values for variables, 191
 - Of Concept variables, 49
 - Of Unity Pro variables, 50
 - Format of CSV file for initial values, 386
 - initialization
 - Generated code, 302
 - Initialization (Quantum only)
 - Generated code, 328
 - Input
 - Checking all inputs, 135
 - Installation
 - Information for new users, 268
 - Instrument, 42
 - Definition, 42
 - Format of CSV file for Instruments, 385
 - Import Instruments, 188
 - Properties, 42
 - Purpose, 42
 - Instrument List
 - Working with the Instrument List, 142
 - Interface
 - Concept of open interfaces for import and export, 184
 - Concepts of the user interface, 134
 - Interfaces with other tools (import and export features), 183
 - Interlock, 43
 - Characterization, 43
 - Copy, 45
 - Generated code, 297, 323
 - How to build an interlock definition, 147
 - Interlock condition, 44
 - Interlock definition, 43
 - Complex functions like SIN, COS in an interlock definition, 149
 - Generation for an interlock definition, 290, 316
 - Interlock input, 43
 - Interlock list
 - Working with tables (lists), 137
 - Interlock variable, 44
 - Introduction of UAG, 15
 - IO scanner, 91
 - PLC-PLC communication method, 95
 - IO statistics, 89
 - IoXlator Tag Definition, 209
 - ISA S88.01
 - Differences to ISA S88.01 naming conventions, 28
 - Physical Model according to, 19
- ## K
- Keyword
 - List of keywords in CSV files for import and export, 369
-

L

LAN redundancy, 238

Libraries

Control Module Type libraries, 36

LIKE

LIKE operator in a search condition, 140

Line feed

Specify a line feed in a free text field in a CSV file, 369

Link number, 152

Links between SCoDs, 46

List

Working with tables (lists), 137

Logging, 215

M

M1E

Communication capabilities, 95

Supported Momentum Ethernet processor adapters, 72

Magelis Export, 118

Mailbox, 215

Monitor Pro, 197

Management

Of versions, 260

Merge, 171

Merging of Physical Models, 178

Merging of Topological Models, 180

Preconditions for project merge, 177

Workflow of distributed project development, 174

Message window

Result of a Find action in the message window, 141

Modbus Plus, 69

Communication method, 91

PLC-PLC communication via Modbus Plus, 93

Modbus TCP/IP

Modbus TCP I/P I/O driver, 231

Modbus TCP/IP Driver Dataset Definition, 207

Modbus TCP/IP Driver Device Definition, 206

ModConnect partner module

definition, 251

How to define a ModConnect partner module with Unity Application customization editor, 256

MODTIOMbx, 207

Module, 98

Drag & Drop of objects, modules and variables, 139

Hot Standby module 140 CHS 110 00, 75

Supported Momentum Ethernet

processor adapters, 72

Supported Premium Ethernet modules, 73

Supported Quantum Ethernet modules, 72

Momentum

Supported Ethernet processor adapters, 72

Monitor Pro, 266

Alarming, 210

Archiving, 215

Client Application, 221

Configuration Explorer, 201

Data conversion, 202

Data Server, 197

Data types, 202

Generation, 332

Generation Mode, 200

Incremental Generation, 200

Logging, 215

Mailbox, 197

Monitor Pro and Unity Application

Generator, 195

New Generation, 200

Rules for generation, 201

Rules for Monitor Pro, 131

Setup, 197

Software requirements, 267

Tag, 197

Monitor Pro database objects

Generated code for Monitor Pro database objects, 336

Monitor Pro pictures
Generated code for Monitor Pro pictures, 340
Generation for a control domain, 340
Generation for a Control Module, 342
Generation for a PLC, 340
Generation for a process cell, 341
Generation for a unit, 341
Generation for a variable, 343
Generation for an area, 341
Generation for an Equipment Module, 341
Generation for an HMI, 340

Monitor Pro screens
Generated code for Monitor Pro screens, 335

Monitor Pro variables
Generated code for Monitor Pro variables, 333

MS Word
Software requirements, 267

MSTR, 93, 95

Multiple users, 134

N

Name
Rules concerning names, 129, 130

Naming convention
For variables, 48

Navigation, 134
Use the topological view for navigation to property dialogs, 151

Net Partner, 117
Communication method, 91
Definition and description, 118
Generation, 365
Properties, 119
Steps to generate import files for Net Partners, 166

Net Partner variables
CSV file, 366

Network
How to define the network, 70
Networks supported by UAG, 65
Steps to define the network, 160
The network group in the Topological Model, 68

Network Nodes, 80, 117
Steps to define the Network Node(s), 161
The Network Nodes group in the Topological Model, 68

Network Segment
Definition, 69, 79
Description and properties, 79
Local Ethernet Network Segments, 71
What kind of Network Segments are supported, 69

Network Segments
Group in the Topological Model tree, 78

New
Item in context menu of lists, 137

New generation
New or incremental generation, 126

New Variable dialog, 138

Node
Ethernet nodes and rules for IP addressing, 72

Nodes
Network, 80

NOE
Communication capabilities, 95
Supported Quantum Ethernet Communication modules, 72

NOM, 93

Number
Creating several free variables, 138
Creating several Instruments, 143

O

Object
Drag & Drop of objects, modules and variables, 139

Office Ethernet, 69

Offset
Input/Output, 107

Older versions of UAG, 270

- Operating system
 - Software requirements, 267
 - Operator
 - In a search condition, Find feature, 140
 - Other Node, 117
 - Definition and description, 120
 - Properties, 121
 - Output of UAG, 18
- P**
- P&ID
 - Import Instruments from the P&ID, 188
 - Parameter, 91
 - Peer Cop, 91
 - PLC-PLC communication method, 93
 - Physical Model, 25
 - According to ISA S88.01, 19
 - Copy and paste in the Physical and Topological Model, 145
 - Format of CSV file for Physical Model elements, 370
 - Hierarchy, 19, 27
 - Import Physical Model hierarchy, 188
 - Merging of Physical Models, 178
 - Overview, 27
 - Short definition of elements, 28
 - Steps how to define the Physical Model, 159
 - Steps to complete the Physical Model, 163
 - Structure, 27
 - Physical Model tree, 17
 - Drag & Drop in the Physical Model tree, 139
 - Plant Ethernet, 69
 - PLC
 - Channels, 90
 - Code generation for individual PLCs, 173
 - Communication method, 91
 - Copy and paste, 92
 - Properties, 87
 - Rules concerning the PLC and Concept, 128
 - Rules concerning the PLC and Unity Pro, 130
 - Steps to define PLCs, 160
 - steps to generate PLC and HMI applications, 164
 - The PLC group in the Topological Model, 68
 - PLC configuration, 294, 320
 - Rules concerning PLC configuration, 128
 - PLC generation
 - Steps to generate the PLCs, 162
 - PLC group, 84
 - Introduction, 85
 - Structure and screen shot, 86
 - PLC project path, 158
 - PLC-PLC communication
 - Via Ethernet, 95
 - Via Modbus Plus, 93
 - PNN
 - Communication capabilities, 95
 - Premium
 - Supported Ethernet modules, 73
 - Principles, 15
 - Primary controller, 75
 - Primary node, 238
 - Process Cell
 - CSV file format, 371
 - Description and properties, 29
 - Generation for a Process Cell, 287, 314
 - Position in the Physical Model hierarchy, 19
 - Short definition, 28
 - Process designe
 - Task of the process engineer, 22
 - Process engineer
 - Role, 22
 - Process engineer, 16
 - Processor adapter
 - Supported Momentum Ethernet processor adapters, 72
 - Project
 - Maintenance, 258
 - Steps to run an existing Unity Application Generator project, 237

- Project configuration
 - Rules for project configuration and generation, 126
 - Project documentation
 - Report generator, 262
 - Project Maintenance, 241
 - Project upgrade to UAG 2.0 and Concept V2.6, 271
 - Properties, 53
 - Area, Process Cell and Unit properties, 29
 - Alarm related variable properties, 54
 - Command related variable properties, 55
 - Control Domain properties, 113
 - Control Module, 41
 - Data Server properties, 114
 - Display related variable properties, 57
 - Equipment Module properties, 33
 - HMI properties, 112
 - Instrument properties, 42
 - Net Partner properties, 119
 - Network Segment properties, 79
 - Other Node properties, 121
 - PLC properties, 87
 - Routing Path properties, 82
 - Variable properties, 51
- Q**
- Quantum
 - Supported Ethernet modules, 72
- R**
- Rack, 98
 - RD, 202
 - Read Data, 202
 - Redundancy
 - Hot Standby configuration, 74
 - Possibilities for setting up redundancy with UAG, 66
 - Redundancy
 - Channel redundancy for PLC-PLC communication via Ethernet, 96
 - Channel redundancy for PLC-PLC communication via Modbus Plus, 93
 - Configuring iFIX redundancy, 238
 - LAN redundancy, 238
 - Protection against NOE module failure, 235
 - Refresh, 136
 - Refresh rate for reading PLC values, 91
 - Register
 - Allocation of registers in a Hot Standby system, 76
 - Hot Standby command register, 77
 - Reserve registers, 88
 - Release Notes Version 2.1, 265
 - New features, 266
 - Software requirements, 267, 266
 - Upgrade of projects, 271
 - Remote nodes, 231
 - Repair of project database
 - Trouble shooting, 262
 - Report
 - Include the topological view into a Word report file, 151
 - Steps to document the application or individual objects, 167
 - Report generator
 - Project documentation, 262
 - Reserve registers, 88
 - Restrictions
 - Fipio and CANOpen, 66
 - Results from general project settings
 - Generation for Concept, 280
 - Generation for Unity Pro, 306
 - Results from settings for Physical Model
 - Generation for Concept, 287
 - Generation for Unity Pro, 313
 - Results from settings for Topological Model
 - Generation for a HW module, 283, 309
 - Generation for a PLC - PLC Channel, 284, 310
 - Generation for a PLC - PLC Channel - other objects, 286, 312
 - Generation for a rack, 282
 - Generation for Concept, 281
 - Generation for PLC, 281, 307
 - Generation for Unity Pro, 307

- Reverse transfer register (Hot Standby), 77
- Router, 69
- Routing Path
 - Description and properties, 82
 - Example path definition, 83
 - Group in the Topological Model tree, 78
- Rule
 - Rules for project configuration and generation, 126
 - Rules concerning HMI, 131
 - Rules concerning the PLC and Concept, 128
 - Rules concerning the PLC and Unity Pro, 130
 - Rules for working with Unity Application Generator, 125
- Run
 - Steps to run an existing Unity Application Generator project, 237

S

- Save command, 135
- Save line, 138
- SCADA node
 - Terminology in iFIX, 236
- Scaling of analog values (Quantum only)
 - Generated code, 303, 328
- SCoD, 38
 - Definition, 36
 - Links between SCoDs, 46
- SCoD (Smart Control Device)
 - Definition, 21
- SCoD Editor, 127
 - Read-only, 127
- Search
 - Find objects with search criteria, 140
- Search condition
 - Components of a search condition, 140
- Security Area
 - Security Area configuration, 230
- Segment
 - Local Ethernet Network Segments, 71
 - Requirements for Ethernet segment, 71
- Select from existing global input, 94
- Separator character in CSV files, 368

- Site
 - Position in the Physical Model hierarchy, 19
 - Short definition, 28
- Smart Control Device SCoD, 36
- Software requirements, 267
- SPANG Power Electronics, 107
- Specific Channel, 93
 - Set up an Ethernet Channel, 96
 - Set up a Specific Channel (Modbus Plus), 94
- Standby controller, 75
- Startnumber
 - Creating several free variables, 138
 - Creating several Instruments, 143
- Status register (Hot Standby), 77
- Subnet mask, 79
- Symbol
 - Free variable, 222
- Synoptic, 91
- System architecture
 - Task of the control engineer, 23

T

- Tab
 - Specify a tab in a free text field in a CSV file, 369
- Table
 - Working with tables (lists), 137
- Tag
 - Monitor Pro, 197
- TAGDATA, 219
- The Data Server group, 114
- Topological Model, 59
 - Copy and paste in the Physical and Topological Model, 145
 - Graphical representation of the Topological Model with the Topological Viewer, 151
 - Format of the CSV file for the topological model, 376
 - Merging of Topological Models, 180
 - Overview of the elements of the Topological Model, 67

- Screen shot of an example Topological Model, 67
- Steps how to define the Topological Model, 160
- Structure of the Topological Model, 67
- Topological Model tree, 17
- Topological Viewer, 151
 - Screen shot, 152
- Topology
 - Control topology, 61
 - Figure showing a typical control topology, 63
 - Topology of a control system, 62
- Topology of a control system, 62
- Trouble shooting
 - Repair of project database, 262

U

- Unit
 - CSV file format, 371
 - Description and properties, 29
 - Generation for a Unit, 288, 314
 - Position in the Physical Model hierarchy, 19
 - Short definition, 28
- Unity Pro
 - Generation, 304
 - Rules concerning the PLC and Unity Pro, 130
 - Software requirements, 267
 - Steps to generate PLC and HMI applications, 164
- Unity Pro PLC configuration
 - Generated code, 320
- Unity Pro variables
 - Generated code, 320
- Unsolicited Read, 202
- Update
 - Steps to update PLC and HMI application, 165
- Upgrade of projects, 271
- UR, 202
- User defined module
 - Overview, 251
 - Properties, 253

- User interface
 - Concepts of the user interface, 134

V

- Variable, 47
 - Assigning variables, 139
 - Create free variables, 138
 - CSV file format, 373
 - Connection types, Data types, Initial values, 49
 - Drag & Drop of objects, modules and variables, 139
 - Example
 - Import initial values for variables, 191
 - General properties, 51
 - Generation for a variable, 291, 317
 - HMI related variable properties, 53
 - Naming conventions, 48
 - Properties of Control Module Type variables, 51
 - Properties of free variables, 51
 - Rules concerning variables, 128, 130
 - Significance of variables, 48
- Version management, 260
- View node
 - Terminology in iFIX, 236

W

- Wildcard character
 - Wildcard characters in a search condition, 141
- Workflow
 - Workflow of distributed project development, 174
 - Workflow to build an application, 155

X

- XML file
 - File format, 389

