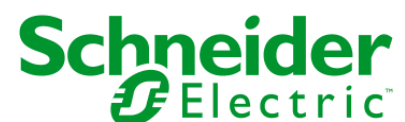
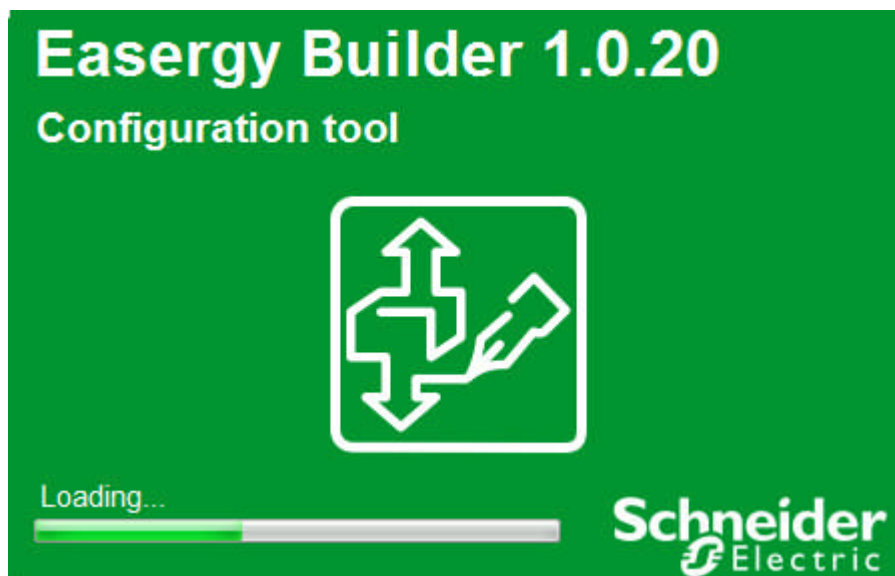


# BaseLine Software Platform

## RTU Configuration Tool

07/2016

## Easergy Builder



## Change Control

Rev	Date	Description
1.0	09-08-2015	Initial revision
1.1	30-09-2015	<ul style="list-style-type: none"> <li>Adapted to version 01.00.11 of Easergy Builder.</li> <li>Included Easergy T300 RTU (HU 250).</li> </ul>
1.2	05-02-2016	<ul style="list-style-type: none"> <li>Completed information about user rights.</li> </ul>
1.3	05-07-2016	<ul style="list-style-type: none"> <li>Updated with the revision 01.00.20 of Easergy Builder.</li> <li>Formula and Lioc devices have been included.</li> </ul>

## Relevant information for the user

As a result of the multiple uses of the product, the personnel in charge of the application and use of this control device must ensure these usages comply with all safety and performance requirements applicable in each application. The requirements include the applicable industry-related laws, norms, regulations and standards.

Throughout this manual some notes are included in order to alert the user about specific circumstances.

### NOTICE

**NOTICE** box identifies information about practices and circumstances which could result in a malfunction of the equipment.

## Restricted Liability

Electrical equipment should be serviced and maintained only by qualified personnel.

No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this manual. This document is not intended as an instruction manual for untrained persons.

The illustrations, dialog boxes, programming models and examples shown in this manual are intended for exemplary purposes. As there are installation-specific variables and requirements, Schneider Electric will not be held responsible for the misuse of the equipment based on the examples herein published.

### NOTICE

An inadequate use of the equipment, or misuse by ignoring these specifications, may compromise the system's security.

It is highly recommendable to backup the application programs frequently using the appropriate storage media to avoid potential data loss.

The Saitel platform and all its components have been developed in accordance to the requirements for a quality management system, complying with the ISO 9001 Norm.	
Document:	TE-HG-0000-MSS-S856
Revision / Date:	Rev 1.3 / 15-05-2016
File:	User Manual of Easergy Builder_EN_Rev1.3.pdf
Retention period:	Permanent throughout its validation period + 3 years after its cancellation.



For any request, problem report or suggestion about the equipment, please contact us through the following email:

[es-infoSaitel@schneider-electric.com](mailto:es-infoSaitel@schneider-electric.com)

# Table of Contents

Chapter 1 - BaseLine Software Platform .....	1-1
1.1 Software Architecture .....	1-1
1.2 Main Components .....	1-3
1.3 Software Tools .....	1-4
Chapter 2 - Introduction to Easergy Builder .....	2-1
2.1 Introduction.....	2-1
2.2 Easergy Builder Installation .....	2-1
2.2.1 Easergy Builder Requirements .....	2-1
2.2.2 Easergy Builder and Plugins .....	2-3
2.2.3 Easergy Builder Update and Uninstall .....	2-4
2.3 Environment Description .....	2-5
2.3.1 Workspace Mode.....	2-5
2.3.2 Configuration Mode .....	2-9
2.4 Getting Started with Easergy Builder.....	2-10
Chapter 3 - Working with RTUs.....	3-1
3.1 Introduction.....	3-1
3.2 RTU Administration .....	3-2
3.2.1 Adding an RTU.....	3-2
3.2.2 Configuring RTU Parameters .....	3-3
3.2.3 Removing an RTU.....	3-15
3.2.4 Rebooting an RTU.....	3-15
3.2.5 Reading the RTU Configuration .....	3-15
3.2.6 Loading a Configuration .....	3-16
3.2.7 Reading the sysLog File from the RTU.....	3-17
3.2.8 Adding a New Configuration.....	3-17
Chapter 4 - Working with Configurations.....	4-1
4.1 Introduction.....	4-1
4.2 Devices.....	4-1
4.2.1 Supervision Device.....	4-2
4.2.2 Formula Device .....	4-9
4.3 Channels .....	4-12
4.3.1 Adding a Channel .....	4-13
4.3.2 Adding a Link.....	4-16
4.3.3 Deleting and Editing a Channel or Link .....	4-17
4.4 Synchronization.....	4-17
4.4.1 Configuring a Synchronization Device (as Source) .....	4-18
4.4.2 Time Configuration .....	4-19
Chapter 5 - coreDb - Real Time DataBase.....	5-1
5.1 Introduction.....	5-1
5.2 Main Menu.....	5-1
5.3 coreDb Tabs.....	5-1

5.4	Automatic Generation of coreDb points .....	5-2
5.5	Basic Operations with coreDb .....	5-2
5.5.1	Point Management .....	5-3
5.5.2	Search .....	5-4
5.5.3	Configuring Data Point Value .....	5-5
5.5.4	Source and Destination Allocations .....	5-6
5.6	Data Point Configuration .....	5-7
5.6.1	Quality Flags .....	5-7
5.6.2	Status Configuration .....	5-8
5.6.3	Analog Configuration .....	5-8
5.6.4	Command and Setpoint Configuration .....	5-9
5.6.5	Other Operations with coreDb .....	5-9
5.7	Sharing Database Information between RTUs .....	5-10
5.8	Configuring Redundant RTUs (Only Saitel DR and Saitel DP) .....	5-11
5.8.1	Control .....	5-12
5.8.2	Mode .....	5-12
5.8.3	Bus .....	5-13
5.8.4	Additional IPs .....	5-13
Chapter 6 -	Working with Saitel .....	6-1
6.1	Introduction .....	6-1
6.2	Creating the RTU into a WorkSpace .....	6-1
Chapter 7 -	Working with Easergy T300 .....	7-1
7.1	Introduction .....	7-1
7.2	Sending Configuration Files to the FRTU .....	7-1
7.3	Configuring Cilo .....	7-2
7.4	Configuring Lioc .....	7-2
Appendix A -	Expressions .....	1
	Introduction .....	1
	NOT ( <i>input</i> ) → return: <i>output</i> .....	2
	TEMPO ( <i>input</i> , <i>time</i> ) → return: <i>output</i> .....	3
	SPSTODPS ( <i>input</i> ) → return: <i>output</i> .....	5
	DPSTOSPS ( <i>input</i> ) → return: <i>output</i> .....	6
	OR ( <i>input_1</i> , <i>input_2</i> , ..., <i>input_N</i> ) → return: <i>output</i> .....	7
	AND ( <i>input_1</i> , <i>input_2</i> , ..., <i>input_N</i> ) → return: <i>output</i> .....	9
	MIN ( <i>numDays</i> , <i>input</i> ) → return: <i>output</i> .....	11
	MAX ( <i>numDays</i> , <i>input</i> ) → return: <i>output</i> .....	12
	SCALE ( <i>factor</i> , <i>offset</i> , <i>input</i> ) → return: <i>output</i> .....	13
	IF ( <i>cond</i> , <i>ifVar</i> ) → return: <i>output</i> .....	14
	IF ( <i>cond</i> , <i>ifVar</i> , <i>elseVar</i> ) → return: <i>output</i> .....	15
	PLUS OPERATION ( <i>input1</i> + <i>input2</i> ) → return: <i>output</i> .....	16
	MINUS OPERATION ( <i>input1</i> - <i>input2</i> ) → return: <i>output</i> .....	17
	MULTIPLICATION OPERATION ( <i>input1</i> * <i>input2</i> ) → return: <i>output</i> .....	18
	DIVISION OPERATION ( <i>input1</i> / <i>input2</i> ) → return: <i>output</i> .....	19



---

GREATER CONDITIONAL OPERATION ( <i>input1&gt;input2</i> ) → return: <i>output</i> .....	20
LESS CONDITIONAL OPERATION ( <i>input1&lt;input2</i> ) → return: <i>output</i> .....	21
EQUAL CONDITIONAL OPERATION ( <i>input1==input2</i> ) → return: <i>output</i> .....	22
COREDB POINT NAME.....	23
VARIABLES .....	23
CONSTANT VALUE .....	23

# Index of Figures

Figure 1-1 - BaseLine Software Platform architecture.....	1-1
Figure 1-2 - Interaction between coreDb and other applications.....	1-2
Figure 1-3 – Saitel DR local acquisition plugin interface.....	1-4
Figure 1-4 – coreDb interface.....	1-4
Figure 2-1 – Startup of the Easergy Builder installation.....	2-3
Figure 2-2 – Selecting Configuration plugins to be installed in Easergy Builder.....	2-3
Figure 2-3 – Uninstalling Easergy Builder.....	2-4
Figure 2-4 – Easergy Builder startup window.....	2-5
Figure 2-5 - Easergy Builder in Workspace mode.....	2-5
Figure 2-6 – Workspace stored in folders.....	2-6
Figure 2-7 – RTUs tree.....	2-7
Figure 2-8 – Information on the editing area for a Saitel DR RTU.....	2-8
Figure 2-9 – Information on the editing area for an Easergy T300.....	2-8
Figure 2-10 – Information on the editing area for a Configuration.....	2-8
Figure 2-11 – Configuration mode.....	2-9
Figure 2-12 – Information about configuration Plugins installed in Easergy Builder.....	2-11
Figure 3-1 – Easergy Builder in Workspace mode (RTU parameters).....	3-1
Figure 3-2 – Easergy Builder in Workspace mode (configuration parameters).....	3-1
Figure 3-3 – Adding a new RTU.....	3-2
Figure 3-4 – Configuration by default for a RTU.....	3-3
Figure 3-5 - Configuring RTU parameters.....	3-3
Figure 3-6 - Acquisition modules in a default configuration.....	3-4
Figure 3-7 – Confirmation for automatic addressing.....	3-4
Figure 3-8 – Adding one (or several) AB.....	3-4
Figure 3-9 - Configuring a Saitel DP RTU.....	3-5
Figure 3-10 - Acquisition modules in the default configuration.....	3-5
Figure 3-11 – Adding new I/O modules.....	3-5
Figure 3-12 – Profibus configuration.....	3-6
Figure 3-13 - Configuring an Easergy T300 RTU.....	3-6
Figure 3-14 – RTU connection parameters.....	3-7
Figure 3-15 – Example of a Saitel DR redundant configuration.....	3-7
Figure 3-16 – Unique ITB in a redundant configuration.....	3-8
Figure 3-17 – Connection parameters in a redundant configuration.....	3-8
Figure 3-18 – Sending information to a redundant RTU.....	3-8
Figure 3-19 – Configuring users.....	3-9
Figure 3-20 – New user.....	3-9
Figure 3-21 – Configuring networks with redundant CPUs.....	3-10
Figure 3-22 – Configuring a network interface.....	3-11
Figure 3-23 – Configuring a WIFI network interface.....	3-11
Figure 3-24 – Example of external subnets.....	3-12
Figure 3-25 – Configuring external subnets.....	3-12

Figure 3-26 – A new subnet .....	3-12
Figure 3-27 – Configuring an external network using the default IP and mask .....	3-13
Figure 3-28 – Environment variables.....	3-13
Figure 3-29 – Dialup modems configuration.....	3-14
Figure 3-30 – PPP configuration. ....	3-14
Figure 3-31 – Removing an RTU.....	3-15
<b>Figure 3-32 – Reboot an RTU. ....</b>	<b>3-15</b>
Figure 3-33 – Error rebooting an RTU.....	3-15
Figure 3-34 – New configuration's name.....	3-16
Figure 3-35 – RTU sysLog. ....	3-16
Figure 3-36 – RTU sysLog. ....	3-16
Figure 3-37 – RTU sysLog. ....	3-17
Figure 3-38 – New configuration .....	3-17
Figure 3-39 – New configuration in the RTU tree .....	3-18
Figure 3-40 – Configuration mode.....	3-18
Figure 3-41 – Contextual menu for Configuration.....	3-18
Figure 4-1 – Easergy Builder configuration mode .....	4-1
Figure 4-2 – Selecting the type of the new Device .....	4-1
Figure 4-3 – Information about plugins that have been installed in Easergy Builder. ....	4-2
Figure 4-4 – Supervision points for HU 250 .....	4-2
Figure 4-5 – Available supervision points for SM_CPU866e.....	4-3
Figure 4-6 – Using formulas in coreDb.....	4-10
Figure 4-7 – Example of triggered expressions and triggers for formula .....	4-11
Figure 4-8 – Configuring communication channels .....	4-12
Figure 4-9 – Channel configuration parameters .....	4-13
Figure 4-10 - TCP channel configuration.....	4-14
Figure 4-11 - UDP channel configuration. ....	4-14
Figure 4-12 - ASYNC channel configuration. ....	4-15
Figure 4-13 – New link. ....	4-16
Figure 4-14 – Synchronization configuration .....	4-17
Figure 4-15 – Configuration of synchronization devices.....	4-18
Figure 4-16 – Configuring the RTU as SNTP server .....	4-19
Figure 4-17 – Configuring the RTU as IRIG server .....	4-19
Figure 4-18 – Configuring the RTU as SNTP server .....	4-19
Figure 5-1 - coreDb administration .....	5-1
Figure 5-2 - coreDb main menu.....	5-1
Figure 5-3 - status table in coreDb. ....	5-2
Figure 5-4 – Configuring local acquisition. ....	5-2
Figure 5-5 – Contextual menu for a selected point.....	5-3
Figure 5-6 – Contextual menu for a new point. ....	5-3
Figure 5-7 – Contextual menu for a new point. ....	5-3
Figure 5-8 - Add point from script. ....	5-3
Figure 5-9 - Add point from textbox. ....	5-4
Figure 5-10 - Add point from separated text .....	5-4

Figure 5-11 – Transferring a double point in two single points to ISaGRAF .....	5-4
Figure 5-12 - Search bar of a coreDb table .....	5-4
Figure 5-13 - Selecting a supervision point as source .....	5-6
Figure 5-14 – Use of the field “Description” .....	5-9
Figure 5-15 – Using Excel® for data importing .....	5-10
Figure 5-16 - Data importing process with error reporting .....	5-10
Figure 5-17 – Configuring dbNet. ....	5-11
Figure 5-18 – Possible “Data Publishing Devices” in Saitel DP .....	5-11
Figure 5-19 – Configuring dbRED (Redundancy control) .....	5-12
Figure 6-1 – Initial environment .....	6-1
Figure 6-2 – A new RTU .....	6-1
Figure 6-3 – Including slave modules .....	6-1
Figure 6-4 – Users and interfaces. ....	6-2
Figure 6-5 – Configuring the connection for Easergy Builder .....	6-2
Figure 6-6 – Showing messages on the Log Console .....	6-2
Figure 6-7 – Creating a configuration .....	6-2
Figure 6-8 – Editing the new configuration .....	6-2
Figure 6-9 – Selecting supervision points to be included in coreDb .....	6-3
Figure 6-10 – Configuring local acquisition .....	6-3
Figure 6-11 – Points defined in coreDb .....	6-4
Figure 7-3 – Editing and sending a HU250 configuration .....	7-1
Figure 7-4 – Select data to be sent to the HU250 .....	7-2
Figure 7-5 – Configuring Cilo .....	7-2
Figure 7-6 – Configuring Lioc .....	7-3
Figure 7-7 – Configuring front leds using Lioc .....	7-3
Figure 7-8 – Leds on the front panel .....	7-4

---

# Index of Tables

---

Table 3-1 – User rights.....	3-10
Table 4-1 – Common supervision points for all Saitel CPUs.....	4-4
Table 4-2 – Saitel DR supervision points.....	4-6
Table 4-3 – Saitel DP supervision points.....	4-7
Table 4-4 – Easergy T300 supervision points.....	4-9
Table 5-1 – Quality flags in coreDb points.....	5-7

# Manual Contents

## I. Scope

This manual provides information about Easergy Builder, the configuration tool for Saitel and Easergy RTUs. It describes the installation and use of the tool, in terms of common operations for all RTUs that use Schneider Electric's Software BaseLine Platform. The particularities of use for each of these RTUs are described in the "Configuration and Startup" manual of each range.

## II. Arrangement

This manual is divided in different chapters. The following lines describe briefly the contents covered by each chapter.

### Chapter 1 – BaseLine Software Platform

Main elements and software architecture of BaseLine Software Platform.

### Chapter 2 – Introduction Easergy Builder

General description of the tool.

### Chapter 3 – Working with RTUs

Use of Easergy Builder in Workspace mode to configure several RTUs at the same time.

### Chapter 4 – Working with Configurations

Use of Easergy Builder in Configuration mode to define the configuration of an RTU.

### Chapter 5 – coreDb – Real Time Database

Description of coreDb. This chapter covers information about the structure, use and functionality of coreDb.

### Chapter 6 – Configuring a Saitel RTU

Use of Easergy Builder to configure a Saitel DR RTU. Configuration process for a Saitel DR RTU is similar to Saitel DP RTU, so that only one of them (DR) is included in this chapter.

### Chapter 7 – Configuring an Easergy T300 RTU (HU 250)

Use of Easergy Builder to configure an Easergy T300 RTU.

### Appendix A - Expressions

Expressions allowed to be used with Formula Device.

## III. Reference Manuals

Additional information to this document can be found in the following documents:

Manual's name	Document
Configuration & Startup of Saitel DR	TE-HG-0000-CYP-F800
Saitel DR Modules	TE-HG-0000-MOD-F800
Configuration & Startup of Saitel DP	TE-HG-0000-CYP-F700
Saitel DP Modules	TE-HG-0000-MOD-F700
T300 Quick Start	NT00383-EN
T300 User Manual	NT00378-EN
Easergy Builder - Configuring IEC101	TE-HG-0000-I1D-S854

Manual's name	Document
Easergy Builder - Configuring IEC104	TE-HG-0000-I4D-S854
Easergy Builder - Configuring IEC103	TE-HG-0000-I3D-S854
Easergy Builder - Configuring Modbus	TE-HG-0000-MBD-S854
Easergy Builder - Configuring ISaGRAF	TE-HG-0000-ISD-S854
Easergy Builder - Configuring DNP	TE-HG-0000-DNP-S854
Easergy Builder - Configuring SOE	TE-HG-0000-SOE-S854

Table 1 - Reference manuals.

## IV. Software Versions

The information included in this document is valid for the versions of the software modules listed below and subsequent:

Software	Version	Setup file
Easergy Builder	01.00.20	Easergy Builder.msi
Cilo (Cilo)	1.0.3.0	Cilo_SPlugin.msi
DNP master (DNPM)	1.0.6.0	DNP_Master_SPlugin.msi
DNP slave (DNPS)	1.0.9.0	DNP_Slave_SPlugin.msi
IEC101 master (IEC101M)	1.0.9.0	IEC101_Master_SPlugin.msi
IEC101 slave (IEC101S)	1.0.7.0	IEC101_Slave_SPlugin.msi
IEC103 master (IEC103M)	1.0.9.0	IEC103_Master_SPlugin.msi
IEC104 master (IEC104M)	1.0.6.0	IEC104_Master_SPlugin.msi
IEC104 slave (IEC104S)	1.0.10.0	IEC104_Slave_SPlugin.msi
IEC61850 (IEC61850)	1.0.23.0	IEC61850_SPlugin.msi
ISaGRAF (ISAGRAF)	1.0.2.0	IsaGRAF_SPlugin.msi
ISaGRAF 5 (ISAGRAF5)	1.0.7.0	IsaGRAF5_SPlugin.msi
Lioc (Lioc)	1.0.4.0	Lioc_SPlugin.msi
Modbus master (MDBM)	1.0.8.0	ModBus_Master_SPlugin.msi
Modbus slave (MDBS)	1.0.6.0	ModBus_Slave_SPlugin.msi
SOE	1.0.9.0	SOE_SPlugin.msi
Saitel DP acquisition (LAQ)	1.0.8.0	LAQ_SPlugin.msi
Saitel DR acquisition (CLAQ)	1.0.4.0	CLAQ_SPlugin.msi

Table 2 - Software versions covered by this manual.

## V. Devices Compatibility

Software features supported by each CPU of the different hardware platforms:

Devices	SM_CPU866	SM_CPU866e	HU_B/HU_BI	HU_A/HU_AF	HU 250
Command Management - Cilo	✗	✗	✗	✗	✓
DNP Master Protocol	✓	✓	✗	✓	✓
DNP Slave Protocol	✓	✓	✗	✓	✓
Formula	✗	✗	✗	✗	✓
IEC101 Master Protocol	✓	✓	✗	✓	✗

Devices	SM_CPU866	SM_CPU866e	HU_B/HU_BI	HU_A/HU_AF	HU 250
IEC101 Slave Protocol	✓	✓	✓	✓	✓
IEC103 Master Protocol	✓	✓	✗	✓	✗
IEC104 Master Protocol	✓	✓	✗	✓	✓
IEC104 Slave Protocol	✓	✓	✓	✓	✓
IEC61850 Plugin	✓	✓	✗	✓	✗
ISaGRAF	✓	✓	✗	✓	✗
ISaGRAF5	✓	✓	✗	✓	✓
MICOM Master Protocol	✓	✓	✗	✓	✗
Modbus Master Protocol	✓	✓	✓	✓	✓
Modbus Slave Protocol	✓	✓	✓	✓	✗
Local Input Output - Lioc	✗	✗	✗	✗	✓
Saitel DP local acquisition (laq)	✓	✓	✗	✗	✗
Saitel DR local acquisition (claq)	✗	✗	✓	✓	✗
Sepam Protocol	✓	✓	✗	✓	✗
Sequence of events	✓	✓	✗	✓	✓
Supervision	✓	✓	✓	✓	✓

Table 3 - Available devices for each CPU



# Chapter 1 - BaseLine Software Platform

## 1.1 Software Architecture

The BaseLine Software Platform of Schneider Electric consists of:

- Real-time operating system (RTOS): VxWorks, Linux,...
- Real-time applications and configuration files.
- Configuration, management, supervision and monitorization tools.

### NOTICE

Saitel DR's basic head units HU\_B and HU\_BF modules do not run VxWorks nor Linux. They operate with a tailored-made OS which includes the database and applications.

The following figure shows the different applications included in the software platform, as well as additional applications that implement new devices or protocols to upgrade Easergy Builder:

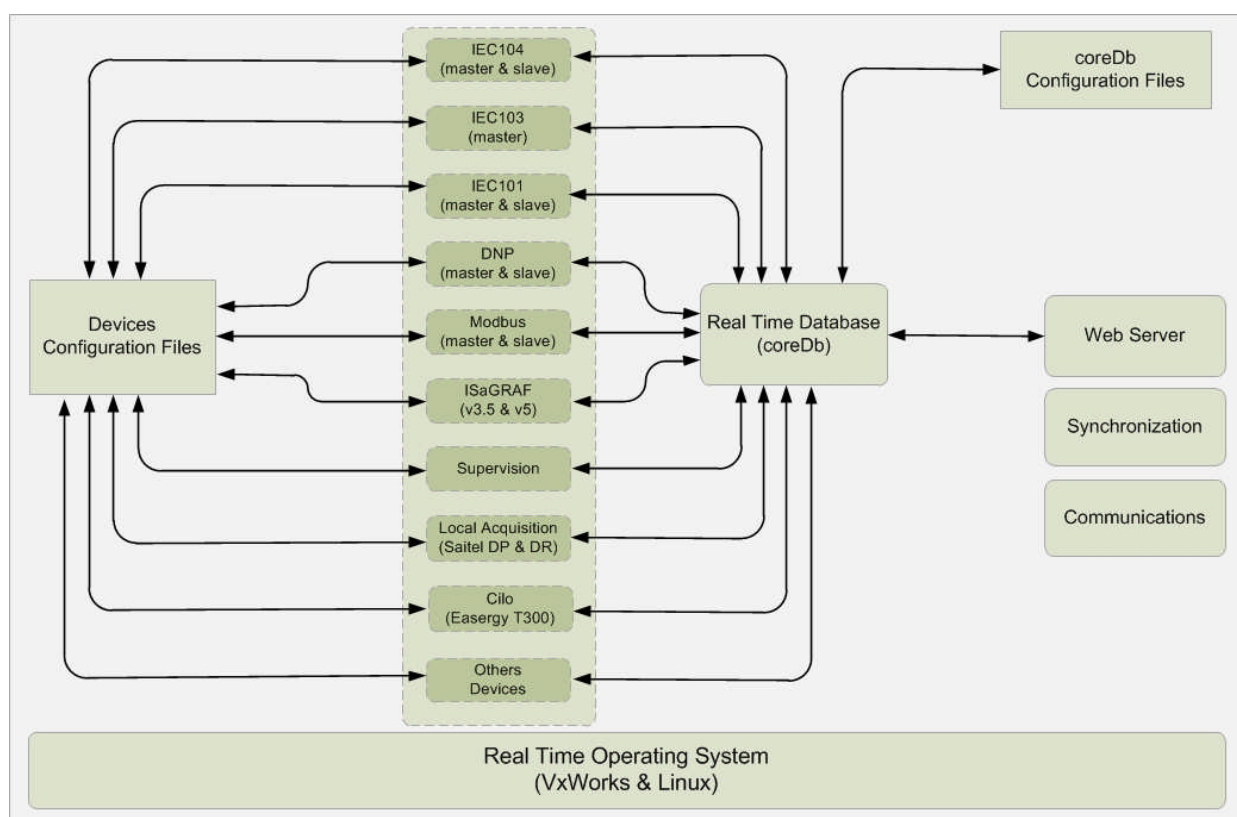


Figure 1-1 - BaseLine Software Platform architecture.

The operating system abstracts the hardware from the software applications and manages the applications in real time. It integrates the basic protocols to access the remote unit (SFTP, SSH, etc.) and manage multiple users.

The real-time database, named coreDb, is probably the most important element. All the other elements are developed around coreDb:

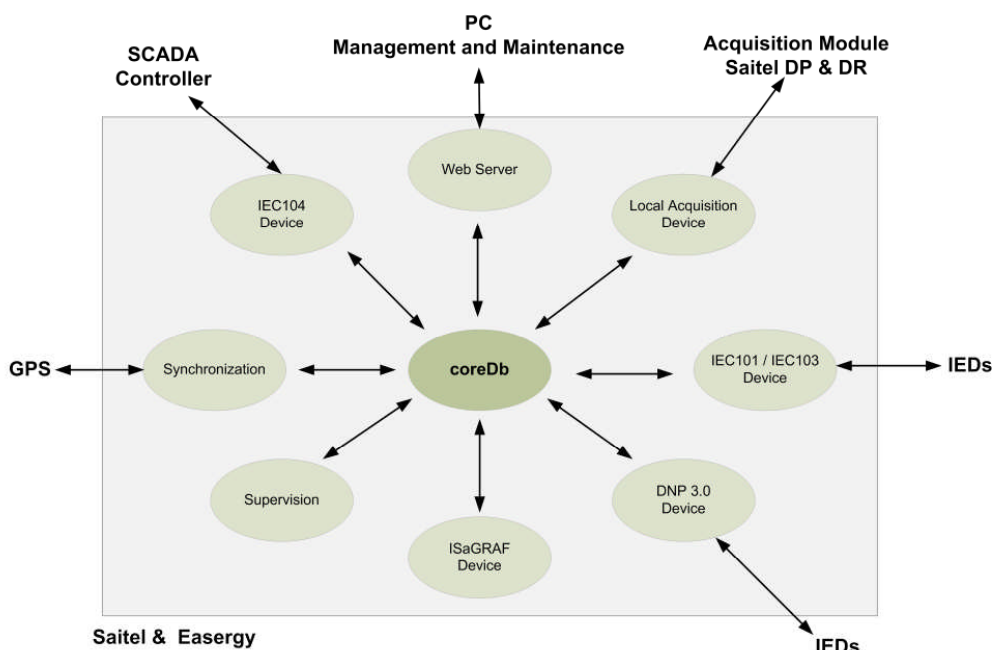


Figure 1-2 - Interaction between coreDb and other applications.

coreDb performs the real-time management of RTU points. This real-time database is associated with data producing and consuming by device controllers.

### NOTICE

The real-time database of the HU\_B and HU\_BF modules (Saitel DR) is named coreDbLite and offers the same functionalities and interfaces as coreDb. All the information included in this manual about coreDb is also applicable to coreDbLite, unless explicitly stated otherwise.

The following concepts are related to coreDb:

- **Device Controller (also referred to as Controller):** Real-time application that accesses coreDb. Each Controller acts as a producer and/or consumer of information managed by coreDb.
- **Point:** Each register of coreDb is a point. A point can be included in the table Status, Analog, Command or Setpoint
- **Device:** A set of I/O points that share a common source/destination. A typical example of a Device is an IED that communicates with the RTU, or the representation of a SCADA exchanging information acquired or generated by the RTU. A Device is always associated to a type of Controller.
- **Source:** Origin of the value of a coreDb data point. Any coreDb data point can have several different sources (in one or several Devices). This means that a value of a database point can be configured to be updated by several different entities.
- **Destination:** Target of the value of a coreDb data point. coreDb data points can be configured to have several different destinations (in one or several Devices).

### NOTICE

It should be noted that although all coreDb data points can be associated to several sources; this isn't recommended for points in the Status and Analog tables.

- **Coordinate:** Point identification within a Device. It is unique for each point and has a different structure for each Controller. It is described in detail in the appropriate manual of each Controller (see Table 1).
- **Configuration Plugin:** Specific Configuration plugins extend the Easergy Builder application to configure Device Controllers. Additional details about these plugins are provided further in this manual.

Each Device communicates with the associated Controller through its specific protocol. The Controller receives the information, processes it following the specified set of rules and, finally, sends it to coreDb. Similarly, a Controller can read information from coreDb and send it to the appropriate Device.

The user can modify the configuration of each Controller and Device using the appropriate Plugin. Once the database is completely configured, the files with the new information can be generated and transferred to the RTU, where they will be processed by the software on startup.

## NOTICE

The information exchange, that is, the exchange of configuration data between the RTU and Easergy Builder is not continuous, but performed through XML files under user's request, sooner the configuration is modified in Easergy Builder and the XML files are sent to the RTU, it is necessary to reboot the RTU.

## 1.2 Main Components

The BaseLine Software Platform's main elements are:

- coreDb.
- Devices.
- User interface.

### coreDb – Real Time Database (RTDB)

coreDb is the real-time database backend on which BaseLine Software Platform is built. All the information controlled and managed by the system is stored in this database.

Thanks to this architecture, the system's functionalities can be easily expanded to manage new protocols, customized controllers, etc. To accomplish this, trained developers only need to implement the required Device Controller and the associated Configuration plugin for Easergy Builder end users to configure the extended functionality.

coreDb registers also are called data points or, simply, points (this term will be used onwards to avoid confusing these with Device points). coreDb points are organized in four tables: Status, Analog, SetPoint and Command to group the different types of point. These internal tables present the following differences:

- **Depending on the point type:** Status and Command points support integer values, whereas Analog and Setpoints points manage floating values.
- **Depending on the treatment of the point:** Status, setpoint and analog points can be locked, reset to initial values, whereas command points cannot. All types can retain the value in a non-volatile memory.

### Devices

Each type of device keeps a list of its associated points, identified by unique labels. These labels allow the identification of each device point unequivocally as source or destination of a coreDb data point.

Each point is a piece of information produced (or consumed) by a Device. Within a single Device, point identifiers (coordinates) are unique and cannot be used by two different points.

Easergy Builder supports the following device configuration plugins:

- ISAGRAF (version 3 of ISaGRAF) and ISAGRAF5 (version 5).
- MODBUS master and slave (with several profiles).
- IEC101 master and slave.
- IEC104 master and slave.
- IEC103 master
- DNP master and slave.
- SOE (Sequence of Events management).
- Saitel DP and Saitel DR local acquisition.
- Formula
- T300 control & configuration (Cilo and Lioc)

Each Device will be available or not depending on the type of CPU to configure. More information about available Devices for each CPU in Table 1.

### User Interface

Easergy Builder has two types of user screens to consult and modify the information stored in coreDb:

- **Graphical Interface (Plugin):** The picture below shows the interface to configure Saitel DR acquisition hardware. Using this graphical interface the user can configure each local acquisition points with its corresponding point in coreDb. There is a graphical interface for each Device Controller.



---

# Chapter 2 - Introduction to Easergy Builder

---


## 2.1 Introduction

Easergy Builder is the configuration tool for Schneider Electric RTUs that use the BaseLine Software Platform. Among other features, it can be used to perform:

- Configuration of the general settings of an RTU (IP address, user administration, communication channels, etc.).
- Design and maintenance of coreDb.
- Administration of the synchronization mechanisms.
- Configuration of the supervision and monitoring features.

Easergy Builder allows managing several RTUs from a single interface. For each type of RTU, several profiles or configurations can be defined and enables the user to choose which active profile to configure.

Easergy Builder has two different working modes:

- **Workspace mode:** Used to administrate RTUs and their associated profiles and configurations. This mode is presented to the user on startup.
- **Configuration mode:** Engineering interface for a specific configuration of an RTU. This mode is activated by double-clicking on an RTU configuration of the tree in the workspace mode or selecting the configuration and clicking button .

## 2.2 Easergy Builder Installation

The next step is executing Easergy Builder installation program, named “Easergy Builder.msi”.

Easergy Builder must be completely installed in the PC before executing it. To install it, the user must follow the steps provided in this chapter.

### 2.2.1 Easergy Builder Requirements

#### Setup File

---

To install this tool on Windows XP is necessary the Easergy Builder.msi file. On Windows 7 is also necessary the setup.exe file and run it as administrator.

#### .NET Framework

---

Before installing the tool, version 2.0 or later of the “.NET” Framework package by Microsoft® must be installed in the PC. Current versions of Windows® operating systems already have .NET Framework 2.0 or later installed.

If this package is not available in the PC, a dialog box will open up informing the user during Easergy Builder installation.

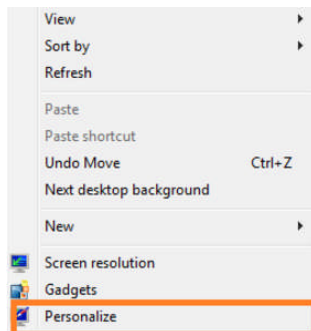
#### DPI on Windows 7

---

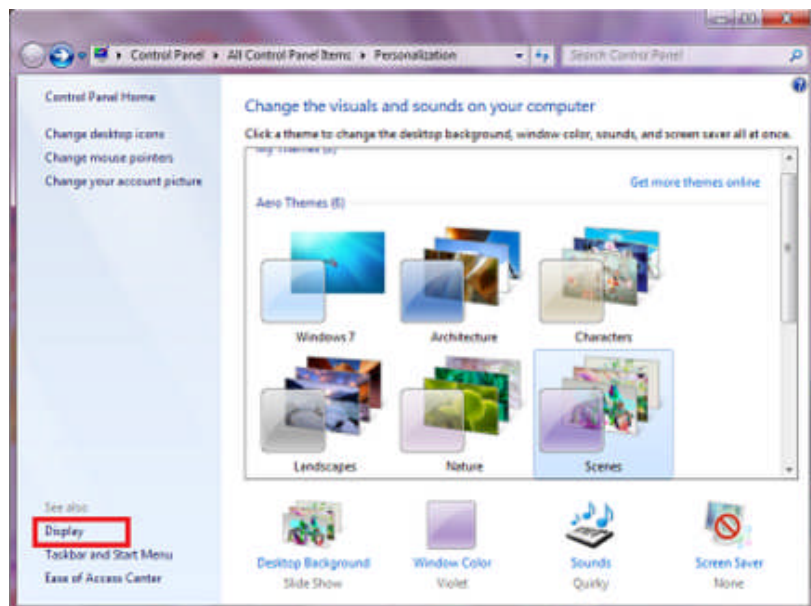
When you execute Easergy Builder, the DPI parameter on Windows 7 must be 96 (100% of the screen resolution).

If not, please follow these instructions:

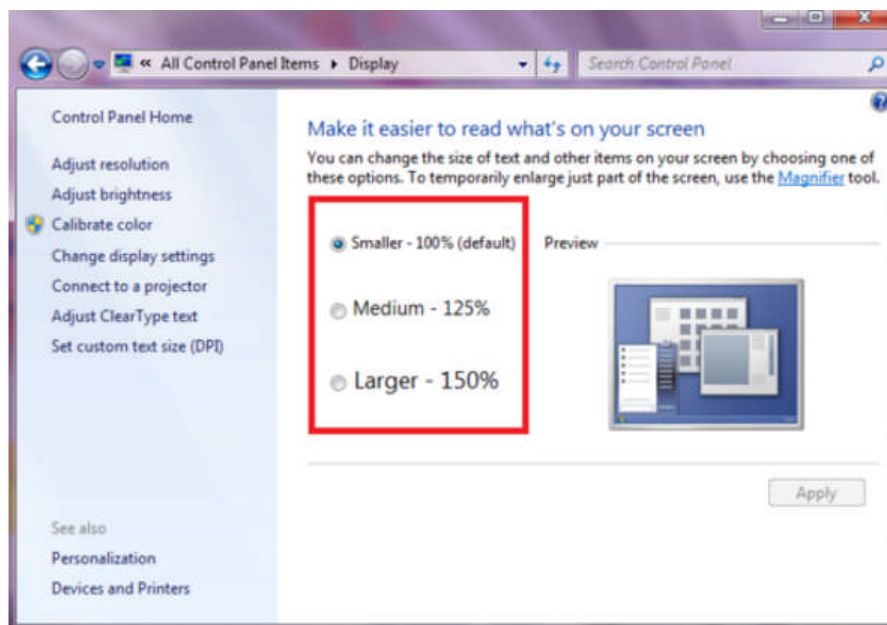
1. Right click on a empty area of your desktop and choose Personalize:



2. Click on the Display link at the bottom left corner:



3. In the follow screen, please select “Smaller – 100%”. This configuration is according to 96 DPI necessary for Easergy Builder execution.



4. Click on the Apply button and select “Log off now” when you are asked. Please, be sure to save anything you are working on before you click on this.

## 2.2.2 Easergy Builder and Plugins

The next step is executing Easergy Builder installation program, named “Easergy Builder.msi” or “setup.exe” depending on the OS.

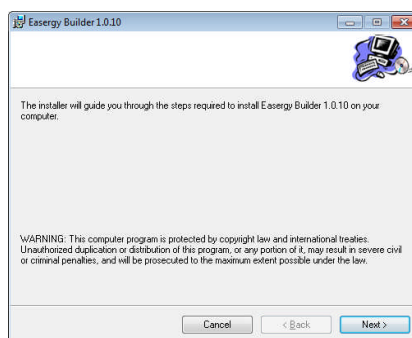


Figure 2-1 – Startup of the Easergy Builder installation.

Follow the steps appearing on the screen to complete the software installation; during the process the user will be prompted to select the configuration plugins to install:

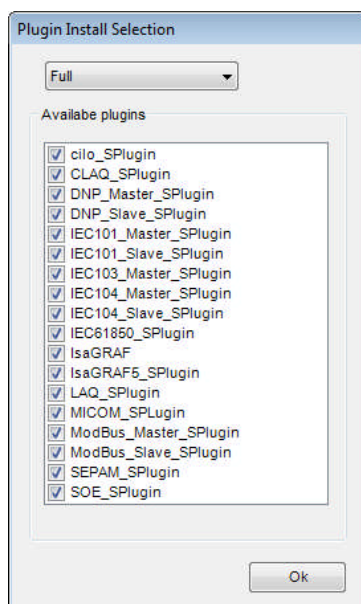
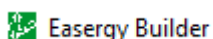


Figure 2-2 – Selecting Configuration plugins to be installed in Easergy Builder.

This screen will show the plugin installation files located in the same directory as the Easergy Builder installer. Each msi file contains a configuration plugin. You need to install a plugin or not depending on the type of CPU. More information about available plugins for each CPU in Table 2 of this manual.

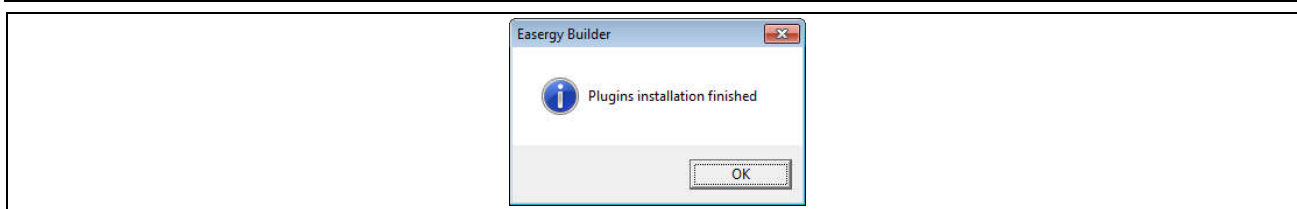
Select the plugins to be installed. Press “Ok” button and the installation process will complete. Once finished, the Windows startup menu will show the Easergy Builder icon under the “Schneider Electric” folder.



### WARNING

The installation of the Plugins is executed in background. Please, be sure the following window is shown before running the application:

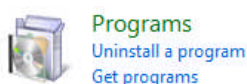




## 2.2.3 Easergy Builder Update and Uninstall

The uninstall process of the tool depends on the host operating system in which Easergy Builder is running. This manual assumes that the user is working with Windows 7 Professional, so the process might differ considerably if the user is using a different operating system.

In order to update or uninstall Easergy Builder or a Configuration plugin on Windows 7 Professional, the user must access the control panel: **"Start → Control Panel"**, and select **"Uninstall a program"** in section **"Programs"**.



This utility allows updating or uninstalling one or several Plugins and also it allows updating or uninstalling Easergy Builder completely from the host PC.

### Uninstall a Plugin

To remove a Plugins from the PC, on the control panel, access to the software installed and select the plugin to be removed. Double-click on it and the uninstall process start.

If the user selects "No" when prompted to confirm the operation, the uninstall process is cancelled. If confirmed, the process will complete with the elimination of the selected plugin.

### Uninstall Easergy Builder

To remove Easergy Builder completely from the PC, it must be selected in the control panel's program list and double-click on it.

The uninstall wizard is executed. Depending on Easergy Builder Version, the user could be prompted to select the plugins that must be removed from the PC. In order to remove Easergy Builder completely, select "Full" in the plugin selection window. All plugins will be selected to uninstall them:

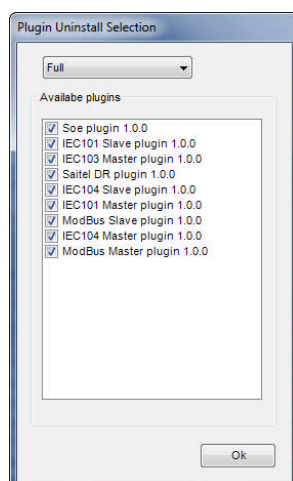


Figure 2-3 – Uninstalling Easergy Builder.

## NOTICE

You should note that the default uninstall process is set to "Minimal". This option uninstalls "Easergy Builder" but all plugins that haven't been selected will remain installed.

Press "Ok" and the uninstall process will complete.



## 2.3 Environment Description

Easergy Builder can be launched from the Windows® Start menu, or through the desktop shortcut created during the installation.



Figure 2-4 – Easergy Builder startup window.

In Easergy Builder there are two working modes:

- Workspace
- Configuration

The workspace mode is shown when Easergy Builder starts.

### 2.3.1 Workspace Mode

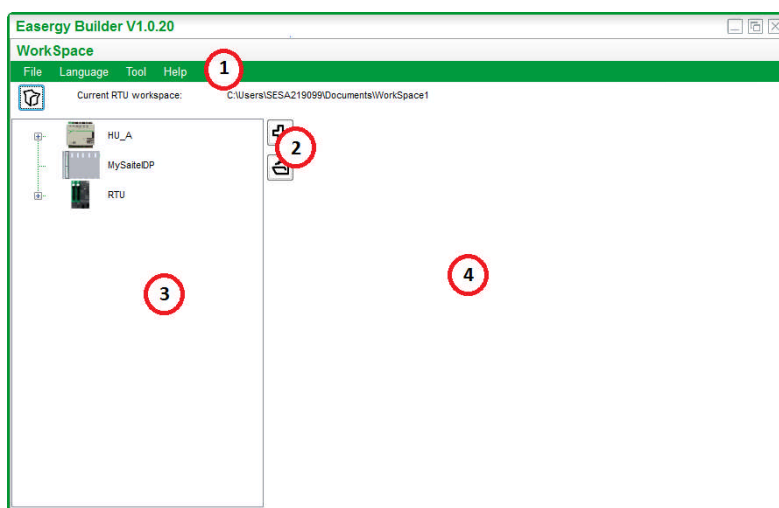


Figure 2-5 - Easergy Builder in Workspace mode.

Easergy Builder's workspace mode has four areas:

- 1: Main menu.
- 2: Administration toolbar.
- 3: RTUs area.
- 4: Editing area

When Easergy Builder is in workspace mode, the user can:

- Create new Workspaces or load other Workspaces.
- Create and configure several RTUs at the same time.
- Select the preferred language (Deutsch, English and Spanish are available by default).
- Consult information about Easergy Builder (Help).

---

## Zone 1. Main Menu

---

The main menu has the following options:

- **File:** This option allows creating and loading workspaces. A Workspace is a set of RTUs stored in the same computer folder. All the RTUs defined in a workspace are shown in the RTU tree. The Workspace path is shown under the main menu identified by the text “**Current RTU workspace**”.

You can reload the actual Workspace using button .

For example, this folder for the Workspace shown previously is stored as follow:

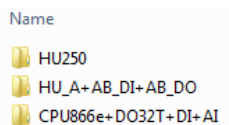


Figure 2-6 – Workspace stored in folders.




- **New WorkSpace:** Create a new workspace.
- **Load WorkSpace:** Upload all RTUs defined in a workspace.
- **Exit:** Exit Easergy Builder.
- **Language:** Language selection. For now, Deutsch, English and Spanish translations are available.
- **Tool:** Configuration of Easergy Builder's general parameters.
- **Help:** Access to Easergy Builder's embedded help.

---

## Zone 2. Administration Toolbar










---

Buttons shown in this toolbar depend on the element selected. On start-up, available buttons in the administration toolbar are:

-  Add RTU. Create a new RTU inside the current workspace.
-  Import an RTU from an EBC file. This file should have been generated previously with Easergy Builder, using button .







These are the only available buttons when a workspace is opened for the first time. If there aren't defined RTUs, no other operations can be done.

When an RTU is selected, the following additional buttons will be shown:

-  “**Remove RTU**”. Remove the selected RTU from the Workspace. All its configurations will be removed also.
-  “**Reboot RTU**”. Reboot the selected RTU
-  “**Create Configuration**”. Create a new configuration for the selected RTU.
-  “**Read Configuration**”. Transfers all the configuration files from the selected RTU to the Workspace. Users, IP addresses and other configuration parameters will be loaded to the RTU profile.
-  “**Completely Configure RTU**”. Send the complete configuration to the RTU (interfaces, coreDb, ...).
-  “**Save the selected RTU**”. Save all configurations for this RTU in a file in order to import into other WorkSpace using button . The file type is depending on the type of CPU to be set.
-  “**Load a Configuration**”. Using this button, you can load a configuration, RTU or a configuration project generated by CATconfig Tool. This option allows using Easergy Builder with legacy configuration projects.
-  “**Syslog**”. Load the syslog file from the RTU to the PC.

---

If a Configuration is selected, the following buttons are available:

-  **“Configure”**. Switch to configuration mode to edit the selected RTU Configuration.
-  **“Send Configuration to RTU”**. Transfer the selected configuration to the RTU.
-  **“Remove Configuration”**. Remove the selected configuration from the RTU.
-  **“Save the selected configuration”**. Save this configuration in order to import into other WorkSpace using button . The configuration can be saved as a RTU configuration file. This configuration will contain only the application part and not the system part (network, modems ...).
-  **“Clone Configuration”**. Copy the selected configuration with a different name in the same RTU. This is useful for versioning.

### Zone 3. RTUs Area

---

In this area, all RTUs defined in the Workspace are shown.

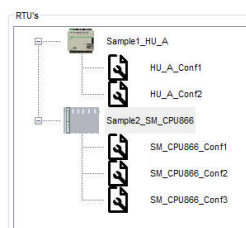














Figure 2-7 – RTUs tree.

Right-clicking on an RTU, the following actions are available:

- Remove RTU ().
- Reboot RTU ().
- Create Configuration ().
- Read Configuration ().
- Completely configure RTU ().
- Save the selected RTU ().
- Load a configuration ().

Right-clicking on a Configuration, the following actions are available:

- Configure ().
- Send Configuration to RTU ().
- Remove Configuration ().
- Save the selected Configuration ().
- Clone Configuration ().

### Zone 4. Editing Area

---

The content of this zone depends on the element selected (RTU or Configuration) and the type of this element (Saitel DR, Saitel DP or T300).

For example, if a Saitel DR-HU\_A is selected in the RTU tree, the following information will be shown in the editing area:

RTU Type: Saitel DR-HU\_A RTU Description: Examples

Default configuration of new Configurations

AB\_SER 1

RTU acquisition

☐ RTU Redundancy

Connection

IP: Transfer Timeout: 50000 ms

Users Network Environment variables

User	Privileges

Figure 2-8 – Information on the editing area for a Saitel DR RTU.

If the RTU selected is an Easergy T300 the editing area shows the following information:

RTU Type: Easergy T300 RTU Description:

Default configuration of new Configurations

Slot 1 2

RTU acquisition

☐ RTU Redundancy

Connection

IP: Transfer Timeout: 50000 ms SSH port: 22

Network Environment variables Dialup Modems PPP

Name	IP Address	Subnet Mask	Type	Channel	SSID	Subr

Figure 2-9 – Information on the editing area for an Easergy T300.

If a Configuration is selected, for example for a T300 RTU, the following information will be shown in the editing area:

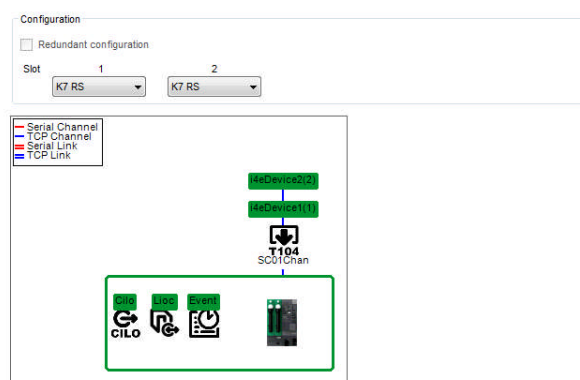



Figure 2-10 – Information on the editing area for a Configuration.

## 2.3.2 Configuration Mode

Easergy Builder switches from Workspace to Configuration mode when the user double-clicks over a Configuration or selects a configuration and presses button .

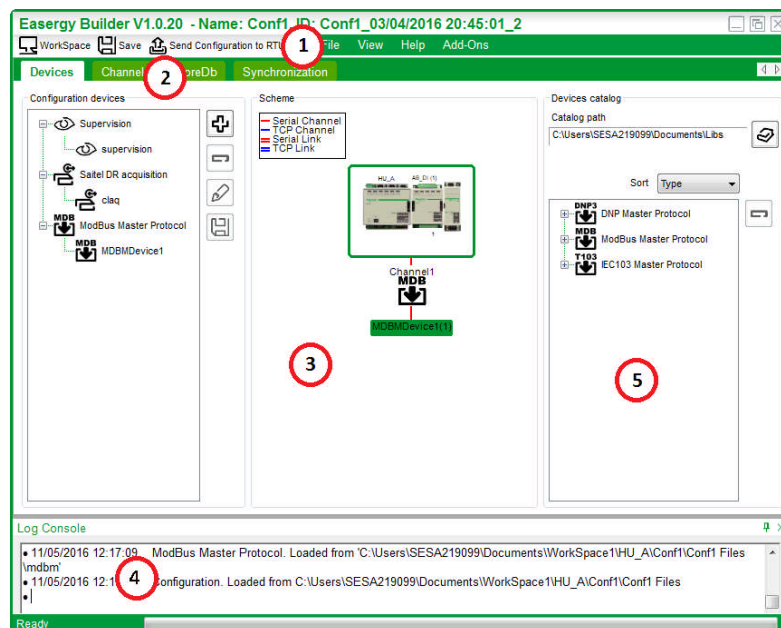


Figure 2-11 – Configuration mode.

The Configuration mode has the following areas:

- 1: Main menu.
- 2: Section menu: Type of information to be shown on the editing zone. Devices, Channels, coreDb and Synchronization are available.
- 3: Editing zone.
- 4: Log view.
- 5: Device catalog

Operations available in configuration mode are:

- Device configuration.
- Channel configuration.
- Editing of coreDb content.
- Synchronization configuration.

The aspect of the editing zone depends on the type of information selected in zone 2.

### Zone 1. Main Menu

The main menu has the following options:

- **WorkSpace:** Switch back to Workspace mode.
- **Save:** Save the information of the open configuration to the PC.
- **Send Configuration to the RTU:** Send the open configuration to the associated RTU – this association is selected previously in workspace mode.
- **File:** Exit is the only available item:
  - **Exit:** Exit Easergy Builder.
- **View:** Allows to show “Log Console” or “sysLog Console” on the bottom of the Easergy Builder window.
- **Help:** General information about Easergy Builder and installed Plugins.

- **Add-Ons:** An Add-on is an external application, implemented as a dynamic library (DLL). This option shows all installed add-ons on the PC.

## Zone 2. Section Menu

The section menu gives access to information about:

- **Devices:** A device is a set of information exchanged between coreDb and a generator/consumer of information.
- **Channels:** RTU communication channels.
- **coreDb:** Real-time database of the RTU.
- **Synchronization:** Definition and configuration of synchronization devices.

## Zone 3. Editing Area

The content of this zone depends on the tab selected in the Section Menu. More information below in this manual.

## Zone 4. Log View

The information shown in this zone depends on the option selected in the main menu: “**View → Log Console**” or “**View → sysLog**”.

- **Log Console:** Shows information about the operations performed by Easergy Builder.
- **sysLog Console:** Show the last log file that was read from the RTU.

### NOTICE

It is important to note that the RTU log messages shown here are not refreshed automatically. The user must request the update of the information using button “sysLog” (📄) in Workspace mode.

## Zone 5. Device Catalog

The Device catalog is a library of configurations that were stored as templates. The user can create and store in the PC a set of configuration templates in order to re-use them in several RTUs and configurations.

You can create a new template:

- By right-clicking on a Device from the Device tree and select “Create template”.
- Using button

The tool will be prompt the user for a template name and then proceed to generate a new template with the same structure and settings as those used in the open Configuration. Generated templates will be shown in the Device catalog.

# 2.4 Getting Started with Easergy Builder

## Devices

The data exchange with coreDb is made by the Device Controllers. Each type of controller defines the rules that must be followed in this data exchange.

Each Device controller is associated with an Easergy Builder "Configuration Plugin". Each plugin implements the rules to follow and offer user-friendly graphical interfaces to set the data exchange with a Device (through its Device Controller). All plugins can be installed and uninstalled independently from the configuration tool core.

## Channels

The ports used to communicate with field devices are configured as communication channels. The number and type of these channels depends on the type of RTU and the communication modules installed (AB\_SER, SM\_SER or K7 modules). A general overview on communication channels is shown in this manual.

## Plugin Information

In the configuration zone, from the main menu, select “**Help → About**” to obtain information about the Configuration Plugins installed with Easergy Builder:

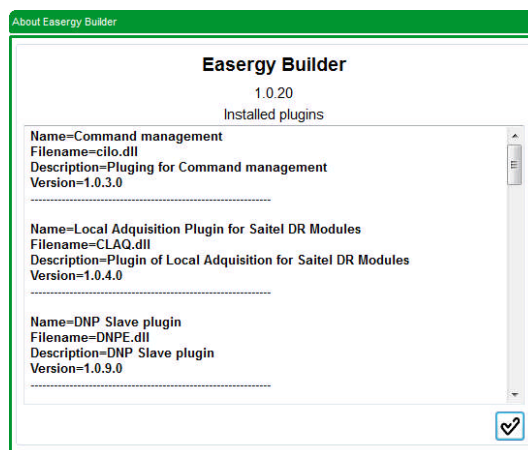


Figure 2-12 – Information about configuration Plugins installed in Easergy Builder.

For each Plugin, following information is shown:

- **Name:** Plugin name.
- **Filename:** File name where it is implemented.
- **Description:** Plugin description.
- **Version:** Last version installed.

## Add-Ons

An Add-on is an external application, implemented as a dynamic library (DLL). Add-ons can be integrated into Easergy Builder simply by placing the DLL file in a specific directory – **C:\Program Files\Schneider Electric\Easergy Builder\PlugsAddOn** - and restarting the application.

### NOTICE

If the user modified the default directory during the installation of the tool, the path for Add-Ons should also change.

As an example of the use of this feature, let's assume that an Add-on providing a calculator has been implemented and there is a DLL file containing the Add-on.

The user needs to create a new directory named, for example, "Calculator" within the "**PlugsAddon**" folder under the main path where Easergy Builder software is installed, and copy the DLL file to this folder.

### WARNING

Do not copy this DLL file in the "PlugsAddon" directory, but in a folder under this directory, labeled as the Add-on name to be installed.

To activate the Add-on, restart Easergy Builder and the calculator will be available from the main menu "**Add-Ons → Calculator**".

# Chapter 3 - Working with RTUs

## 3.1 Introduction

With the tool in workspace mode, the user can manage several RTUs from a single interface. Different configurations can be assigned to each RTU, and the user can switch from one to another.

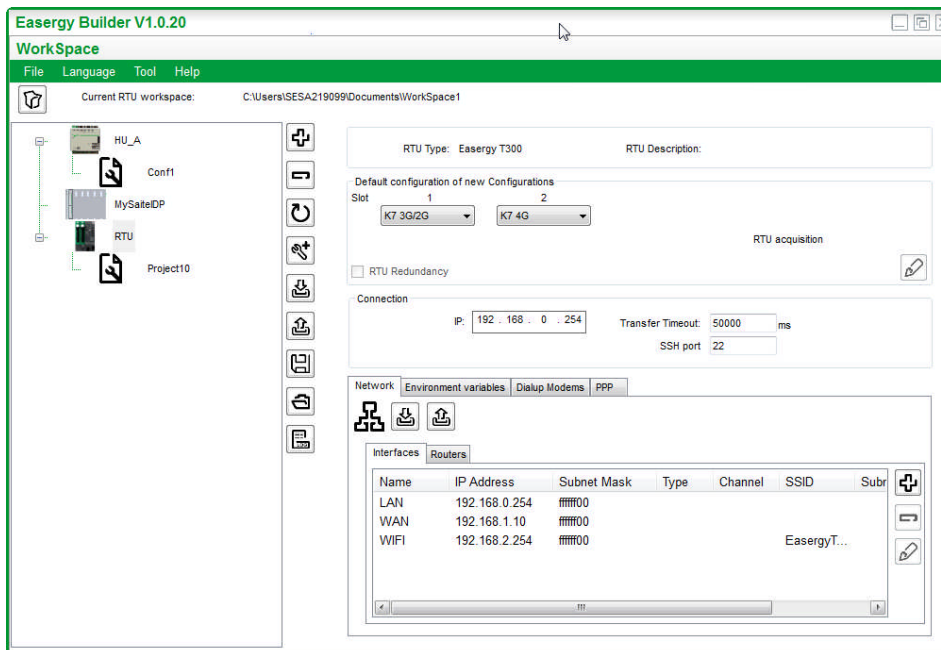


Figure 3-1 – Easergy Builder in Workspace mode (RTU parameters).

In the previous image, there are two RTUs defined. Both, Saitel DR (HU\_A) and Easergy T300 (RTU) have one configuration assigned.

Depending on the element selected in the RTU tree, the editing zone will show the RTU information (previous picture) or the configuration information (a communications schematic diagram).

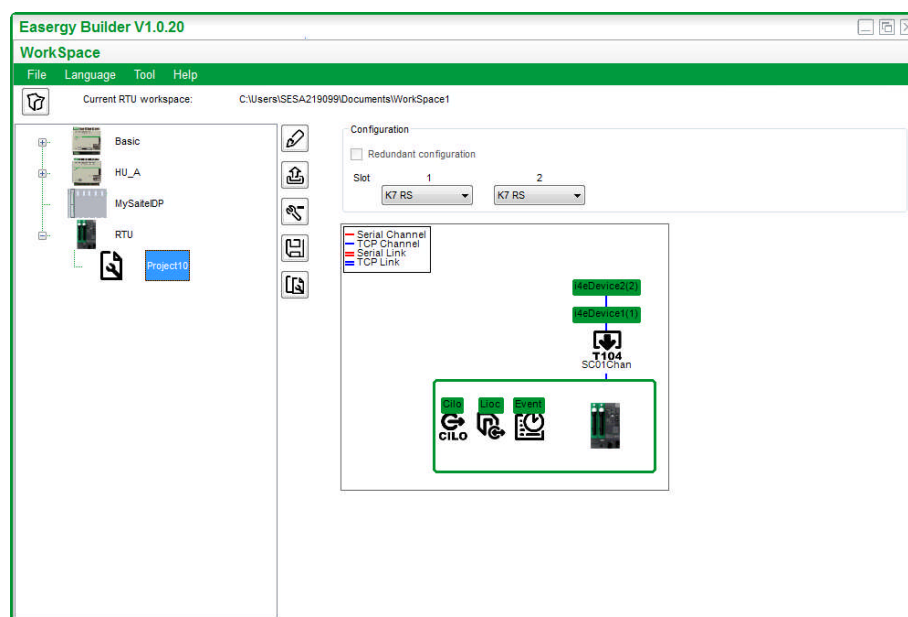


Figure 3-2 – Easergy Builder in Workspace mode (configuration parameters).



## 3.2 RTU Administration

Working with RTUs in the Workspace mode, you can:

- Create a new RTU (🔼).
- Remove RTU (🗑).
- Reboot RTU (🔄).
- Create a new configuration (🔧).
- Read a configuration (📄).
- Send the complete configuration to the RTU (📤).
- Save the selected RTU (💾).
- Load a configuration (📁).
- Read SysLog from the RTU (📖).

### 3.2.1 Adding an RTU

Press button 🔼 or right-click on the RTU tree to add a new RTU by entering the required information in following fields:

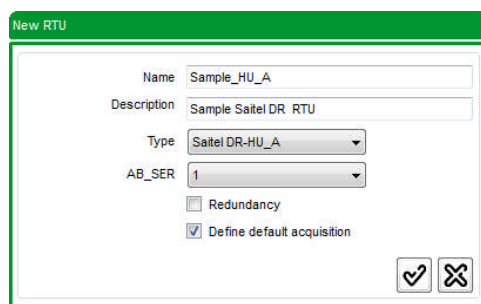


Figure 3-3 – Adding a new RTU.

Where:

- **Name:** RTU name. The length can't exceed 64 characters and can't contain the characters \, /, :, \*, ?, ", <, > or |. In the RTU tree, a new RTU will be identified with this name.
- **Description:** RTU description, 128 characters maximum (optional).
- **Type:** Depending on the CPU, Easergy Builder supports the following types of RTU:
  - Easergy T300: T300 using the HU 250 as CPU.
  - Saitel DP-SM\_CPU866: Saitel DP RTU using the SM\_CPU866 module as CPU.
  - Saitel DP-SM\_CPU866e: Saitel DP RTU using the SM\_CPU866e module as CPU.
  - Saitel DR-HU\_A: Saitel DR RTU using the HU\_A module as CPU.
  - Saitel DR-HU\_AF: Saitel DR RTU using the HU\_A module as CPU.
  - Saitel DR-HU\_B: Saitel DR RTU using the HU\_B module as CPU.
  - Saitel DR-HU\_BI: Saitel DR RTU using the HU\_BF module as CPU.
- **SM\_SER, AB\_SER, Slot 1 / Slot 2:** Number of communication modules available in the RTU. These modules won't be shown in Easergy Builder but their ports will be available to be used as additional communication channels. This field depends on CPU types;
  - SM\_SER is available for SM\_CPU866 and SM\_CPU866e
  - AB\_SER is available for HU\_A and HU\_AF
  - Slot 1 and Slot 2 is available for HU 250. In this fields you can select K7 RS, K7 3G/2G or K7 4G.
- **Redundancy:** Check this field if the RTU is CPU-redundant. Not available for HU\_B, HU\_BI or HU 250.
- **Define default acquisition:** When the CPU is created, you will be prompted to select the acquisition modules installed in the RTU. These modules will be added by default when you create a configuration for this RTU.

All these values will be used by default when you create a new configuration for this RTU.

For example, if you check “Define default acquisition”, all of the RTU's acquisition modules will be added too into each new configuration created for this RTU. In this case, you will be prompted to select the acquisition modules to be included in the default configuration:

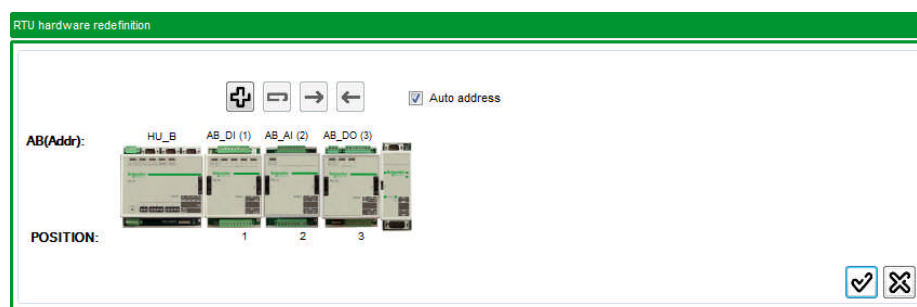


Figure 3-4 – Configuration by default for a RTU.

## 3.2.2 Configuring RTU Parameters

Select an RTU and its general parameters are shown.

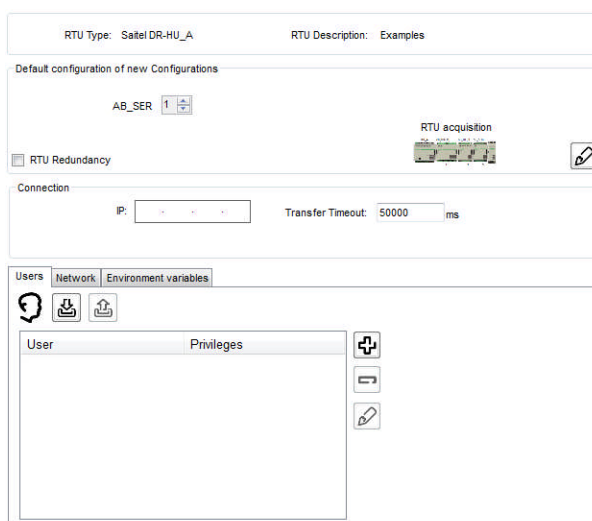


Figure 3-5 - Configuring RTU parameters.


For the selected RTU in the tree, the following information is shown:

- RTU type and description.
- Default information that will be used when a new Configuration is created:
  - Communication modules (AB\_SER, SM\_SER or K7).
  - RTU with redundancy or not (only for HU\_A, HU\_AF, SM\_CPU866 and SM\_CPU866e).
  - RTU acquisition: I/O modules installed in the RTU (only for Saitel).

This information can be edited to modify the default values for new Configurations.

### 3.2.2.1 Defining a Default Configuration with Saitel DR

In “AB\_SER”, select the number of AB\_SER modules to be included for each new configuration.

Pressing button  next to the graphical ITB, you can add, remove or change the I/O modules included on the default configuration.

The user needs understand some basic concepts about Saitel DR before configuring the acquisition:

- An **ITB** is a set of acquisition blocks connected to a CPU (HU).
- An **Acquisition Block** or **AB** is a Saitel DR input/output module.

- Each acquisition block is allocated to a unique address in the ITB, the **Node Number**; this number identifies both the module and its I/O points. In the case of HU\_AF and HU\_BI, the HU has also a node number assigned (In this case, the HU node will take address number 1 and cannot be changed).
- The procedure **AAP (Automatic Addressing Procedure)** is performed by the operator every time an AB module is added, deleted, replaced or moved inside the ITB. It can be launched manually or automatically by setting switch #3 in the HU module to auto (please, refer to “Saitel DR Modules Manual”).

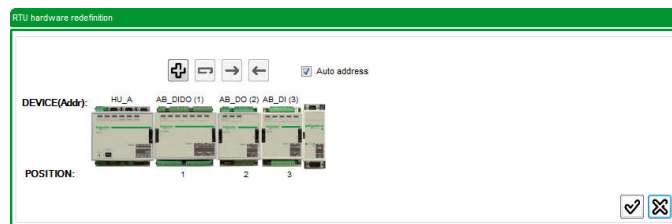


Figure 3-6 - Acquisition modules in a default configuration.

The number between parentheses next to each module's name is the node number. You can select an AB and use buttons to change its physical position.

When “**Auto Address**” box is checked (by default), if you reorder, add or delete an AB, all addresses are automatically recalculated matching their physical position in the rail. Address number 1 is assigned to the AB closest to the HU module (for HU\_AF or HU\_BI, address number 1 will be attached to the HU itself).

If “**Auto Address**” box is unchecked, modules will retain the allocated address, ignoring any changes made. If rechecked, the following message will appear:

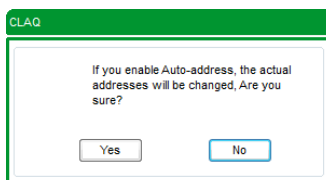


Figure 3-7 – Confirmation for automatic addressing.

Select an AB (click on the AB image) and use button to remove it.

Use button to add a new AB. Select the type of AB in the following window:

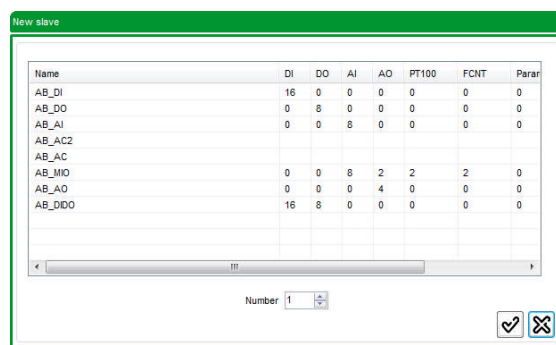


Figure 3-8 – Adding one (or several) AB.

If “**Auto Address**” is checked, you can add several AB at one time. This window allows selecting the quantity of modules to be added. If “**Auto Address**” is unchecked, you only can add one AB each time and you have to select the address to be assigned.

### 3.2.2.2 Defining a Default Configuration with Saitel DP

Figure 3-9 - Configuring a Saitel DP RTU.

In “SM\_SER”, select the number of SM\_SER modules to be included for each new configuration.

Pressing button you can add, remove or change the RTUs default acquisition modules.

Figure 3-10 - Acquisition modules in the default configuration.

Select a module (click on the image) and use button to remove it. Use button to add a new I/O module. Select the type of module in the following window:


Name	Type	DI	DO	AI	AO
SM_DI32	SM_DI32	32	0	0	0
SM_DO16R	SM_DO...	0	16	0	0
SM_DO32T	SM_DO...	0	32	0	0
SM_AI8AO4	SM_AI8...	0	0	8	4
SM_AI16	SM_AI16	0	0	16	0
SM_GAS	SM_GAS	12	4	7	2
SM_AC_Power	PLC				
SM_AC_Syncrocheck	PLC				
PLC	PLC				

Figure 3-11 – Adding new I/O modules.

## NOTICE

Legacy devices are shown in orange color.

The address assigned to the I/O module is shown under its image. You can change this address clicking button . The configuration of any module integrated in Saitel DP, including all the changes made to it can be stored as a template. Thus, this configuration can be repeated in this project or in any other project in the future.

You can use button  to use a template, that is, an XLB file containing a pre-configured set of I/O modules

Press button  in order to configure Profibus configuration and acquisition strategy:

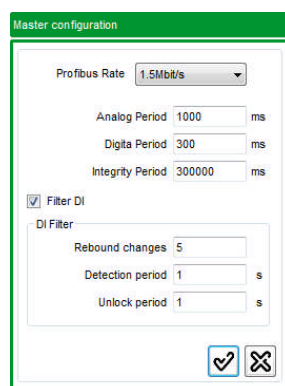


Figure 3-12 – Profibus configuration.

- **Profibus Rate:** CPU communication rate with I/O modules.
- **Analog and Digital Period:** Interval of time for the acquisition of analog and digital input points when they are configured to be updated periodically (ChgEvt of the point is set to “N”). Default value for digital points is 1000 ms (10 ds) and for analog points is 300 ms (3 ds). Both values can be changed in intervals of 100 ms.
- **Integrity period:** When a digital point is set to be updated by event (ChgEvt of the point is set to “Y”), this value indicates that if during this time no event occurred, the point is updated anyway. This assures the integrity of the point. Default value is 300 s and it can be changed in intervals of 100 ms.
- **Filter DI:** This checkbox allows you to configure the filtering parameters for digital inputs.
  - **De-bounce changes:** Number of changes necessities in order to activate the anti-debounce filter (default value, 5).
  - **Detection period:** Time window when the number of rebound will be counted in order to activate the debouncing filter blocking the point. This time is expressed in seconds (default value, 1 s).
  - **Unlock period:** Time without changes in a blocked point in order that this point is unblocked. This time is expressed in seconds (default value, 1 s).

### 3.2.2.3 Defining a Default Configuration with Easergy T300

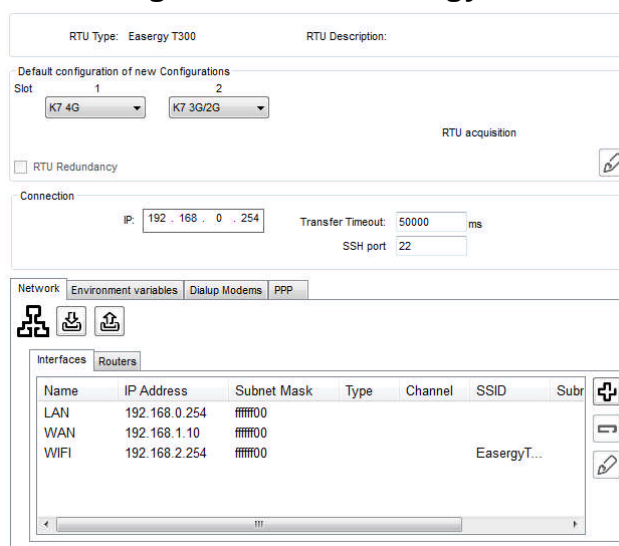


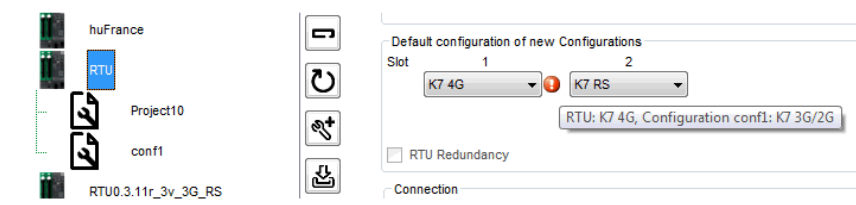
Figure 3-13 - Configuring an Easergy T300 RTU.

For T300 RTU, the button  has no effect for this RTU.

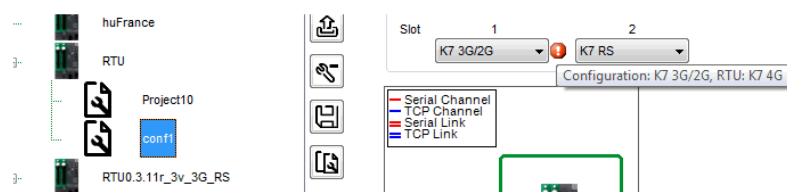
In Slot 1 and 2, select the modem type to be used to manage the communication with the control centre or to manage communications with third party device. You can select:

- No modem (blank).
- K7 RS: Serial modem RS-232 / RS-485
- K7 3G/2G: A 3G/2G modem, including GPS.
- K7 4G: A 4G modem, including GPS

A warning message could be shown next to Slot 1 or Slot 2. This message means that for this RTU, at least one configuration has associated a different device for this slot. For this configuration, this message will be shown too. For example, for the RTU:



And, for the configuration:



### 3.2.2.4 Configuring the Connection with RTU (Simple or Redundant)

If “RTU Redundancy” is unchecked only the IP address (corresponding to a unique CPU) has to be configured:

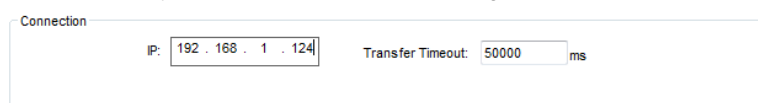


Figure 3-14 – RTU connection parameters.

FTP Timeout is the time (in milliseconds) to consider the FTP connection broken when a configuration is being sent to the RTU (default value: 50 s).

In redundant architectures (only for Saitel), such as the one depicted below, the user can check “RTU Redundancy”:

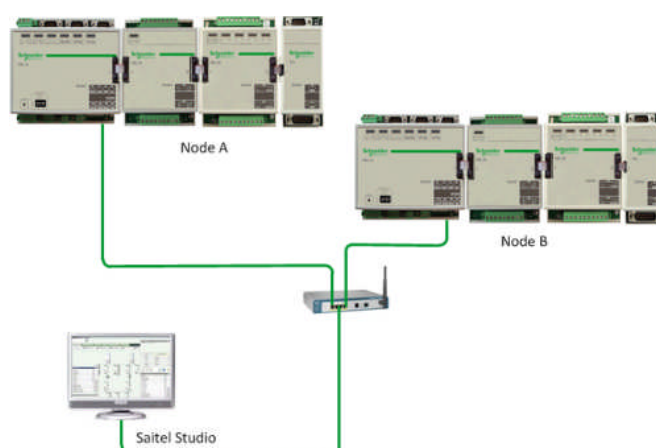


Figure 3-15 – Example of a Saitel DR redundant configuration.

In this case you have to configure a default acquisition with only one ITB.

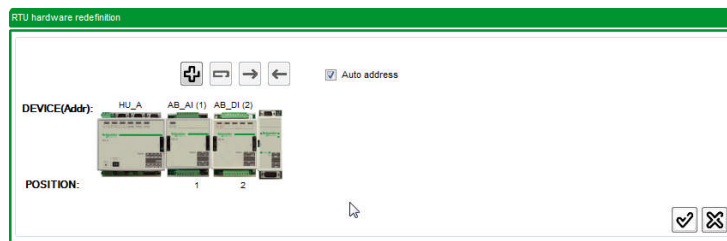


Figure 3-16 – Unique ITB in a redundant configuration.

On Connection of the RTU parameters, the IP address of each HU must be configured:

Figure 3-17 – Connection parameters in a redundant configuration.

You have to configure in Easergy Builder only one ITB with two IP addresses, and when the configuration is sent to the RTU, you will be prompted to select “CPU A”, “CPU B” or “Both”.

Figure 3-18 – Sending information to a redundant RTU.

If option “Both” is selected, Easergy Builder will send the configuration in this order:

- Send configuration to CPU A.
- If the file transfer is OK, send the configuration to CPU B.
- If OK, ask to reboot CPU A.
- If OK, ask to reboot CPU B.

If an error occurs, the operation will stop.

### 3.2.2.5 Configuring RTU Users

#### 3.2.2.5.1 HU250

For HU250, the Security Administration tools shall be used for user configuration. Please, consult T300 User Manual and Quick Start.

#### 3.2.2.5.2 (Only Saitel)

You can manage users only for HU\_B, HU\_BI and CPUs running VxWorks (HU\_A, HU\_AF and SM\_CPU866). Others CPUs running Linux, you can't manage users with Easergy Builder if the Cyber-security Brick is used.

If the RTU is redundant, all users are common to both CPUs. In order to configure users, please, select tab “Users” in the RTU configuration window:

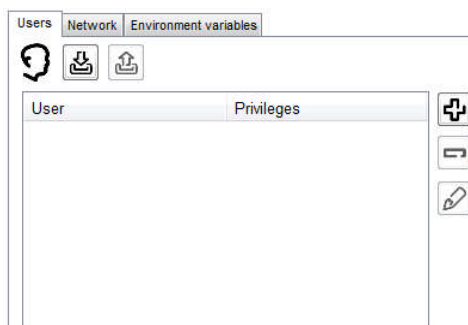


Figure 3-19 – Configuring users.

This tab shows for each defined user its username and privileges. New users can be added and existing users can be modified and deleted.

Pressing button  in the Users tab in will attempt to load in Easergy Builder the users defined in the RTU. To export the user list defined from Easergy Builder to the RTU, use button .

Easergy Builder will ask for a valid user. By default, these credentials to access these files are: **'target'** as Login and **'password'** as Password if the CPU running VxWorks, **"admin"** as Login and **"11111111"** as Password if the CPU running Linux and the Cyber-security Brick is not used.

This default user allows you to connect to the RTU when there aren't other users defined. If there is a user list defined in the RTU you have to use a valid user from this list.

## NOTICE

Buttons "Read" and "Overwrite" in the Users tab only reads and transfers the file userLogin.xml. These mustn't be mistaken with button "Read Configuration" in the Administration toolbar and "Send Configuration to RTU" in the main menu.

## Adding Users

Press button  to add new users by entering the required information in following fields:

Figure 3-20 – New user.

Where:

- **Login:** User identifier.
- **Privileges:** 3 privilege levels can be defined:
  - Administrator. The user can connect to the database both via telnet and SFTP. The SFTP connection enables access through Easergy Builder.
  - Advanced User: The user can force and block database points using Webtool.
  - User: The user can monitor database points from Webtool. No points can be forced or blocked.
- **Password:** User-associated keyword. This password length is 8-40 characters.
- **Confirm Password:** The same password must be entered again in this field (case sensitive).

Depending on the privileges selected for the user, the following actions are available:




Right	Access to the file system	Access to the application shell	Access to the OS shell	Saitel Webtool (Read)	Saitel Webtool (Write)
Role					
Administrator	✓	✓	✓(*)	✓	✓
Advanced user	✓	✓		✓	✓
User					


Table 3-1 – User rights.

(\*) Only for HU\_A, HU\_AF or SM\_CPU866. If the CPU is a SM\_CPU866e or HU 250 the user needs to have root privileges.

## Deleting Users

To delete a user, select it in the list and press button .

## Editing Users

Change the information associated to a specific user (name, privileges or password) pressing button  next to the user list. The window shown in Figure 3-20 will appear with the information about the user selected.

### 3.2.2.6 Configuring Network

You can see two or four tabs for configuring the network nodes depending on if the RTU is redundant or not.

If the RTU has one CPU, only sections “**Network**” and “**Environment variables**” will be shown. If the RTU is redundant, sections “**Network – CPU A**”, “**Network-CPU B**”, “**Environment variables A**” and “**Environment variables B**” will be available. IP addresses for CPU A are included in section “Network – CPU A”, and IP address for CPU B are included in “Network – CPU B”.

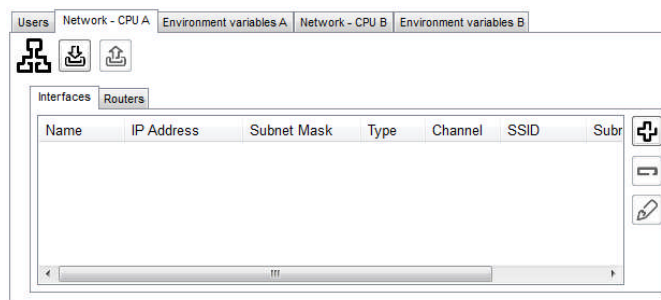




Figure 3-21 – Configuring networks with redundant CPUs.

Press button  in the Network tab in order to load in Easergy Builder the interfaces and routers defined in the RTU (CPU A or CPU B if it is a redundant RTU). Use button  to export the network interfaces and routers from Easergy Builder to RTU.

## NOTICE

This button in the Network tab only transfers to the RTU the file net\_config.xml. You mustn't confuse this button with button “Read Configuration” in the Administration toolbar.

## Interfaces




The number of interfaces to define depending on the CPU type:

- **HU\_A** and **HU\_AF**: EHT1, ETH2 and PRP1 (ETH1 & ETH2).
- **HU\_B** and **HU\_BI**: ETH.
- **SM\_CPU866** and **SM\_CPU866e**: ETH1, ETH2, ETH3, ETH4, PRP1 (ETH1&ETH2) and PRP2 (ETH3&ETH4).
- **HU 250**: WAN, LAN and WIFI.

PRP (Parallel Redundancy Protocol) allows using two physical ports as a unique logical port, with a same MAC address and IP. This protocol isn't available for basic CPUs of Saitel DR (HU\_B and HU\_BI) nor HU 250.

## NOTICE

If you define PRPx interfaces, the associated ETH port mustn't be defined.

Use buttons ,  and  to add, remove and edit network interfaces.

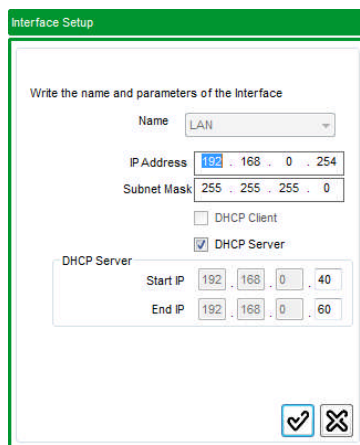


Figure 3-22 – Configuring a network interface.

Depending on the CPU type, you can define this interface as a DHCP client, DHCP Server or any.

- If “DHCP Client” is checked, it isn't necessary fill the “**IP Address**” and “**Subnet Mask**” fields. These data are generated automatically by other DHCP server and they are assigned to the interface.
- If “DHCP Server” is checked, you have to fill the fields “IP Address” and “Subnet Mask” with the IP address for this interface. On the other hand, you have to define an IP range to be assigned to other DHCP clients on the network.

When a WIFI interface is being defined (only for HU 250), the configuration windows is as follow:

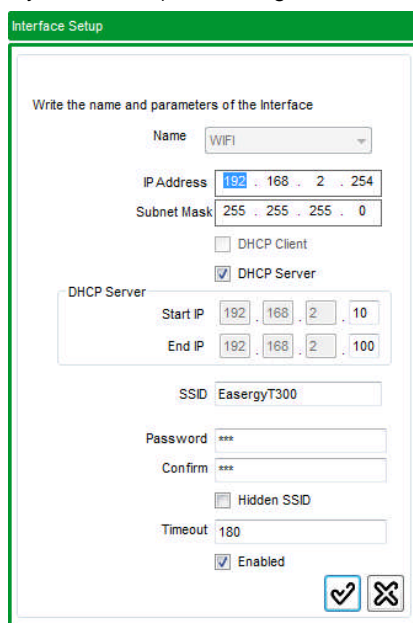


Figure 3-23 – Configuring a WIFI network interface.

Where:

- **SSID**: Unique identifier for the wireless LAN. By default, this name is “EasergyT300”.
- **Password** and **Confirm**: Necessary password that should be set in a device in order to it can be connected to the wireless LAN
- **Hidden SSID**: When this box is checked the SSID visibility is deactivated.

- **Timeout:** Maximum time to wait before deactivate the wireless LAN when any connection is established. This disconnection timeout doesn't apply when the WIFI was started trough the Local/Remote button.
- **Enabled:** Checking this box allow the use of the wireless LAN. If this box is unchecked, the wireless LAN can't be started any way.

## Routers

If there are Interfaces on different networks, you need a router in order to access them. Each one of this interface has to be define in the label "Routers" with the router IP that allows the access. The following figure shows a possible situation where there are two external subnets:

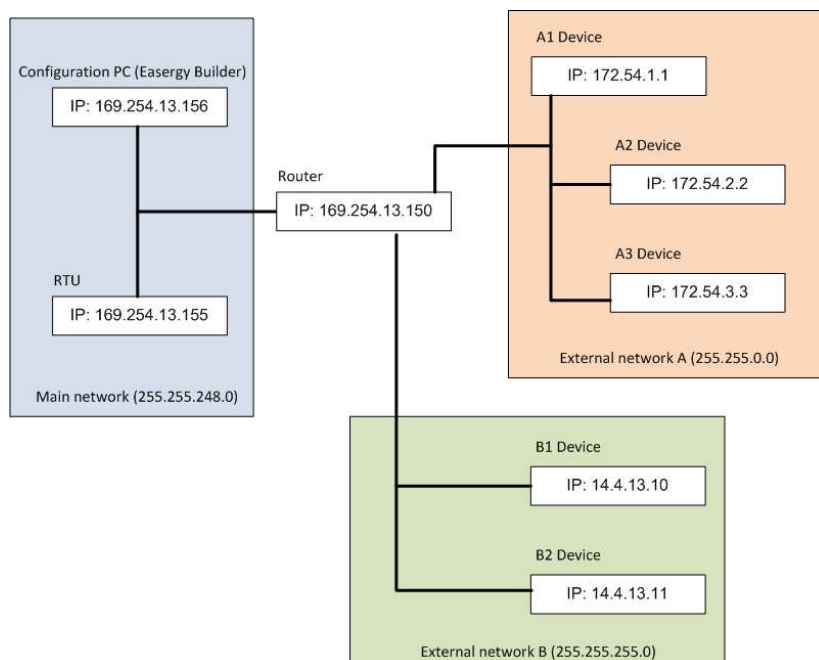


Figure 3-24 – Example of external subnets.

The configuration is shown in the next figure:

Interfaces Routers			
Subnet IP	Subnet Mask	Router IP	
172.54.0.0	ffff0000	169.254.13.190	
14.4.13.0	ffff00	169.254.13.190	

Figure 3-25 – Configuring external subnets.

To add new routers, press button  and the following screen will appear:

The 'Router Setup' dialog box has a green title bar and a blue background with a router icon. It contains the text 'Write the parameters of the router'. Below this, there are three input fields: 'Subnet IP:', 'Subnet Mask:', and 'Router IP:'. At the bottom right, there are two buttons: a checkmark icon and a close icon.

Figure 3-26 – A new subnet

- **Subnet IP y Subnet Mask:** Set of external IP addresses which are accessible through the router.
- **Router IP:** Router's IP address in the main network.

It is possible to include a single record for an external network where the IP address as well as its mask are the default. Using this unique record allows the CPU access to any external device on a network connected to the router.

Subnet IP	Subnet Mask	Router IP
0.0.0.0	00000000	169.254.13.190

Figure 3-27 – Configuring an external network using the default IP and mask

### 3.2.2.7 Environment Variables

Use this tab in order to edit the environment variables for a CPU. If you are setting a redundant system, two tabs will be available, for CPU A and CPU B.

Figure 3-28 – Environment variables.

You can change add, removed or edit variables into the RTU only if you previously read all defined variables from the RTU with button . If not, the others buttons on this tab will be unavailable.

The environment variables are stored in the main\_cfg.xml file. Depending on the type of CPU, the software baseline includes the followings variables:

- **CONFIG\_DIR:** Path for configuration files (Default value: /mnt/flash/cfgFiles/)
- **WEB\_DIR:** Path for Saitel WebTool files (Default value: /mnt/bf/webFiles/)
- **BIN\_DIR:** Alternative path for binary files. If the binary files can't be located in the main path (/mnt/flash), then they will be looked for into the alternative path. (Default value: /mnt/bf).
- **SLOT:** In a dual system, indicates if this CPU has to start as HOT (value A) or BACKUP (value B). (Default value: A. This value must be changed to B in the secondary CPU)
- **MONITOR:** Path for the BLMon tool (Default value: /mnt/flash/BLMon/)
- **WEB\_IS\_REMOTE:** When this variable is set to N, it doesn't have effect. If it is set to S, in case that RTU is in LOCAL mode, commands from Saitel WebTool can't be executed. (Default value: N)

### 3.2.2.8 Dialup Modems and PPP (Only HU 250)

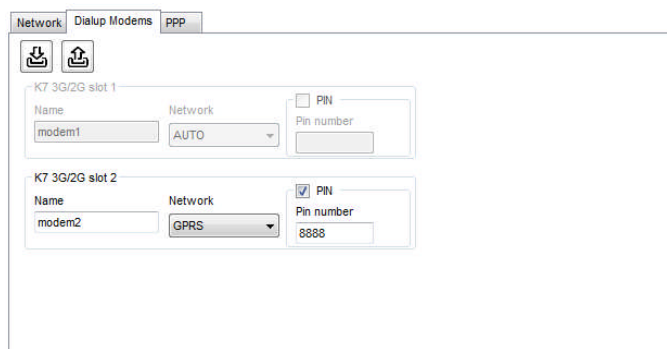


Figure 3-29 – Dialup modems configuration.

You can configure only the slots with a 3G/2G or 4G modem assigned.

- **Name:** Name of the connection.
- **Network:** Connection type: 3G, 4G, GPRS or AUTO. If Auto
- **PIN and Pin number:** Check PIN box if this code is necessary to initialize the connection. If during initialization, the PIN number is empty or wrong, an error will be returned by the modem. In this case, the PIN initialization will be retried 3 times and stop if the issue is still present.

Select tab “PPP” to configure a PPP connection:

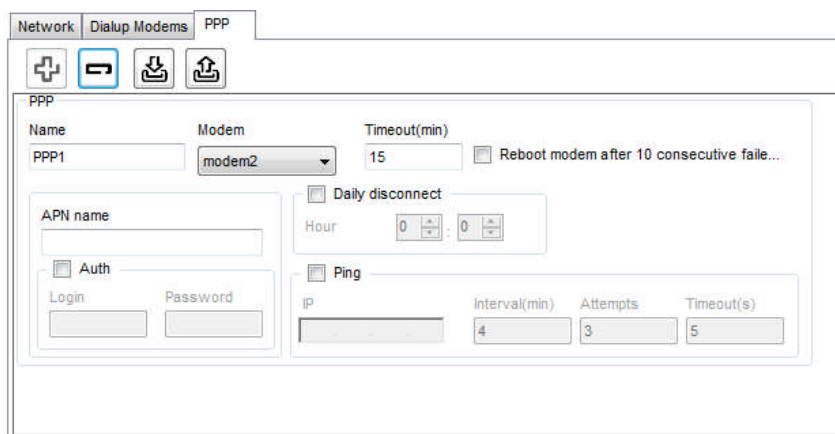




Figure 3-30 – PPP configuration.


Use buttons  and  to add or remove a PPP connection.

- **Name:** PPP connection's name.
- **Modem:** Name of the connection by modem used (This name must be defined in tab “Dialup Modems”)
- **Timeout** (in minutes): If no data flow is detected during this time, a disconnection is made. (0 means disabled).
- **Reboot modem after failed connection:** Mark this box if the modem must be rebooted after max connection retries.
- **APN name:** Name of the Access point.
- **Daily disconnect and Hour:** If this box is checked a disconnection is made daily at indicated time in Hour.
- **Auth, Login and Password:** If authentication is required to connection, check Auth box and indicate a valid login and password.
- **Ping:** Check this box to monitor the IP connection. The following data have to be indicated:
  - IP address of the host to be ping
  - Interval: Interval repetition in minutes of the ping process.
  - Attempts: Number of successive ping.

- Timeout: Timeout in seconds for ping request.

Use buttons  and  to read / send the information from / to the RTU.

### 3.2.3 Removing an RTU

In order to remove an RTU and all its associated configurations from Easergy Builder, select the RTU and press the  button. A confirmation window will be shown:

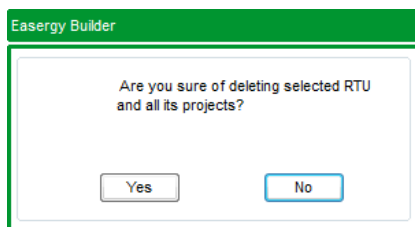



Figure 3-31 – Removing an RTU.

Press "Yes" and the RTU will disappear from the workspace.

### 3.2.4 Rebooting an RTU

Use button  to reboot an RTU (It is like executing "reboot" on the console tool). If the RTU is configured as redundant, Easergy Builder will ask which CPU is rebooted.

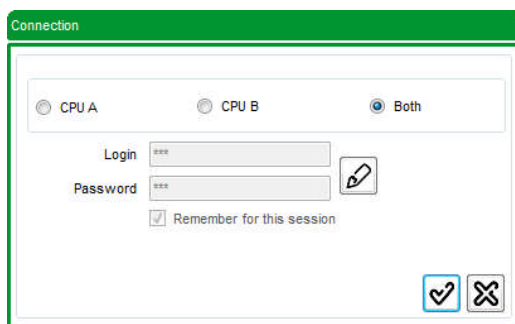


Figure 3-32 – Reboot an RTU.

You can select: only CPU A, only CPU B or both.

When you select Both, CPU A will be rebooted first. If the reboot is completed successfully, Easergy Builder will try to reboot CPU B. If the reboot fails (for example caused by a connection problem), the reboot operation will be stopped and an error message shown:

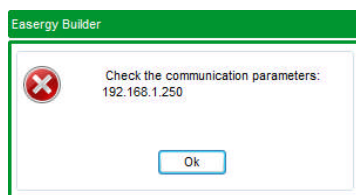



Figure 3-33 – Error rebooting an RTU

### 3.2.5 Reading the RTU Configuration

For an RTU, you can create in Easergy Builder a new configuration using the information stored inside the RTU.

To accomplish this, select the RTU in the Workspace and press button . Easergy Builder will ask for a valid user to retrieve the information. If the RTU is redundant, you also have to select CPU A or CPU B.

Next, select a name for the new configuration:

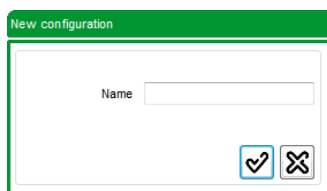


Figure 3-34 – New configuration's name

If the connection is established, all the files containing the database information will be transferred from the RTU to the PC. In the Workspace, the RTU tree view will add this new configuration associated to the selected RTU.

## NOTICE

This button only imports the database configuration files. To import other configuration files (e.g. users and network) consult paragraphs 3.2.2.5 and 3.2.2.6.

### 3.2.6 Loading a Configuration

Working with Easergy Builder, you can use button  to load a configuration (only the information in coreDb) or a complete RTU configuration (including users, network interfaces, ...).

Following window will be shown if you select a Saitel RTU:

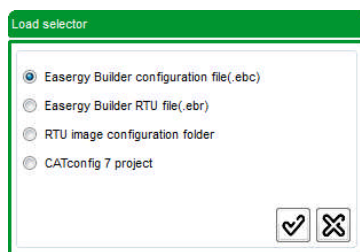


Figure 3-35 – RTU sysLog.

This window allows loading:

- A configuration (coreDb) for an RTU stored in an EBC file.
- A complete configuration (including users, network interfaces, ...) for an RTU stored in an EBR.
- An RTU image configuration folder. Select in your PC the folder where the configuration files for the new RTU are stored.
- Import an old project generated with CATconfig 7.

When a Easergy T300 RTU is selected, following window is shown:

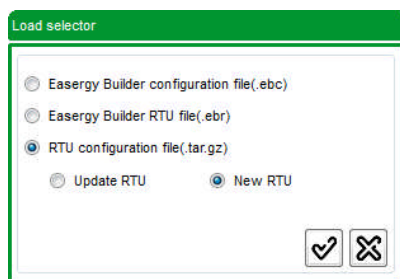



Figure 3-36 – RTU sysLog.

In this case you can load an RTU image that is stored in a ".tar.gz file". This file was generated by the WebApp tool of the T300. Depending on the selected radio button, configuration will be loaded over an existing RTU or a new RTU will be created.

## 3.2.7 Reading the sysLog File from the RTU

If you need consult the log file in the RTU (sysLog), you can import it using Easergy Builder. Press button  and use a valid user for the connection.

Easergy Builder will import the file and show its content in the Log view zone, next to the Log console.

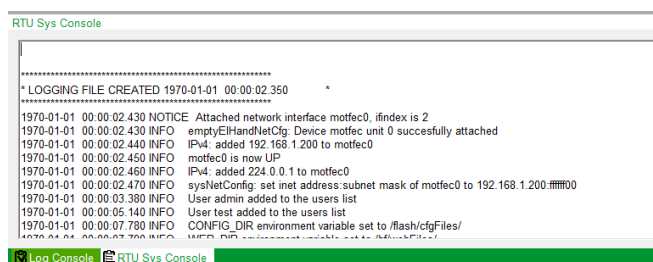



Figure 3-37 – RTU sysLog.

In the Log view zone you can switch between the Log console (information messages from Easergy Builder) and the RTU Sys Console (contents of the sysLog)

There is a big difference between these two views. The Log console shows dynamic information about the operations in Easergy Builder. The RTU Sys Console shows a “snapshot” of the information in the sysLog taken at the moment it was loaded from the RTU. If you want to update the information in the RTU Sys Console you have to reload the sysLog file again from the RTU.

## 3.2.8 Adding a New Configuration

This section shows how you can create a new configuration using Easergy Builder.

Select the RTU and press button :

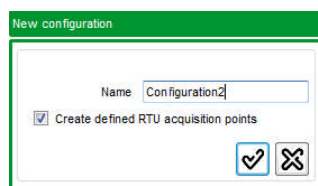


Figure 3-38 – New configuration

Write the name of the new configuration.

If the RTU was created with an acquisition configuration by default (only for Saitel), for a new configuration you can select the field “Create defined RTU acquisition points” in order to include all points of the pre-defined I/O modules in coreDb.

For example, if a Saitel DR RTU was created with a default configuration including one AB\_DI, one AB\_AI and one AB\_DO, if you select “Create defined RTU acquisition point”, in coreDb the following registers will be included:

- 16 digital inputs in status table.
- 8 analog inputs in analog table.
- 8 digital outputs in status table.

Other points associated to each AB will be included too.

For Easergy T300 RTU, only supervision points will be generated in coreDB (status, command and analog). You can find more information about these points in section 4.2.1.4.

The new configuration is available for the RTU.



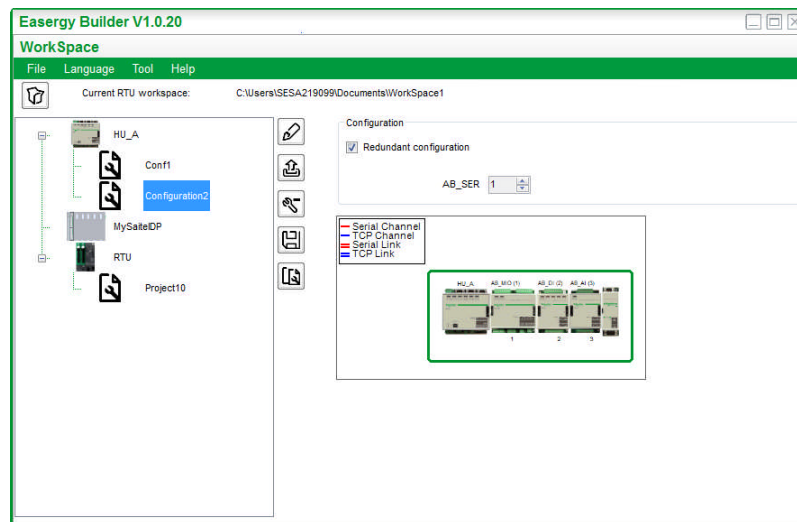


Figure 3-39 – New configuration in the RTU tree

Double clicking on it will switch to Configuration mode where the new configuration can be edited.

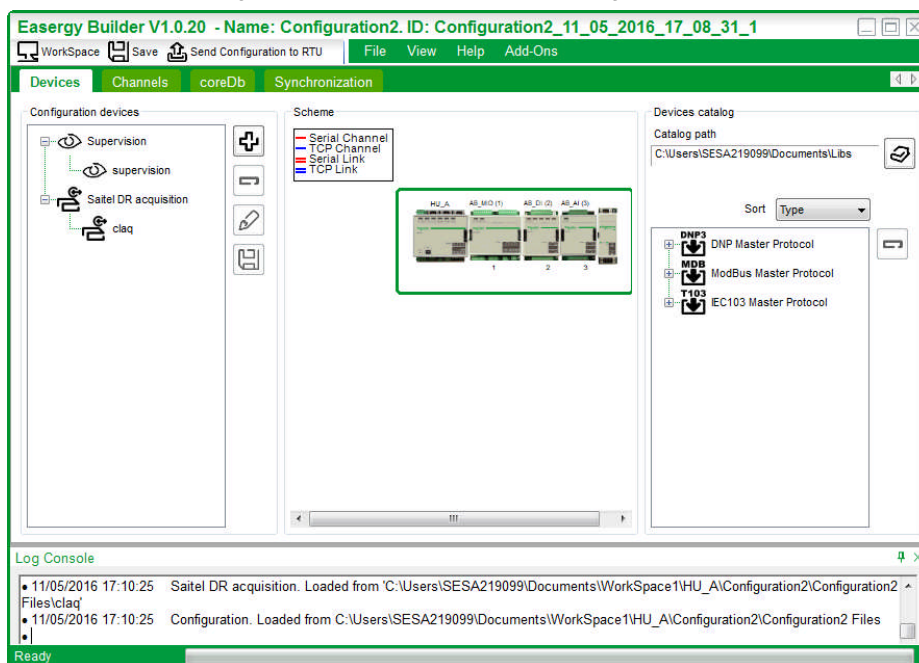


Figure 3-40 – Configuration mode

Right-clicking on the configuration name, a contextual menu is displayed:

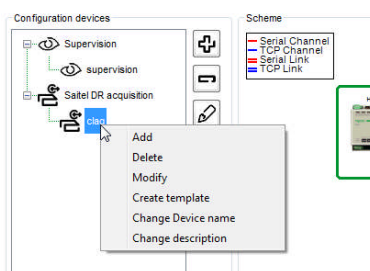
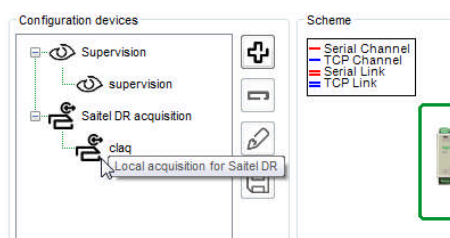


Figure 3-41 – Contextual menu for Configuration

---

This menu allows:

- Add, remove or modify a configuration.
- Create a template with the information associated to this configuration.
- Change the name or description of the Device (description field allows 128 characters maximum. The description is shown when you locate the mouse over the Device name as follow:



# Chapter 4 - Working with Configurations

## 4.1 Introduction

The next figure shows an example of Easergy Builder when you are editing a specific configuration for an RTU

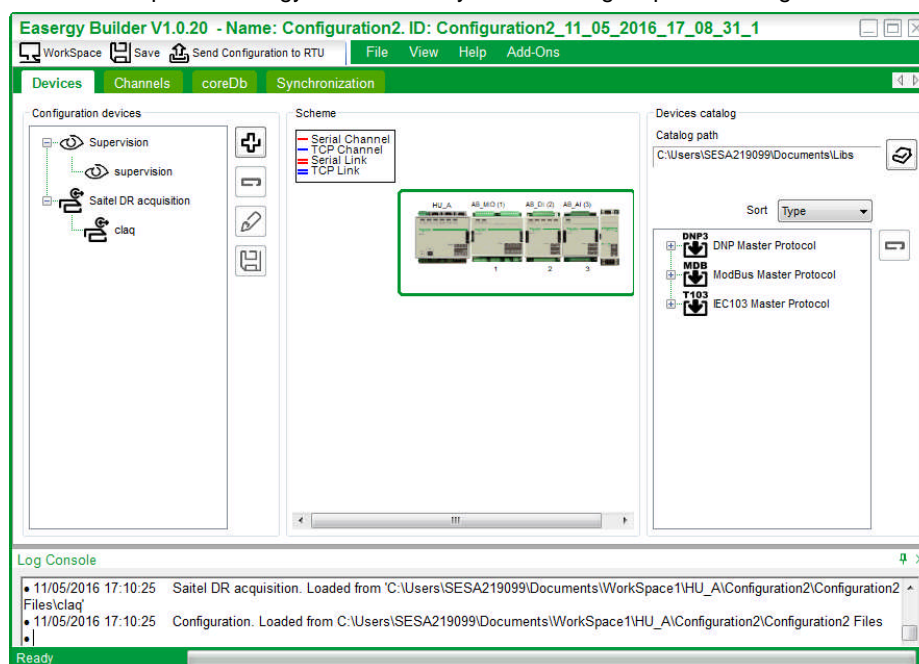


Figure 4-1 – Easergy Builder configuration mode

Please, consult section 2.3.2 for detailed information about this mode.

## 4.2 Devices

The information exchange between the environment and coreDb is made through Devices. Each type of Device defines the rules that must be followed when designing the set of real-time points, and the relationship between them and with coreDb points.

Each Device is associated with a controller that must be installed in order to be available in Easergy Builder.

When you add a new Device, you have to select the type depending on the RTU:

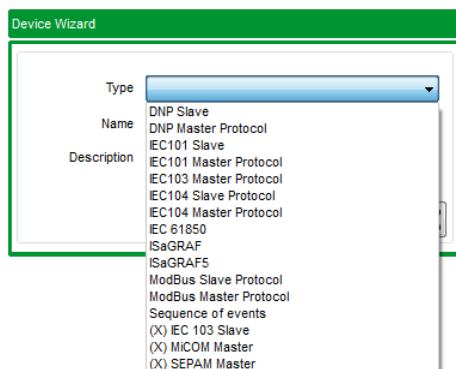


Figure 4-2 – Selecting the type of the new Device

When a Device is available for a type of RTU but its configuration plugin hasn't been installed, its name is shown with an (X) before the name. You can consult information about available devices for each CPU in Table 3.

This software implements the rules to follow and offers a friendly graphical interface for points configuration. Every Device controller into this list can be installed and uninstalled independently of Easergy Builder.

To know which Devices are available in Easergy Builder and the plugin versions, please, select **Help → About** in the main toolbar.

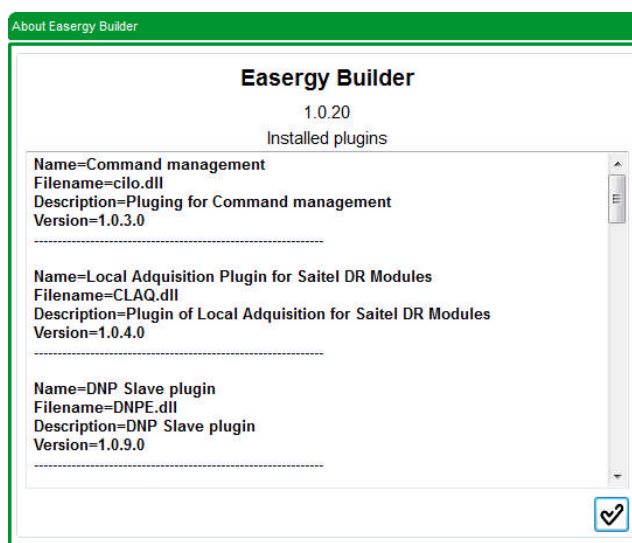



Figure 4-3 – Information about plugins that have been installed in Easergy Builder.

In order to add a new Device, you can use button  in the toolbar of the tab Device, while in Configuration mode or you can click right-button of the mouse on the Device tree and select “Add”.

## 4.2.1 Supervision Device

The Supervision Device is installed by default with Easergy Builder. This device is used to monitor the states of all CPU components, and to generate status information which can be used by other RTU components.

The supervision device is closely linked to for each CPU hardware, so different mappings are defined for the generated points. About supervision, the user only has to select the correct points depending on the type of RTU.

The following section details the monitoring points generated by the supervision module and the monitoring points available for each CPU type.

To see a select the monitored elements, please, double click on the tree, in the supervision Device:

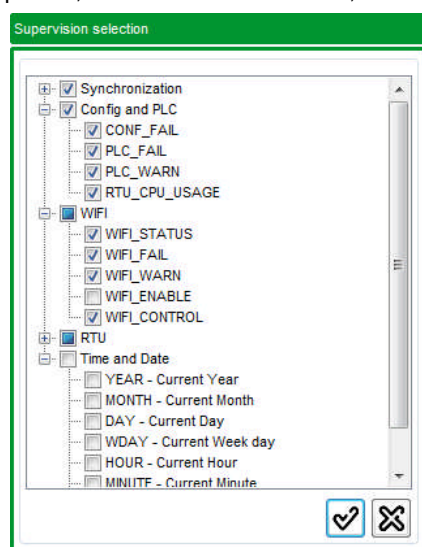


Figure 4-4 – Supervision points for HU 250

In this window you can select or deselect the supervision points depending on the RTU type.

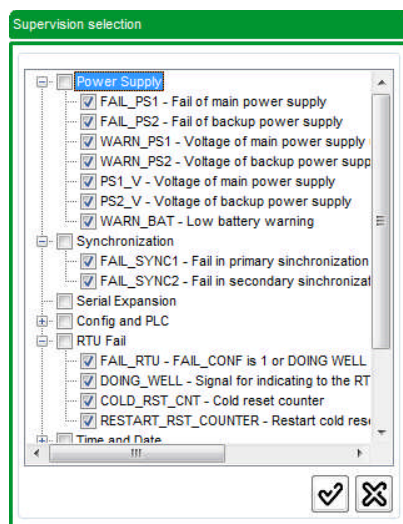


Figure 4-5 – Available supervision points for SM\_CPU866e

It is always advisable to load the supervision Device, which is required to monitor the above mentioned elements of the CPU and essential in redundant configurations. The CPU itself cannot arbitrate with another CPU in redundant systems, so there must be a monitoring part. Currently, only the supervision Device can do these tasks, and in practice only this Device is used.

The information generated by the monitoring module is supplemented with the control and diagnostic information generated in each Device.

### NOTICE

For supervision device, coordinates match the name. For example, the coordinate associated to the FAIL\_PS1 point is "FAIL\_PS1".



### WARNING

Depending on the RTU type, available points are different. The user should not map the points which are not available for the CPU model under installation.

In following sections, for each point, the description corresponding to the value 1 is explained.

#### 4.2.1.1 Common Supervision Points (Only Saitel DP and Saitel DR)

Following points are available for HU\_A, HU\_AF, HU\_B, HU\_BI, SM\_CPU866 and SM\_CPU866e:

Point	Table	Type	Description
<b>Power Supply</b>			
WARN_BAT	Status	Source	Allows knowing the system battery state.
<b>Synchronization</b>			
FAIL_SYNC1	Status	Source	Indicates if there's a failure in the main synchronization source.
FAIL_SYNC2	Status	Source	Indicates if there's a failure in the secondary synchronization source.
<b>Serial Expansion</b>			
FAIL_SER1 ... FAIL_SER8	Status	Source	Active, indicates if there's a failure in the correspondent module (FAIL_SER1 indicates failure in module 1, and so on). Only will be available the points corresponding to the communication modules installed in the RTU.

Point	Table	Type	Description
<b>Configuration and PLC</b>			
<b>FAIL_CONF</b>	Status	Source	Active (1) indicates configuration failure.
<b>CPU_USAGE</b>	Analog	Source	CPU usage (%).
<b>RTU Fail</b>			
<b>FAIL_RTU</b>	Status	Source	If active (1), indicates that the RTU is in an anomalous state. If FAIL_RTU is 0, there's no configuration error (FAIL_CONF == 0), input signal (DOING_WELL == 1) and all tasks registered to the watchdog are responding. While (FAIL_RTU == 0), RTS and DTR pulses are generated. If any task is unresponsive (does not refresh watchdog timer), FAIL_RTU will be 1.
<b>Time and Date</b>			
<b>YEAR</b>	Analog	Source	Current year.
<b>MONTH</b>	Analog	Source	Current month.
<b>DAY</b>	Analog	Source	Current day.
<b>WDAY</b>	Analog	Source	Current week day. (0: Sunday, 1: Monday ... 6: Saturday).
<b>HOURL</b>	Analog	Source	Current hour.
<b>MINUTE</b>	Analog	Source	Current minute.
<b>SECOND</b>	Analog	Source	Current second.
<b>Temperature</b>			
<b>TEMP</b>	Analog	Source	Current chip temperature.
<b>GPS Synchronization</b>			
<b>FAIL_SYNCHW</b>	Status	Source	Indicates hardware failure if active (1). If there's no failure, FAIL_SYNCHW will be 0.
<b>FAIL_SYNCDESV</b>	Status	Source	If active (1) indicates that there's a 3 millisecond deviation.
<b>Link</b>			
<b>LINK:MOTFEC0</b>	Status	Source	Link in ETH1.

Table 4-1 – Common supervision points for all Saitel CPUs.

#### 4.2.1.2 Saitel DR Supervision Points

Following points are available for Saitel DR CPUs depending on its type, advanced (HU\_A and HU\_AF) or basic (HU\_B and HU\_BI).

Point	Table	Type	Description	CPU
<b>Configuration and PLC</b>				
<b>FAIL_PLG</b>	Status	Source	If ISaGRAF is used, it will indicate that there's no PLC program or that the program is stopped if FAIL_PLG is 1.	Advanced
<b>PLC_WARNING</b>	Status	Source	If ISaGRAF is used, it will indicate that there are unmapped ISaGRAF signals in coreDb if PLC_WARNING is 1.	Advanced
<b>Redundancy</b>				
<b>RED_VIA1_FAIL</b>	Status	Source	Active (1) indicates failure in the main communication line of the RCAP protocol.	Advanced

Point	Table	Type	Description	CPU
RED_VIA2_FAIL	Status	Source	Active (1) indicates failure in the secondary communication line of the RCAP protocol.	Advanced
RED_I_STATE	Status	Source	Indicates redundancy state of the RTU where the supervision controller is installed. If RED_I_STATE is 1, the local node is ONLINE or STANDBY. If RED_I_STATE is 0, the local node state is FAIL.	Advanced
RED_IT_FAIL	Status	Source	Indicates redundancy state of the other RTU (dual). If the other RTU is in FAIL state, this signal will be 1. If this signal is 0, then the other RTU is ONLINE or STANDBY.	Advanced
COM_CTS	Status	Source	Indicates the state of the CTS pin of the serial port that communicates with MSAC. COM_CTS will be 0 and 1 if MSAC is sending 0 or 1. If COM_CTS is 2, the signal is being received by DTR.	Advanced
DB_UPDATE	Status	Source	Active (1) indicates that a redundant system configured as "Hot data", the database has been successfully updated.	Advanced
NODE_A	Status	Source	Active (1) indicates that the current system is configured as node type A.	Advanced
NODE_B	Status	Source	Active (1) indicates that the current system is configured as node type B.	Advanced
ONLINE	Status	Source	Active (1) indicates that the current CPU is ONLINE in the redundant system.	Advanced
<b>RTU Fail</b>				
DOING_WELL	Status	Destination	This point indicates to the RTU that something external is running ok. <b>If no source is defined for it, the initial value 1 should be assigned.</b> Usually, this signal has ISaGRAF as origin, and indicates that system is working properly when PLC is working	Advanced
COLD_RST_CNT	Status	Source	Cold reset counter (it currently counts all system reboots).	Advanced
RESTART_RST_COUNTER	Command	Destination	Command to restart the system reboot counter.	Advanced
<b>Local Acquisition</b>				
LOCALREMOTE	Status	Source	This signal will show the value of the digital input 2 of the HU_A. If LOCALREMOTE is 1, module is in local mode. In local mode, commands are not activated in the digital outputs modules. If LOCALREMOTE is 0, RTU is in remote mode, which means normal RTU functioning (will also be in remote mode when this point is not mapped). This point can be also configured as LOCALREMOTE:I. This way, the digital input 2 value will be inverted. In this case, if LOCALREMOTE:I is 1, RTU is still in local mode, and if LOCALREMOTE:I is 0, RTU will still be in remote mode. The difference is in how to process the digital input 2. When the digital input 2 is 0, LOCALREMOTE:I will be 1 (local mode) and when digital input 2 is 1, LOCALREMOTE:I will be 0 (remote mode).	Basic / Advanced

Point	Table	Type	Description	CPU
POL_OK_ABDI	Status	Source	If its value is 1, the polarization of the digital inputs module is correct. If its value is 0, there's a failure in polarization or the correspondent point has not been mapped (polarization of the digital input is not watched).	Basic / Advanced
LAQ_FAIL	Status	Source	Active (1) means that there's a failure in acquisition. This means that one of the acquisition modules is out of service or in a failure state.	Basic / Advanced
Link				
LINK:MOTFEC1	Status	Source	Link in ETH2 for HU_A and HU_AF.	Advanced

Table 4-2 – Saitel DR supervision points.

### 4.2.1.3 Saitel DP Supervision Points

Following points are available for Saitel DP CPUs. All points are available for standard CPU (SM\_CPU866) and advanced CPU (SM\_CPU866e).

Point	Table	Type	Description
<b>Power Supply</b>			
FAIL_PS1	Status	Source	Active (1) indicates if there's a failure in the main power supply (SLOT 1 in the backplane).
FAIL_PS2	Status	Source	Active (1) indicates if there's a failure in the secondary power supply (SLOT 2 in the backplane).
WARN_PS1	Status	Source	Line PS1 voltage is below the warning level (5.3 V). PS1 is the main power line in the bus. It's associated to the PS in SLOT1
WARN_PS2	Status	Source	Line PS2 voltage is below the warning level (5.3 V). PS2 is the secondary power line in the bus. It's associated to the PS in SLOT2.
PS1_V	Analog	Source	Voltage of PS1.
PS2_V	Analog	Source	Voltage of PS2.
<b>Configuration and PLC</b>			
FAIL_PLC	Status	Source	If ISaGRAF is used, it will indicate that there's no PLC program or that the program is stopped if FAIL_PLC is 1.
PLC_WARNING	Status	Source	If ISaGRAF is used, it will indicate that there are unmapped ISaGRAF signals in coreDb if PLC_WARNING is 1.
<b>RTU Fail</b>			
DOING_WELL	Status	Destination	This point indicates to the RTU that something external is running ok. <b>If no source is defined for it, the initial value 1 should be assigned.</b> Usually, this signal has ISaGRAF as origin, and indicates that system is working properly when PLC is working.
<b>Redundancy (Only available in redundant configurations)</b>			
RED_VIA1_FAIL	Status	Source	Active (1) indicates failure in the main communication line of the RCAP protocol.
RED_VIA2_FAIL	Status	Source	Active (1) indicates failure in the secondary communication line of the RCAP protocol.
RED_I_STATE	Status	Source	Indicates redundancy state of the RTU where the supervision controller is installed. If RED_I_STATE is 1, the local node is ONLINE or STANDBY. If RED_I_STATE is 0, the local node state is FAIL.



Point	Table	Type	Description
RED_IT_FAIL	Status	Source	Indicates redundancy state of the other RTU (dual). If the other RTU is in FAIL state, this signal will be 1. If this signal is 0, then the other RTU is ONLINE or STANDBY.
COM_CTS	Status	Source	Indicates the state of the CTS pin of the serial port that communicates with MSAC. COM_CTS will be 0 and 1 if MSAC is sending 0 or 1. If COM_CTS is 2, the signal is being received by DTR.
DB_UPDATE	Status	Source	Active (1) indicates that a redundant system configured as "Hot data", the database has been successfully updated.
NODE_A	Status	Source	Active (1) indicates that the current system is configured as node type A.
NODE_B	Status	Source	Active (1) indicates that the current system is configured as node type B.
ONLINE	Status	Source	Active (1) indicates that the current CPU is ONLINE in the redundant system.
<b>Local Acquisition</b>			
LOCAL	Status	Destination	If set to 1 and good quality (IV_LQF, IV_LQF, NT_LQF, NT_LQF are 0) coreDb will be in "Local state". If set to 0 and good quality (IV_LQF, IV_LQF, NT_LQF, NT_LQF are 0), coreDb will be in "Remote state". In other cases, coreDb will be in "Unknown state".
<b>Link</b>			
LINK:MOTFEC0	Status	Source	Link in ETH1.
LINK:LNC0	Status	Source	Link in ETH2 for SM_CPU866 and SM_CPU866e.
LINK:LNC1	Status	Source	Link in ETH3.
LINK:LNC2	Status	Source	Link in ETH4.

Table 4-3 – Saitel DP supervision points.

#### 4.2.1.4 Easergy T300 Supervision Points

Following points are available for Easergy T300:

Point	Table	Type	Description
<b>Synchronization</b>			
SYNC1_FAIL	Status	Source	Failure in primary synchronization source.
SYNC2_FAIL	Status	Source	Failure in secondary synchronization source.
<b>WIFI</b>			
WIFI_STATUS	Status	Source	WIFI network connected.
WIFI_FAIL	Status	Source	Error connecting the WIFI.
WIFI_WARN	Status	Source	There is a connection to WIFI in progress.
WIFI_ENABLE	Status	Destination	The destination status permits to associate a status point to enable the wifi.  For example, if you want to enable the wifi by a door contact plugged on the HU250 digital input, you can put this digital input with WIFI_ENABLE as destination
WIFI_CONTROL	Command	Destination	Command to enable temporally the WIFI network. Using field "Timeout" in WIFI configuration window, you can define the period of inactivity to disable the WIFI network.
<b>Configuration and PLC</b>			
CONF_FAIL	Status	Source	Failure in the configuration.
PLC_FAIL	Status	Source	ISaGRAF project is stopped.

Point	Table	Type	Description
PLC_WARN	Status	Source	If an ISaGRAF Device is used, this point indicates that there are ISaGRAF points which are not mapped in coreDb.
RTU_CPU_USAGE	Analog	Source	CPU usage (%).
<b>RTU</b>			
RTU_FAIL	Status	Source	An anomaly has been detected in the RTU. This signal is deactivated when FAIL_CONF =0, DOING_WELL =1, K7_S1_FAIL=0, K7_S2_FAIL=0 and watchdog without timeout expired for any task.
SYS_FAIL	Status	Source	Error in PLC, HU250 or modules (SC150, PS50...). When SYS_FAIL is detected the equipment fault led is red.
SYS_WARN	Status	Source	Warning in PLC, HU250 or modules (SC150, PS50...). When SYS_WARN is detected, the equipment fault led is orange.
SYS_COM_FAIL	Status	Source	At least one modules has a COM fault error.
DEV_FAIL	Status	Destination	Error in a module. Each device has a module health error variable that are linked to a DEV_FAIL point. You can define several points DEV_FAIL.  DEV_FAIL is destination that can be used to link a status from a device that has Failure indication that you want to aggregate in the SYS_FAIL.
DEV_WARN	Status	Destination	Warning in a module. Each device has a module health warning variable that are linked to a DEV_WARN point. You can define several points DEV_WARN.  DEV_WARN is destination that can be used to link a status from a device that has warning indication that you want to aggregate in the SYS_WARN.
DEV_COM_STATUS	Status	Destination	. You can define several points DEV_COM_STATUS.
DOING_WELL	Status	Destination	This signal indicates to the RTU that something external is running ok. <b>If no source is defined for it, the initial value 1 should be assigned.</b> Usually, this signal has ISaGRAF as origin, and indicates that system is working properly when PLC is working.
<b>Time and Date</b>			
YEAR	Analog	Source	Current year.
MONTH	Analog	Source	Current month.
DAY	Analog	Source	Current day.
WDAY	Analog	Source	Current week day. (0: Sunday, 1: Monday ... 6: Saturday).
HOUR	Analog	Source	Current hour.
MINUTE	Analog	Source	Current minute.
SECOND	Analog	Source	Current second.
<b>Modem and PPP</b>			
K7_SX_DET	Status	Source	"X" can take the values of 1 and 2. Set to one when a modem is connected to the slot "X".
K7_SX_FAIL	Status	Source	"X" can take the values of 1 and 2. Set to one when a channel or a modem dial-up is configured and unable to detect the right K7 in slot "X".
MODEM_SX_DET	Status	Source	"X" can take the values of 1 and 2. Set to one when a modem dial-up is configured on slot "X".

Point	Table	Type	Description
<b>MODEM_SX_FAIL</b>	Status	Source	<p>"X" can take the values of 1 and 2. Set to one when an error is detected on modem configured on slot "X". Errors to be considered:</p> <ul style="list-style-type: none"> <li>• No SIM detected. For K7_3G will check the SIM slot 2.</li> <li>• PIN error.</li> <li>• Error starting modem firmware library.</li> <li>• Error at modem initialization script.</li> <li>• Error at connect script, that covers network and hardware issues.</li> <li>• No answer to AT commands.</li> </ul>
<b>MODEM_SX_RSSI</b>	Status	Source	<p>"X" can take the values of 1 and 2. Will show the level of reception of the modem configured in slot "X".</p> <p>In case of error, the invalid flag will be activated.</p>
<b>MODEM_SX_RNET</b>	Status	Source	<p>"X" can take the values of 1 and 2. Will show the network type of the modem configured in slot "X". It can take the following values:</p> <ul style="list-style-type: none"> <li>• 0 : GPRS</li> <li>• 1 : EDGE</li> <li>• 2 : 3G</li> <li>• 3 : HSDPA</li> <li>• 4 : 4G</li> </ul> <p>In case of error, the invalid flag will be activated.</p>
<b>PPP_SX_STATUS</b>	Status	Source	<p>"X" can take the values of 1 and 2. Set to one when the interface is up on ppp over modem on slot "X", otherwise 0. In case of error, the invalid flag will be activated.</p>

Table 4-4 – Easergy T300 supervision points.

## 4.2.2 Formula Device

### NOTICE

For now, this Device is only available for Easergy T300.

The form Plugin is integrated by default with the core of Easergy Builder. It allows configure a software developed in order to do calculation of expression depending on the value of its input variables.

An expression can be any of the defined functions (NOT, SPSTODPS, DPSTOSPS, TEMPO, OR, AND, SCALE, MIN, MAX, IF), an operator expression (+, -, \*, /, <, >, ==), a coreDb point name, an own variable name (FORM\_PERIOD, FORM\_CYCLETIME), a constant value or any combination of them. For example, a function with operator expressions as parameters, an operator expression with functions as parameter, a function or operator expression with coreDb point names or constant as parameters, ...

These expressions are introduced to coreDb as source coordinate of a point. When an expression is calculated, its value in database (value, quality flag and timestamp) is written at the coreDb point which this coordinate belongs to (as value, quality flag and timestamp, respectively, of this coreDb point).

In the next picture, we can see an example of formula expressions at Easergy Builder.

Name	Description	Source1 Device	Source1 Coordinates
st1		formula	NOT(st2)
st2		formula	TEMPO(st3,10)
st3		formula	DPSTOSPS(st3)
st4		formula	DPSTOSPS(NOT(TEMPO(SPSTODPS(st4),21)))
st5		formula	st3
st6		formula	3.42
st7		formula	-99999901
st8		formula	-81
st9		formula	OR(st1,DPSTOSPS(st2),NOT(st3),TEMPO(st4,10))
st10		formula	st3-SPSTODPS(st1)
st11		formula	-2-(-4)

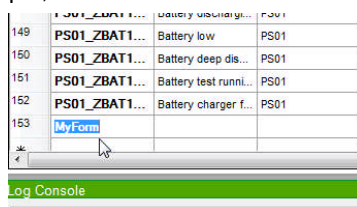
Figure 4-6 – Using formulas in coreDb

Formula Controller also accepts destination coordinates of a point. They are used as triggers to execute source coordinates formula in a coreDb point designed by this destination coordinate name.

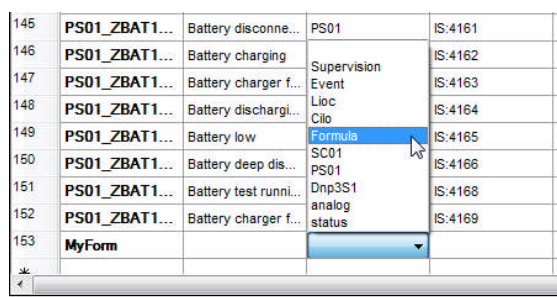
### 4.2.2.1 Using Formulas

In order to be easier building formulas, you can use a wizard in the coreDb tab.

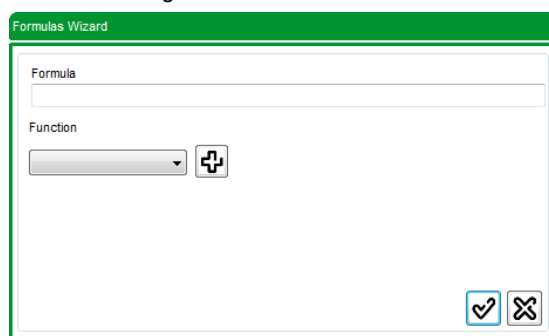
1. Create a signal in coreDb (for example, double-click on the field name in Status table)



2. In the field “Source1 Device” select “Formula”:



3. Select field “Source1 Coordinates” and right-click. Select Launch Point wizard:



This window allows using an easier way to write formulas.

Select a Function and a help will be shown:

Formulas Wizard

Formula

Function

DPSTOSPS

**DPSTOSPS(input)**

Where *input* is a dbValue defined by an expression. The function is a DPSTOSPS of this input so its return, another dbValue called as *output*, could be:

Input variable value (input.value)	Calculated value (output.value)	Quality returned (output.qFlag)	Timestamp returned (output.tSpec)
0	0	input.qFlag   IV_LQF	Same as input
1	0	input.qFlag	Same as input
2	1	input.qFlag	Same as input
3 or other	0	input.qFlag   IV_LQF	Same as input

**Table 6: DPSTOSPS function table**

✓ ✕

Add this function to the Formula using button .

If you need to use a coreDb point, you can copy its name from the corresponding tab and then paste it on the Formula field:

Easergy Builder V1.0.20 - Name: Project10. ID: Project10\_28/04/2016 13:17:01\_4

WorkSpace Save Send Configuration to RTU File View Help Add-Ons

Devices Channels coreDb Synchronization

Status Command Analog Setpoint dbNET

Name Source AND Destination

Name Description Source1 Device Source1 Coordinates

106 SC01\_XcSFPI... Cross-country tr... SC01 413 MIT

107 SC01\_LCCH1... FEA SPI Communi... SC01 414 MIT

108 SC01\_LCCH1... FEA SPI Communi... SC01 415 MIT

109 SC01\_LCCH2... FEA DC Communi... SC01 416 MIT

110 SC01\_LCCH2... FEA DC Communi... SC01 417 MIT

111 SC01\_CbrRR... Calculated break... SC01 1001 MMEC

112 SC01\_GenSF... PTOC instance n... SC01 1002 MMEC

113 SC01\_LLNO... Active settings g... SC01 1003 MMEC

114 SC01\_MainSS... Main switchgear... SC01 1006 MMEC

115 SC01\_LCCH... Main switchgear... SC01 1006 MMEC

116 PS01\_LCCH... Copy

117 PS01\_LPHD... Paste

118 PS01\_LPHD... Add Points from script

119 PS01\_AcZA... Add Points from textbox

120 PS01\_AcZA... Add Points from char separated text

121 PS01\_AcZA... Modify Points from script

122 PS01\_AcZA... Modify Points from textbox

123 PS01\_AcZA... Split in two signals

124 PS01\_LLNO... Configuration fault PS01 IS 4132

125 PS01\_STMP1... Overtemperature... DS01 RS 4133

Formulas Wizard

Formula

NOT

Funcio

NOT

NO

Wh

anc


expression. The function is a NOT of this input so its return,

Input variable value (input.value)	Calculated value (output.value)	Quality returned (output.qFlag)	Timestamp returned (output.tSpec)
0	1	input.qFlag	Same as input
1	0	input.qFlag	Same as input
2	1	input.qFlag	Same as input
3 or other	0	input.qFlag	Same as input

**Table 1: NOT function table**

✓ ✕

In order to complete the desired formula, you can write directly on the field, add other function from the Function field or copy the name of others coreDb points.

Finally, press button  or press Enter and your formula is added as coordinate for the new coreDb signal. This coordinate will be use as a Trigger or an Expression depending on if is a Source coordinate or a Destination coordinate.

#### 4.2.2.2 Triggers

When the formula Controller receives destination coordinate, they are called as triggers. A trigger definition has 2 or 3 field separated by colon (:).

In the next picture, we can see some examples of these triggers with source formulas waiting to be executed:

Name	Description	Source1 Device	Source1 Coordinates	Source1 Ymask	Destination1 Device	Destination1 Coordinates
st1		dnpSlave1	1:OB01:0		formula	st5.EVENT
st2		dnpSlave1	1:OB10:2		formula	st6:VAL:1
st3		dnpSlave1	2:OB03:1		formula	st7:VAL:0
st4		dnpSlave1	1:OB12:2		formula	st8.EVENT
st5		formula	st3-SPSTODPS(st1)			
st6		formula	2.3*st1+(-2)NOT(st4)			
st7		formula	IF(FORM_CYCLETIME>FORM_PERIOD,1)			
st8		formula	AND(st1,st10,MAX(2,FORM_CYCLETIME) < 10)			

Figure 4-7 – Example of triggered expressions and triggers for formula

For each coreDb point:

- A trigger is executed when its coreDb point gets a defined value or when is changed and generates an event. When these requirements happen, the coreDb point with a formula as source device, designed in the destination formula coordinate, is executed.
- If the trigger is set as destination coordinates, it waits an event on their coreDb point to satisfy the trigger type (EVENT or VAL). When it is satisfied, the source formula, of the coreDb point designed on the trigger destination coordinate, will be calculated. We have two triggers type:
  - EVENT: This trigger waits for a new event on its point, to make the trigger first field formula source calculated.
  - VAL: This trigger waits for an event with a determined value to make the trigger first field formula source calculated. This value is set as third field of the destination coordinate name.

A source coordinate formula, which is being triggered, is never executed until the trigger is fired.

An event happens, for example, if:

- Status points: There is a change on the value, quality flag or a crunch data was received with the mode KDM\_EVENT.
- Analog points: When is the first event received by this point, or if the difference between the new and the last analog values is higher than a determined value.
- Command or setPoints: There is a change on the value, quality flag or timestamp.

### 4.2.2.3 Expressions

When the formula Controller has source coordinates, they are called as expression.

The formula module's goal for source coordinates is to calculate a value depending on its input variables. Its value is updated at each execution of the Controller entry (the time between each entry execution is set as the shortest for the RTU system, approximately 20 ms depending on the platform).

At the first execution entry, the calculation and writing of results is mandatory for every valid expression without trigger. In the case of calculation, the value, quality flag and timestamp will depend on the initial values of coreDb points.

If an expression has associated a trigger (or triggers), its calculation will be never done until the trigger is fired.

In a division, if the second input is equal to zero, the calculation result will be equal to Not a Number and written once with this value and invalid quality flag in coreDb. If other expression uses this result on its calculation process, its result will be equal to NAN (0 at status or command), with invalid quality flag, while the precedence expression result is equal to NAN.

The expressions allowed to be calculated are detailed in the Appendix A at the end of this manual.

## 4.3 Channels

The ports used to communicate with field devices are configured as communication channels. They can be configured in the Channels tab of the configuration mode of Easergy Builder. The number and type of these channels depends on the type of CPU and communication modules installed in the RTU.

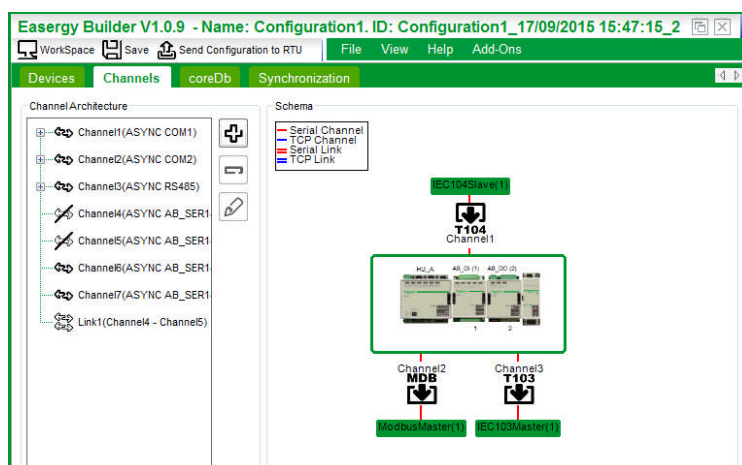


Figure 4-8 – Configuring communication channels


For example, this figure shows an RTU with a HU\_A and an AB\_SER module. The “Channel Architecture” zone displays all channels and links (channel associations) which are defined.

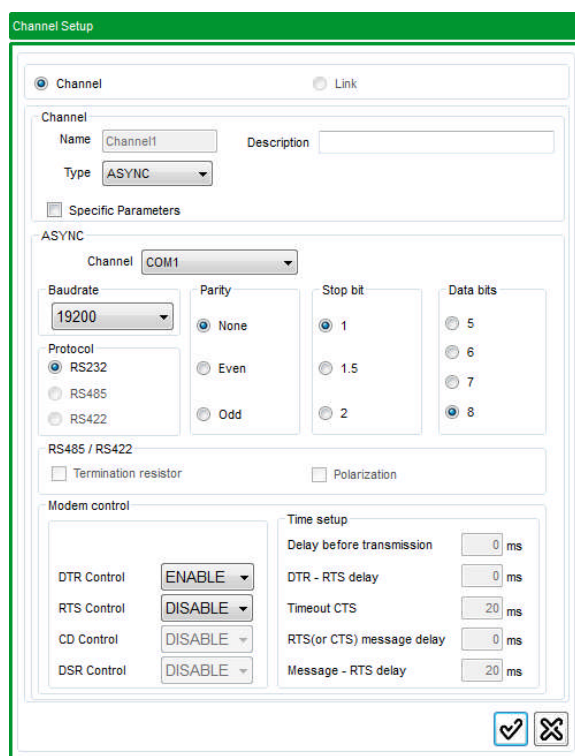
- Channel1 and Channel2: COM1 and COM2 (HU\_A)
- Channel3: RS-485 (HU\_A)
- Channel4, Channel5, Channel6 and Channel7: COM1, COM2, COM3 and COM4 (AB\_SER)

Using buttons in the toolbar, you can:

- Add a new channel or link.
- Delete a channel or link.
- Modify the configuration of a channel or link

### 4.3.1 Adding a Channel

Use button  to create a new communication channel:



The image shows a 'Channel Setup' dialog box with a green title bar. It has two tabs: 'Channel' (selected) and 'Link'. Under the 'Channel' tab, there are fields for 'Name' (set to 'Channel1') and 'Description'. Below these is a 'Type' dropdown menu set to 'ASYNC'. A checkbox for 'Specific Parameters' is present but unchecked. The 'ASYNC' section includes a 'Channel' dropdown set to 'COM1', a 'Baudrate' dropdown set to '19200', a 'Parity' section with radio buttons for 'None' (selected), 'Even', and 'Odd', a 'Stop bit' section with radio buttons for '1' (selected), '1.5', and '2', and a 'Data bits' section with radio buttons for '5', '6', '7', and '8' (selected). There is also a 'Protocol' section with radio buttons for 'RS232' (selected), 'RS485', and 'RS422'. Below this is an 'RS485 / RS422' section with checkboxes for 'Termination resistor' and 'Polarization', both unchecked. The 'Modem control' section has dropdown menus for 'DTR Control' (set to 'ENABLE'), 'RTS Control' (set to 'DISABLE'), 'CD Control' (set to 'DISABLE'), and 'DSR Control' (set to 'DISABLE'). The 'Time setup' section has input fields for 'Delay before transmission' (0 ms), 'DTR - RTS delay' (0 ms), 'Timeout CTS' (20 ms), 'RTS(or CTS) message delay' (0 ms), and 'Message - RTS delay' (20 ms). At the bottom right are two buttons: a checkmark and an 'X'.

Figure 4-9 – Channel configuration parameters

Where:

- **Name:** Identifying name.
- **Description:** Description.
- **Type:** Channel type depending on the implemented communication protocol. There are three types:
  - TCP
  - UDP
  - ASYNC (RS-232, RS-485 or RS-422 serial communication)
- **Specific Parameters:** Reserved, this box mustn't be checked.

Other fields depend on the type of channel.

### 4.3.1.1 TCP Channel

Figure 4-10 - TCP channel configuration.

- **Mode:** Type of communication trigger supported by this channel:
  - CALLED: The RTU acts as a “server”, accepts incoming tries of connection.
  - CALLING: The RTU acts as a “client”, tries to connect a remote server.
  - BOTH: Both are applicable.
- **Local Port:** Local TCP port associated to input messages. This field has no effect when the specified mode is "Calling".
- **Remote Port:** Client's TCP port associated to output messages. This field has no effect when the specified mode is "Called".

#### NOTICE

Some of the reserved ports are 2404 for IEC-104, 20000 for DNP 3.0 and 502 for Modbus TCP.

- **Remote IP List:** Client-associated IP addresses from which communication requests are accepted through this channel. If the list is empty, it would accept requests from any known client.

On a channel type CALLED, this is the address that will be validated after accepting the connection.

On a channel type CALLING, it will attempt to establish connection to the addresses specified in this list. Only after an unsuccessful try or disconnection, the following remote IP address in the list (if any) is used to the next try.

New clients can be added to the list by entering its IP in the field over the list and pressing the “+” button. To remove a client from the list, select it and press the “-” button.

- **Connect timeout:** For CALLING channels. Timeout (in milliseconds) waiting an answer for a connection.
- **Reconnect time:** For CALLING channels. Minimum time to wait for connection retry.
- **Use local IP:** It is optional. If is checked, IP field specifies the only local IP address to be used in the TCP connection.

### 4.3.1.2 UDP Channel

If selecting UDP, the following information can be specified:

Figure 4-11 - UDP channel configuration.

- **Remote IP List:** The list of remote IPs is optional and indicates the addresses from which UDP packets are accepted through the local port. Messages will be always transmitted to the IP and port of the last received UDP packets. Until the reception of the first UDP packet, UDP packets are sent to the first IP address of the list and the Remote Port.
- **Local Port:** Port where to receive UDP packets from. Mandatory and no zero.
- **Remote Port:** Remote UDP port to send UDP packets to, until the reception of the first UDP packet. Optional.
- **Use local IP:** It is optional. If selected, IP specifies the only local IP address to be available to receive and send UDP packets.



New clients can be added to the list by entering its IP in the field over the list and pressing the “+” button. To remove a client from the list, select it and press the “-” button.

### 4.3.1.3 ASYNC Channel

Figure 4-12 - ASYNC channel configuration.

- **Channel:** The physical port that will be used for this channel. The list of physical channels available depends on the CPU type:
  - SM\_CPU866 and SM\_CPU866e: COM1, COM2, COM3, COM4 and SM\_SERx-COMy. Only ports of the SM\_SER configured will be available.
  - HU\_A and HU\_AF: COM1, COM2, RS-485 and AB\_SERx-COMy. Only ports of the AB\_SER configured will be available.
  - HU\_B: COM1 and COM2.
  - HU\_BI: COM1/RS-485. Only one of these ports can be used at the same time.
  - HU 250: RS-485, K7 RS SLOT(1) and K7 RS SLOT(2). K7 RS SLOT 1 and 2 will be available or not depending on the configuration of the slots 1 and 2.
- **Baudrate:** To specify the communication speed. The speed ranges between 300 and 256000 bps.
- **Protocol:** It defines the asynchronous protocol which will be used. The available protocols are RS-232, RS-485 and RS-422. RS-485 allows 2 or 4-wire communications.
- **Parity, Stop bit, Data bits:** To configure communication parameters.
- **RS-485 / RS422:**
  - Termination resistor
  - Polarization
- **Modem control:** These parameters allow configuring the signals for modem control in the communication ports. Not all values are available for each CPU
  - **DTR control (Data Terminal Ready):** Flow control:
    - ENABLE: DTR at high logical level (1).
    - DISABLE: DTR at low logical level (0).
    - TOGGLE (only available for K7 modem): DTR control depends on the parameter “DTR - RTS delay”.
  - **RTS control (Request to Send):** To configure the RTS output:
    - DISABLE. Disables the use of the RTS signal.
    - ENABLE: It enables the RTS signal, and keeps it active.
    - AUTO: The RTS signal timing will be defined automatically.
    - TOGGLE: It allows defining timing for RTS signal.

- **CD Control (Carrier Detect):**
  - ENABLE: CD at high logical level (1).
  - DISABLE: CD at low logical level (0).
- **DSR Control (Data set Ready):**
  - ENABLE: DSR at high logical level (1).
  - DISABLE: DSR at low logical level (0).

The time setup when DTR is configured as TOGGLE:


- **Delay before transmission:** Elapsed time from the data transmission is ready and activation of the RTS.
- **DTR – RTS delay:** Timeout before establish the RTS signal when DTR has been activated.
- **Timeout CTS:** Standby time from the activation of the RTS to the activation of the CTS. If the CTS value is zero, it means that the transmission will be done regardless of CTS value. For non-zero values, channel transmission is controlled by the CTS; if the time defined in this attribute elapses without a CTS activation, then the package pending to be sent is discarded.
- **RTS (or CTS) message delay:** Elapsed time from the activation of the CTS to data transmission.
- **Message – RTS delay:** time from the end of the data transmission to RTS deactivation.

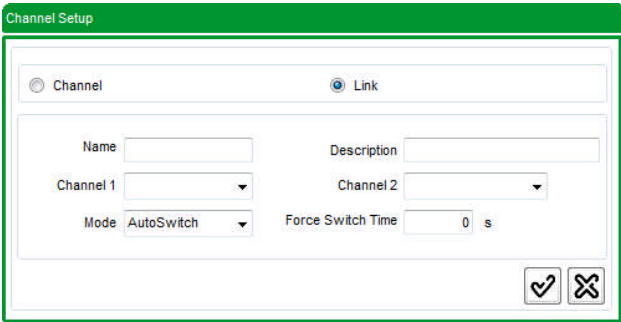
## NOTICE

For 2-wire RS-485 communications, it is very often necessary to set the RTS control to TOGGLE, and define all times to 0 ms. Thus, reception while transmitting is disabled (to avoid echo).

### 4.3.2 Adding a Link

Some devices support double channel. The functionality will be different for each protocol. A Link is an association of two channels and it can be used to identify a double channel.

Use button  to create a new communication link:





The image shows a 'Channel Setup' dialog box with a green title bar. Inside, there are two radio buttons: 'Channel' and 'Link'. The 'Link' radio button is selected. Below these are two columns of input fields. The left column has 'Name' and 'Channel 1' (a dropdown menu). The right column has 'Description' and 'Channel 2' (a dropdown menu). Below these columns are two more fields: 'Mode' (a dropdown menu set to 'AutoSwitch') and 'Force Switch Time' (a text box with '0' and a unit 's'). At the bottom right of the dialog are two buttons: a checkmark icon and a close 'X' icon.

Figure 4-13 – New link.

Where:

- **Name:** Link name.
- **Description:** Description.
- **Channel 1:** First channel to associate.
- **Channel 2:** Second channel to associate. It is not mandatory.
- **Mode:** There are two options:
  - **AutoSwitch:** In the slave, channel module is configured to switch channels automatically. The slave receives and transmits through a channel. At any moment, if it stops receiving through this channel, it switches to the other, which becomes active.
  - **SwitchByMaster:** In the master, this option defines the Device to control switching.
- **Force Switch Time:** A periodic switching between channels can be defined for the item "Mode → SwitchByMaster". Master protocols must switch the channel to verify state every TIME\_FORCE\_SWITCH seconds. When the Link has been set to AUTO\_SWITCH mode, this value will be not considered. When SwitchByMaster has been defined, it is recommended to set this parameter to a non-zero value when the module is able to indicate (supervision points) the channel's error state.

### 4.3.3 Deleting and Editing a Channel or Link

Select the channel or link in the tree and use button  to remove it. Select the channel or link in the tree and use button  to change the data associated to it.

## 4.4 Synchronization

The synchronization can be configured in the tab Synchronization of the configuration mode of Easergy Builder. This functionality offers a wide range of capabilities to synchronize the RTU.

The screenshot shows the 'Synchronization' configuration window in Easergy Builder V1.0.9. The window title is 'Easergy Builder V1.0.9 - Name: Configuration1.ID: Configuration1\_17/09/2015 15:47:16\_2'. The 'Synchronization' tab is selected. The 'Synchronization period' is set to 30 s. The 'Primary Device' is set to GPS, with a 'Timeout' of 0 s, 'GPS Port' of COM1, and 'GPS Type' of GPS16HVS and compatibles. The 'Secondary Device' is set to IRIG, with a 'Timeout' of 0 s, 'FORMAT' of B, and 'CODE' of 003. The 'Synchronization server' section has 'SNTP' selected, with 'Mode' set to PASSIVE and 'Frequency' of 10 s. The 'Time Configuration' section has 'Summer Time' checked, 'Local time Zone' set to +1, and 'Start Day/End Day type' set to Relative Day/Relative Day. The 'Relative Start Date' is set to 03 Sunday 02 Last, and the 'Relative End Date' is set to 10 Sunday 03 Last.

Figure 4-14 – Synchronization configuration

You can configure up to two synchronization devices: Primary and Secondary.

### NOTICE

With Saitel modules, the console tool always can be used as a default synchronization device (with the minimum priority).

The following console command could be used:

- **thmShow**: Shows the status of the synchronization devices. You can see information about the actual time too.
- **thmConsoleSetTime “YY:MM:DD:HH:NN:SS”**: Manual synchronization of the CPU.

The RTU can be configured as an SNTP server and as a PTP master.

Depending on the CPU type, the synchronization can be performed by:

- **Protocol**: Most telecontrol protocols allow slave devices to synchronize.
- **SNTP**: The synchronization module includes a SNTP client and server, which can be used to synchronize from a network SNTP clock or as a time reference for other modules.
- **IRIG** (Only available for HU\_A, HU\_AF and SM\_CPU866e). It synchronizes IRIG-B-compliant devices.
- **GPS** (Not available for HU 250. Both Saitel DP and Saitel DR allow direct connection to a GPS for time synchronization): The following devices have been validated as GPS: GPS35, GPS16, TSU (Telvent) with protocol NMEA and TKR2 with protocol PTAREE.
- **PTP** (Only available for SM\_CPU866e). As indicated in the IEEE-1588 standard, a PTP master can synchronize other PTP devices (slaves) through one or several Ethernet interfaces.

The synchronization module allows the time zone and summer/winter (day light saving) calendars to be configured.

In this window there are two main zones:

- **Synchronization:** Parameters about the synchronization source.
- **Time Configuration:** Parameters related to the processing of the synchronization data from the source.

### 4.4.1 Configuring a Synchronization Device (as Source)

For Primary and Secondary Device, depending on the synchronization method selected you need to define a set of fields for each device (PROTOCOL, SNTP, IRIG, GPS or PTP).

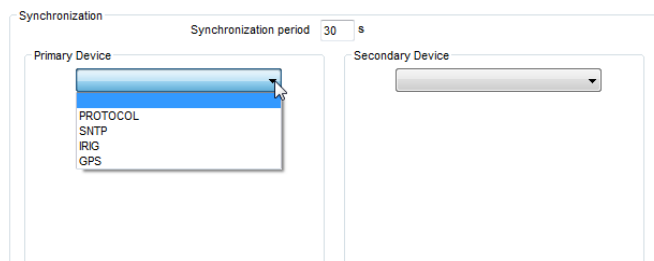


Figure 4-15 – Configuration of synchronization devices

- **Synchronization period** is the value (in seconds) of the synchronization period. All devices (Primary Device y Secondary Device) are scanned at the interval defined in "Synchronization period" and the system connects to the top priority device.
- **Timeout:** (Only visible when you select a type of device) Time in seconds to mark the device as "off-line" If no synchronization messages are received.

If Primary Device and Secondary Device are considered enabled (Primary Device is the online device synchronization) and Timeout time elapses without receiving synchronization from the Primary Device, then the Secondary Device becomes the synchronization device. If secondary device also times out, then the active synchronization source will be the console.

#### 4.4.1.1 Synchronization Device: PROTOCOL

When a telecontrol protocol is configured as synchronization source, you have to define:

- **Device Sources:** It allows selecting the synchronization sources from the existing devices or from a point into the table analog or status. You can select more of one Device as source.

#### 4.4.1.2 Synchronization Device: SNTP

Synchronization using SNTP can be in active or passive mode.

In active mode, the SNTP client asks to a server for the time, then, you have to configure each SNTP server available as synchronization source. Please, add a Server (using "Add Server" button) for each one of them.

- **SNTP Server IP:** SNTP server's IP address.
- **Period:** Time (in seconds) for a new interrogation to this server.

Only one of them is used as active server for each moment. Only this server will be asked for the client while it is answering. If this server doesn't answer to the client, the following server in the list will be marked as active.

In passive mode (mark "**Passive**") the client is synchronized when receive a message from a server. In this case, any server has to be configured.

#### 4.4.1.3 Synchronization Device: IRIG

Synchronization using IRIG-B is only available for HU\_A, HU\_AF and SM\_CPU866e. You can configure the following formats (consult the standard IRIG STANDARD 200-04): IRIG-B002, IRIG-B003, IRIG-B006 and IRIG-B007. Information to be defined for this type of device includes:

- **FORMAT:** Selection of a format, supported by IRIGB, in which the time data will be managed (A, B, D, E, G, or H). This version only admits the format 'B'.
- **CODE:** The value selected in this field together with the previous item will complete the configuration of the IRIGB streaming. You can choose from 002, 003, 006 and 007, which are the formats recognized by the IRIG-B 200-98 standard.

#### 4.4.1.4 Synchronization Device: GPS

When a GPS is defined as synchronization source you have to configure:

- **GPS Port:** Serial port used to connect the GPS.
- **GPS Type:** It allows choosing a GPS from a list. One of the two last options must be selected if synchronizing by a Schneider Electric's TSU module; the rest of the options are available if synchronizing by a Schneider Electric's MSAC module or directly from the GPS. Available GPS include:
  - GPS 16HVS and compatible devices. This is the adequate option when synchronizing through MSAC.
  - GPS TKR2.
  - TSU-G1 (IRIG-B).
  - TSU-G2 (GPS).
- **PPS:** It indicates if the PPS signal is taken from the GPS.

#### 4.4.1.5 Configuring the RTU as SNTP and/or IRIG Server

The RTU can be configured as a SNTP server and it could be used to synchronize other SNTP slave devices.

The screenshot shows the 'SNTP' tab selected. The 'SNTP Server' checkbox is checked. The 'Mode' dropdown is set to 'ACTIVE'. The 'Frequency' is set to '10' seconds. Under the 'Destination' section, the 'Device' radio button is selected, and a text box for the device name is visible. The 'Broad...' radio button is also present but not selected.

Figure 4-16 – Configuring the RTU as SNTP server

Where:

- **SNTP Server:** Allows configuring the RTU as a SNTP server.
- **Mode:** Can be ACTIVE or PASSIVE. If PASSIVE mode is selected, the server only will answer when a request is received from a SNTP client. If ACTIVE mode is selected, the server sends (periodically) a synchronization broadcast message. In ACTIVE mode the server will answer too each request from a client.
- **Device or Broadcast IP:** Only available in ACTIVE mode. Ethernet port or IP address where the server will send the synchronization broadcast message.
- **Frequency:** Only available in ACTIVE mode. Time (in seconds) between broadcast messages.

If you need set the RTU as an IRIG server, please, select IRIG tab and configure the IRIG, check IRIG Server and select the format (only B is available), and the code used (IRIG-B002, IRIG-B003, IRIG-B006 and IRIG-B007).

The screenshot shows the 'IRIG' tab selected. The 'IRIG Server' checkbox is checked. The 'Format' dropdown is set to 'B'. The 'Code' dropdown is set to '003'.

Figure 4-17 – Configuring the RTU as IRIG server

#### 4.4.2 Time Configuration

The following information defines the timetable:

The screenshot shows the 'Time Configuration' window. The 'Local time Zone' is set to '+1' hours and '0' minutes. The 'Summer Time' checkbox is checked. The 'Start Day/End Day type' section has four radio buttons: 'Day/Day', 'Day/Relative Day', 'Relative Day/Day', and 'Relative Day/Relative Day' (which is selected). There is also a 'Custom' radio button. Below this, there are two sections: 'Relative Start Date' and 'Relative End Date'. Each section has a table with columns: Month, Week Day, Hour, and Week. For 'Relative Start Date', the values are 03, Thursday, 02, and Last. For 'Relative End Date', the values are 10, Thursday, 03, and Last.

Figure 4-18 – Configuring the RTU as SNTP server

- 
- **Local Time Zone:** It allows configuring the time zone. H and M indicates the difference (H: hours and M: minutes) in the geographic zone with the GMT.
  - **Summer Time:** It enables the summer time option. This configuration can be defined (depending on each country) using relative or absolute dates. You have to configure the initial and final date of the summer time. Example: For example, in Europe, summer time starts the last Sunday of March and finishes the last Sunday of October.

# Chapter 5 - coreDb - Real Time DataBase

## 5.1 Introduction

This chapter describes in detail the real-time database of the BaseLine platform, as well as the administration tasks which can be performed in the database using Easergy Builder.

In the Configuration mode, select tab coreDb:

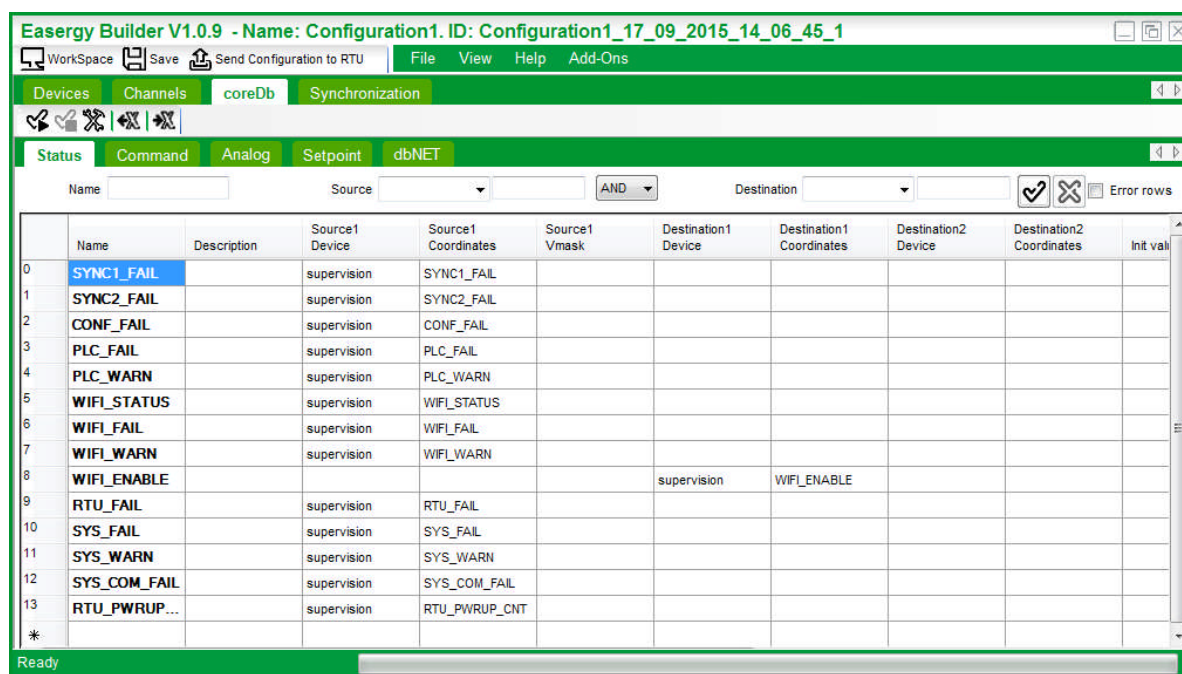


Figure 5-1 - coreDb administration

## 5.2 Main Menu

coreDb is the real-time database of the BaseLine Software Platform for RTUs. This database stores all the information associated to the points and the relationships with the I/O information managed by defined Devices. These relationships are implemented through the source and destination allocations.

All this information is arranged in different tables. Each table stores a different type of point (Status, Analog, SetPoint and Command tables)

## 5.3 coreDb Tabs

Selecting a tab, you can access data points defined for each coreDb table.

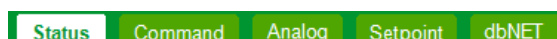


Figure 5-2 - coreDb main menu.

Information will be displayed in table format. For example, if Status is selected, the status table will be shown:




Name	Description	Source1 Device	Source1 Coordinates	Source1 Vmask	Destination1 Device	Destination1 Coordinates	Destination2 Device	Destination2 Coordinates	Init value	Blocked	Non volatile	Shared Publish	Shared Subscribe
0	SYNC1_FAIL	supervision	SYNC1_FAIL							<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
1	SYNC2_FAIL	supervision	SYNC2_FAIL							<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	CONF_FAIL	supervision	CONF_FAIL							<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3	PLC_FAIL	supervision	PLC_FAIL							<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4	PLC_WARN	supervision	PLC_WARN							<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5	WIFI_STATUS	supervision	WIFI_STATUS							<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
6	WIFI_FAIL	supervision	WIFI_FAIL							<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
7	WIFI_WARN	supervision	WIFI_WARN							<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
8	WIFI_ENABLE				supervision	WIFI_ENABLE				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
9	RTU_FAIL	supervision	RTU_FAIL							<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
10	SYS_FAIL	supervision	SYS_FAIL							<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
11	SYS_WARN	supervision	SYS_WARN							<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
12	SYS_COM_FAIL	supervision	SYS_COM_FAIL							<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
13	RTU_PWRUP...	supervision	RTU_PWRUP_CNT							<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
*										<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure 5-3 - status table in coreDb.

This window shows all status points in coreDb. Each point has an associated name, description, producers (sources) that will update the point's information (value, timestamp, quality flags, etc.) and the consumers (destinations) that will retrieve the information.

## 5.4 Automatic Generation of coreDb points

When you are creating a configuration, you can mark the field "Create defined RTU acquisition points" when you are creating an RTU and then, all supervision and I/O points for the default configuration will be created in coreDb.

On the other hand, when you are configuring the local acquisition, you can use button  in order to generate automatically all acquisition data points in coreDb.

Input	DI_LSM	Invert	TF	TM
<input checked="" type="checkbox"/> PROFI_STS 1	DI_LSM	Invert	TF 10	TM 0
<input checked="" type="checkbox"/> PROFI_STS 2	DI_LSM	Invert	TF 10	TM 0
<input checked="" type="checkbox"/> PROFI_STS 3	DI_LSM	Invert	TF 10	TM 0
<input checked="" type="checkbox"/> PROFI_STS 4	DI_LSM	Invert	TF 10	TM 0
<input checked="" type="checkbox"/> PROFI_STS 5	DI_LSM	Invert	TF 10	TM 0
<input checked="" type="checkbox"/> PROFI_STS 6	DI_LSM	Invert	TF 10	TM 0
<input checked="" type="checkbox"/> PROFI_STS 7	DI_LSM	Invert	TF 10	TM 0
<input checked="" type="checkbox"/> PROFI_STS 8	DI_LSM	Invert	TF 10	TM 0

Figure 5-4 – Configuring local acquisition.

## 5.5 Basic Operations with coreDb

### NOTICE

Avoid the use of the letter "ñ" and vowels with accents in the field Name. This applies throughout the document.



## 5.5.1 Point Management

If the user right-clicks on the field Name of one or more selected points, the following menu will appear:

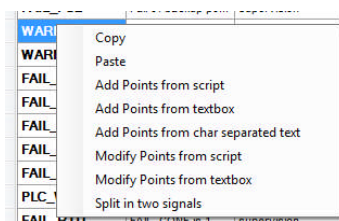


Figure 5-5 – Contextual menu for a selected point.

If the user right-clicks on the Name field of a non-selected cell, the menu will be:

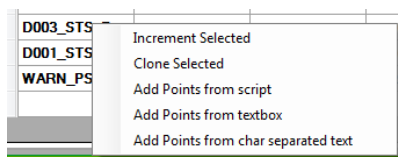


Figure 5-6 – Contextual menu for a new point.

If the user right-clicks on the Name field of a blank register, the menu will be:

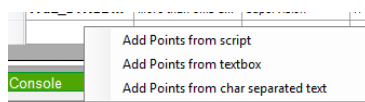


Figure 5-7 – Contextual menu for a new point.

- **Copy:** The point selected is copied to the Windows® clipboard.
- **Paste:** The content is copied from the Windows® clipboard.
- **Add point from script:** It allows adding a series of names with a common part and a variable one. Variables: [X], [Y] and [Z] in this priority order. Example: by entering ST[XXX]\_[Y]A[X], first "X" (X0 on Figure 5-8) is numeric from 1 to 2, "Y" (Y0 on the Figure 5-8) is list type with values "a, b and c", and second "X" (X1 on the Figure 5-8) is numeric from 3 to 4. This example generates the following points: ST001\_aA3, ST001\_bA3, ST001\_cA3, ST002\_aA4, ST002\_bA4 and ST002\_cA4.

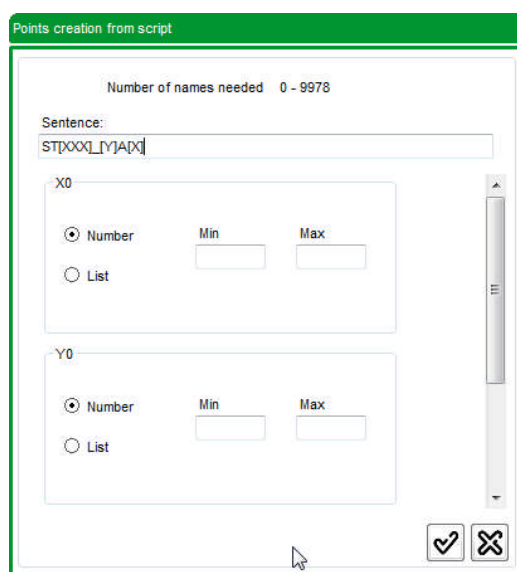


Figure 5-8 - Add point from script.

The configuration area will change while the user is writing the sentence.

- **Add point from textbox:** It allows entering the names in a text box.

Figure 5-9 - Add point from textbox.

- **Add point from char separated text:** The file with the name to add is specified, separated by a character, which is entered in the “**Separation Character**” field.

Figure 5-10 - Add point from separated text.

- **Modify Points from script:** All selected points will be modified using a series of names with a common part and a variable one. The number of the generated points by the script must be the same that selected points.
- **Modify Points from textbox:** All selected points will be modified using a list of points indicated in a text box.
- **Split in two points:** Duplicates the selected point. This option is used to create two simple points from one double. For example, this is useful when transferring points to ISaGRAF.

Name	Description	Source1 Device	Source1 Coordinates	Source1 Vmask	Destination1 Device	Destination1 Coordinates	Destination2 Device	Destination2 Coordinates	D
D001_00000	PROFI_STS 1	laq	2001020000		status	D001_00000_1	status	D001_00000_2	D
D001_00000_1		status	D001_00000	0x01	Isa	D001_00000_1:B			
D001_00000_2		status	D001_00000	0x02	Isa	D001_00000_2:B			

Figure 5-11 – Transferring a double point in two single points to ISaGRAF

- **Clone Selected:** The content in this cell is copied on the selected cell.
- **Incremented Selected:** The content in this cell is incremented and copied on the selected cell.


## 5.5.2 Search

Each tab corresponding to coreDb has a set of buttons with a common functionality:

Figure 5-12 - Search bar of a coreDb table

The information to be completed in this bar is:

- **Name:** Name of the point to search, if it is known and only one. If it is completed with a point name which doesn't exist, nothing is returned.
- **Source:** The drop-down box is used to select points which have a specific device as their source. The next field is to search a point by its source coordinate.
- **Destination:** The drop-down menu is used to select points that have a specific device as their destination. The next field is to search a point by its destination coordinate.
- **"AND" "OR":** In the case of fill the Source and Destination fields, this drop-down menu is used to combine the other two filters to refine the search on the table shown.
- **Error rows:** By checking this box only points that are badly configured will be shown.

Press button  to apply the defined search filter.

### 5.5.3 Configuring Data Point Value

The following table shows the information associated to each point depending on the coreDb table. Each letter represents a table in coreDb, where **S**:status, **A**:analog, **C**:command and **P**:setPoints. A mark (✓) means the column is into the table:

Column	Table in coreDb				Description
	S	A	C	P	
Init Value	✓	✓			Point's initial value.
Blocked	✓	✓			If it is checked, the value is allocated manually. All its sources are ignored.
Non Volatile	✓	✓	✓	✓	If it is checked, the point value is stored in NVRAM (non-volatile RAM) to avoid an information loss during a power-off and an RTU reboot. This box is ignored in the initial database load and the point will obtain the "INIT VALUE". (HU250 doesn't support this feature).
Shared Publish	✓	✓			The point's value can be accessible for other RTUs connected to the control system. If this column is checked, the point is stored in the RTU database and will be accessible by the rest of the system. If this case, information in tab <b>dbNET</b> must be complain.
Shared Subscribe	✓	✓			If it is checked, SOURCE columns in this point must be empty. The point is stored in an external database (in other RTU). In this external database, this point must be defined as publish.
Engineering units Convert		✓			It allows a unit conversion for the information source. If it is checked, the following four fields are available to define the conversion straight.
Engineering units Min raw		✓			Minimum value of the straight entrance (RAW). Only available if "Engineering units Convert" is checked.
Engineering units Max raw		✓			Maximum value of the straight input (RAW). Only available if "Engineering units Convert" is checked.
Engineering units Min Egu		✓			Minimum value of the straight output (EGU). Only available if "Engineering units Convert" is checked.
Engineering units Max Egu		✓			Maximum value of the straight output (EGU). Only available if "Engineering units Convert" is checked.
Alarm Use		✓			It allows define up to four types of alarms to be set which are triggered when they cross a defined threshold value. These limits are defined with the following four fields (Lowest limit, Low limit, High limit, and Highest limit).
Alarm Lowest limit		✓			Lowest limit associated to this point. If it is crossed, an alarm is triggered, an event is generated and sent to all destinations and QF (quality bit) = <b>0x00000800</b> . This field is only available if "Alarm Use" is checked.

<b>Alarm Low limit</b>		✓			Low limit associated to this point. If it is crossed, an alarm is triggered, an event is generated and sent to all destinations and QF (quality bit)= <b>0x00000400</b> . This field is only available if "Alarm Use" is checked.
<b>Alarm High limit</b>		✓			High limit associated to this point. If it is crossed, an alarm is triggered, an event is generated and sent to all destinations and QF (quality bit) = <b>0x00000200</b> . This field is only available if "Alarm Use" is checked.
<b>Alarm Highest limit</b>		✓			Highest limit associated to this point. If it is crossed, an alarm is triggered, an event is generated and sent to all destinations and QF (quality bit) = <b>0x00000100</b> . This field is only available if "Alarm Use" is checked.
<b>DESTINATION THRESHOLD</b>		✓			For Analog points, it is possible to configure a change threshold for each destination. This threshold allows all destinations to update their value by events instead of by polling

## NOTICE

- Engineering units limits are only available if "**Engineering units Convert**" is checked.
- Alarm limits are only available if "**Alarm Use**" is checked.

## 5.5.4 Source and Destination Allocations

Each coreDb point must be associated with source(s) or destination(s).

Source(s) and destination (s) for a point are allocated by selecting coordinates from one or multiple Devices (which must be previously defined in Easergy Builder).

It should be noted that two points cannot have the same source or destination. However, it is possible to use a point declared as source or destination in coreDb, but considering the followings factors:

## NOTICE

By default, there are two Devices defined in order to use coreDb points as sources. They are called "**status**" and "**analog**" and are included in all Configurations by default.

To use a coreDb point as source, this point must have the point(s) which will use the coreDb point as destination; the status or analog Device and the point defined as source will be the source of these destination points.

Following section illustrates this consideration.

Information points can be selected graphically. Right-click on the Source or Destination Device and select "Select Device" or select the Device and right-click on the coordinate field and select "**Launch Point wizard**". The displayed window will depend on the Device selected.

For example, the following window will appear for the supervision Device in a Saitel DP RTU:

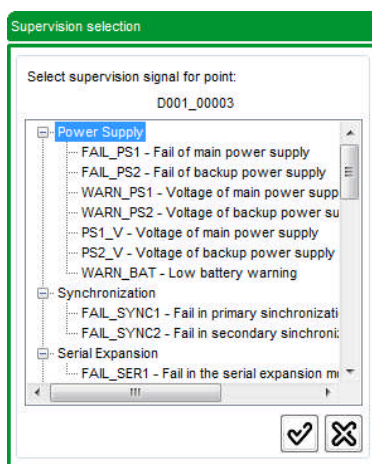


Figure 5-13 - Selecting a supervision point as source.

This window shows all points for the selected Device, which are not used yet. Select any point, and then click the which green button (assuming it's the 'tick').

For more information about how to assign Device points as destination and/or source of a coreDb points, please refer to the appropriate Device controller manual. In order to use the local acquisition points, please refer to the manual "Configuration and Startup of ..." for the hardware platform.

## 5.6 Data Point Configuration

### 5.6.1 Quality Flags

Every information point in coreDb is associated to a set of bits which provide information about the point. These quality flags have been generated to prevent the loss of information, and we can find two types:

- **Remote quality flags:** They reflect the value of the points sent by the information source which represents the point.
- **Local quality flags:** They reflect the value of the point considering only its treatment in the device.

The format of this quality flags or mask is: **0x[hexadecimal number]**.

Local Quality Flags	
Values (hexadecimal)	Description
0x00000001	An overflow occurred.
0x00000002	An overflow or roll – over occurred in a counter.
0x00000004	The counter has been adjusted.
0x00000008	Excessive changes in a digital input.
0x00000010	Blocked point.
0x00000020	Manual point replacement.
0x00000040	The point has not been written to the database yet.
0x00000080	Invalid point.
0x00000100	The point value has exceeded Hi-Hi Alarm.
0x00000200	The point value has exceeded Hi Alarm.
0x00000400	The point value is lower than Lo Alarm.
0x00000800	The point value is lower than Lo-Lo Alarm.
0x00001000	Invalid time.
0x00002000	An override occurred.
0x00004000	A forced override occurred.
Quality Flags from the Device	
Values (hexadecimal)	Description
0x00010000	An overflow occurred.
0x00020000	An overflow or roll – over occurred in a counter.
0x00040000	The counter has been adjusted.
0x00080000	Excessive changes in a digital input.
0x00100000	Blocked point.
0x00200000	Manual point replacement.
0x00400000	The point has not been written to the database yet.
0x00800000	Invalid point.
0x10800000	Invalid time.

Table 5-1 – Quality flags in coreDb points..

## 5.6.2 Status Configuration

Apart from the basic settings shown in chapter 5.5, coreDb status data points support additional settings. To configure these, Easergy Builder has to be working in configuration mode and the "Status" tab has to be selected inside the coreDb view.

### NOTICE

Even though several masks can be defined to each source associated to the point, only the last association will be applied.

The following examples illustrate this functionality:

#### Example 1 - Using masks to create two simple points from one double

This is useful when transferring points to ISaGRAF. The following example shows how to change from a double point from the local acquisition (Saitel DR local acquisition Device) to two simple ISaGRAF points, using the mask.

Status Command Analog Setpoint dbNET								
Name		Source		AND		Destination		
	Name	Source1 Device	Source1 Coordinates	Source1 Vmask	Destination1 Device	Destination1 Coordinates	Destination2 Device	Destination2 Coordinates
0	Ddouble	claq	1003020000		status	DdobA	status	DdobB
1	DdobA	status	Ddouble	0x01	Isagraf	DdobA:B		
2	DdobB	status	Ddouble	0x02	Isagraf	DdobB:B		

#### Example 2 - Using masks to create two simple Modbus points from one double IEC104 point

The mask is also applied to coreDb points to map the quality bits corresponding to another point in a database point (by applying a mask).

Status Command Analog Setpoint dbNET								
Name		Source		AND		Destination		
	Name	Source1 Device	Source1 Coordinates	Source1 Vmask	Destination1 Device	Destination1 Coordinates	Destination2 Device	Destination2 Coordinates
0	Ddouble104	IEC104master	15000:MDP		status	DdobMDBA	status	DdobMDBB
1	DdobMDBA	status	Ddouble104	0x01	MDBe	IS:1		
2	DdobMDBB	status	Ddouble104	0x02	MDBe	IS:2		

#### Example 3 - Using the mask to map quality bits

The suffix ":Q" in the mask indicates that it is applied to quality bits. The value is written in hexadecimal format. Since the points are composed of 32 bits, the mask has eight digits. In the mask, the value of the mapped bits is 1. If fewer digits are written, they will be the least significant and their value will be 0 ("0x00000001:Q" equals "0x0001:Q").

If all bits for the value are 0, then all are mapped, that is, "0x00000000:Q" equals "0xFFFFFFFF:Q". If the mask is "0x00F0:Q", bits 4 to 7 are taken, as if there are no characters on the right.

Status Command Analog Setpoint dbNET						
Name		Source		AND		Destinat
	Name	Source1 Device	Source1 Coordinates	Source1 Vmask	Destination1 Device	Destination1 Coordinates
0	Dinfo	IEC101master	01000:MSP		status	DinfoQ
1	DinfoQ	status	Dinfo	0x00000001:Q	Isagraf	DinfoQ:B

## 5.6.3 Analog Configuration

For the information source (Source) it is also possible to define a mask. The mask is also applied to map the quality bits corresponding to another point in a database point (by applying a mask). The explanation for the points in Status table also applies here.

### NOTICE

To map the quality bits of an Analog point it is necessary to set up as destination of this point another of the same table. Though only a bit is mapped, it must be stored in a point of the Analog table.

## Example 1 - Using the mask to map quality bits.

Status	Command	Analog	Setpoint	dbNET		
Name <input type="text"/> Source <input type="text"/> AND <input type="text"/> Destination <input type="text"/>						
	Name	Source1 Device	Source1 Coordinates	Source1 Vmask	Destination1 Device	Destination1 Coordinates
0	Dinfo	IEC101master	13000:MMEA		analog	DinfoQ
1	DinfoQ	analog	Dinfo	0x0001:Q	Isagraf	DinfoQ:B

## Example 2 – Using the Field DESTINATION THRESHOLD.

A data point corresponds with a temperature measurement. The destination Device is a slave protocol. The master protocol allows data to be sent by event and performs an integrity update on an hourly basis. The change threshold is adjusted to 10 (degrees, assuming that the engineering units are degrees). coreDb will generate events when the temperature is 0, 10, 20, 30, 40... degrees.

## 5.6.4 Command and Setpoint Configuration

From coreDb's main view, the user can select the Command tab to configure command points and Setpoint tab to configure setpoints. There are no special considerations for the configuration of any of them.



## 5.6.5 Other Operations with coreDb


In the coreDb configuration view, the following buttons are available:



Using these buttons you can:

- Check information in coreDB
- Configure if the field "Description" is loaded or not on memory for each table.
- Import / Export coreDb information using Excel files.

Press button  to check the information in coreDb before sending it to the CPU. Press button  to stop the current check.

Button  allows you configure for each table if the field Description is loaded on memory or not.

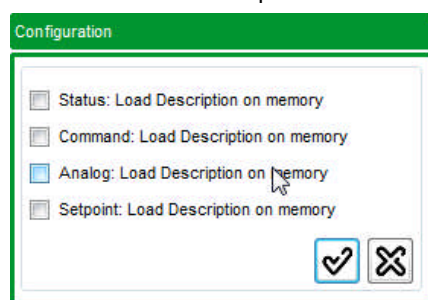



Figure 5-14 – Use of the field "Description".

Button  allows you import databases from an Excel® file to the coreDb of the configuration. Press this button and select the file with the information to be imported.

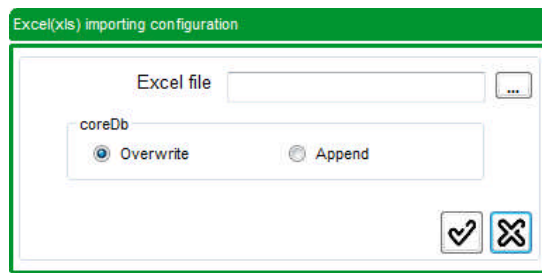


Figure 5-15 – Using Excel® for data importing.

You have to select if the information to import will overwrite the information in coreDb or if it will be added. If the chosen Excel file does not exist, the importing process is cancelled and an error message appears.

## NOTICE

To perform a successful data import, no empty rows can be found in the top or middle rows of a table.

During the importing process, the tool performs pre-validations to incorporate the points to coreDb. The tool verifies that the point's attributes comply with database rules:

- Field length
- Allowed characters.
- Valid value range.

This tool also validates the coordinates of each Device point, defined as source or destination for another point. These verifications can also be performed as explained in the previous section.

In the Log console information messages about the process are shown.

An import is successful if all points in the Excel file have been seamlessly inserted in coreDb. Should there be an error due to the validation criteria, this point will not be inserted in the database and the console will display an error message. For example:

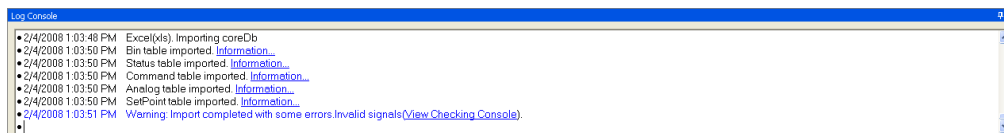



Figure 5-16 - Data importing process with error reporting.

Some cases where this happens include:

- If the source or destination is correct but the Device hasn't been defined in the configuration, the point is imported, leaving the field Device empty and completing the Coordinate field. The table row will display an error.
- If the source or the destination is complete, but the coordinate is syntactically incorrect, the point will not be imported, and a message will appear in the console.
- If the source or destination's coordinate is empty, the point will not be imported, and a message will appear in the console.
- If the source or destination's Device is empty, the point will not be imported, and a message will appear in the console.

Button  allows exporting the information stored in coreDb to an Excel file. This file can be used in order to import data in other configurations, as explained in previous section.

## 5.7 Sharing Database Information between RTUs

The tab dbNet in coreDb section allows multiple Saitel RTUs to share their databases within an IP network. dbNET operates on a broadcasting basis. This proprietary protocol has been optimized to let the RTUs within the same network share a limited number of database points.



The requirements to share data include:

- The RTU that shares data needs to define a coreDb point with a name and the **"SHARED PUBLISH"** checkbox needs to be enabled.
- The RTU that acquires the shared data point needs to define a coreDb point with the same name as the coreDb of the RTU that publishes the information. The **"SHARED SUBSCRIBE"** checkbox needs to be enabled. RTUs that "subscribe" to a published coreDb point do not need to define a source for these points in their databases - the value will be acquired by the publishing RTU.
- Data points shared in both coreDb must have the same name.

More information about "SHARED PUBLISH" and "SHARED SUBSCRIBE" can be found in section 5.5.3.

The configuration window (tab dbNet) for this functionality is shown below:

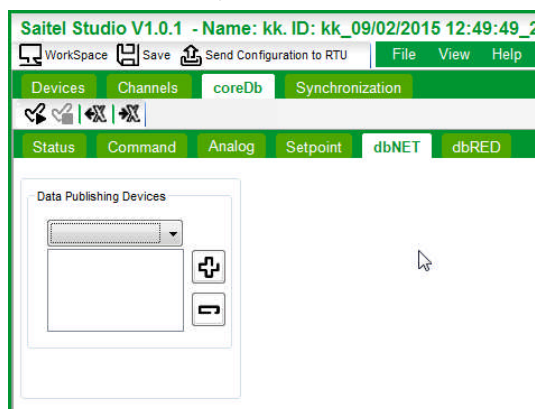


Figure 5-17 – Configuring dbNet.

For Saitel DP, you can configure up to 5 (LAN) devices through which coreDb points are to be published. For Saitel DR, the maximum number of devices is 3:

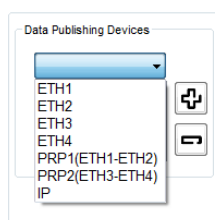


Figure 5-18 – Possible "Data Publishing Devices" in Saitel DP.

You have to configure a Data Publishing Device for each element publishing coreDb points. A "Data Publishing Device" can be a network interface or the broadcast address of another network.

Limits to be taken in account are:

- Only status and analog points can be shared.
- A maximum of 32 status points can be published.
- A maximum of 32 analog points can be published.
- A maximum of 128 status points can be subscribed.
- A maximum of 128 analog points can be subscribed.

## 5.8 Configuring Redundant RTUs (Only Saitel DR and Saitel DP)

Configuration of the redundancy is only available if the RTU has been defined as redundant (see field "Redundancy" in Section 3.2.1).

### NOTICE

HU\_B and HU\_BI don't allow redundancy.

By selecting tab **dbRED** in the coreDb configuration window, the user can configure the redundancy:

Figure 5-19 – Configuring dbRED (Redundancy control).

## 5.8.1 Control

The redundancy can be controlled using two different mechanisms:

- **MSAC** module. (Check field MSAC in Control zone). This is a legacy hardware that manages the switchover of CPUs as well as other supervisory features.
- **RCAP protocol** (Redundancy Control Asynchronous Protocol). In this case, there is a redundant switching channel between the CPUs, which is used to manage the switching operation using a Schneider Electric-proprietary protocol.

“Via #1” and “Via #2” will be available when “Protocol” is selected:

- **BACKPLANE**. Only available for Saitel DP and only when both CPUs are installed on the same backplane. Communications will be established by UDP over SLIP, over the backplane's serial port.
- **NET** (by Ethernet). It is necessary to configure the specific IP addresses used to exchange RCAP information between CPUs A and B.
- **SERIAL**. Two dedicated serial ports (“/tyCo/...”) in each CPU will be used to carry the RCAP information.

## 5.8.2 Mode

Two different redundancy modes are available:

- **Cold**: There is no coreDb synchronization between the two CPUs. When a switchover occurs, the new ONLINE CPU will start using its own coreDb with default values.
- **Hot**: There is a high speed communication channel (Ethernet or backplane) between the two CPUs, which is used to update the BACKUP CPU's database with the ONLINE CPU's database. When a switchover occurs, the new ONLINE CPU starts with updated values.

### NOTICE

In Hot mode, database IDs must be identical, i.e., it is very important to use the SAME Easergy Builder Configuration in both CPUs.

In Hot mode, “Via” allows to select if the replication is made by:

- **BACKPLANE**: Only available for Saitel DP.
- **NET** (by Ethernet): it is necessary to set the IP addresses used to synchronize coreDb between the CPUs A and B.

---

coreDb updates are synchronized by exception (only the variables that have changed), except the first time where the complete database is updated. The supervision point **DB\_UPDATE** monitors the process.

### 5.8.3 Bus

The Bus field indicates if the CPUs share the same Profibus or not (SHARED or DIFFERENT, respectively), regardless of whether they are in the same backplane or use RS-485 expansion. This is useful to detect failures in dual redundant systems.

- **SHARED:** Only available for Saitel DP. In this case, the bus of the STANDBY CPU is disabled.
- **DIFFERENT:** If checked, the bus is enabled even if the CPU is in STANDBY mode, so it can receive diagnostics from the modules.

### 5.8.4 Additional IPs

This is a list of IP addresses associated to the CPU that is in ONLINE. These addresses are associated in a dynamic way, so that in a redundant system they allow to communicate always with the CPU that is active. Regarding virtual addresses, it is even possible to assign multiple IP addresses to each port:

NOTICE
<p>If a static IP address and a virtual address are defined for the same device in the same subnet, a warning console message will be displayed to notify this abnormal situation (sup_redAddIPs: dev xxx ip x.x.x.x subnetMask xxxxxxxx).</p> <p>This message is a warning from the operating system; nevertheless, it will not cause a system malfunction, since the configuration will operate properly.</p>

# Chapter 6 - Working with Saitel

## 6.1 Introduction

This chapter covers, step by step, how to configure a Saitel RTU. The configuration process for a Saitel DR RTU is similar to Saitel DP, so only one of them (DR) is included in the following examples

In this example, an ITB (Intelligent Terminal Block) composed of an HU\_A as CPU and two slaves modules; AB\_DI and AB\_DO.

## 6.2 Creating the RTU into a WorkSpace

Open Easergy Builder and make sure the right Workspace is loaded. This workspace show should show all the RTUs defined in it:

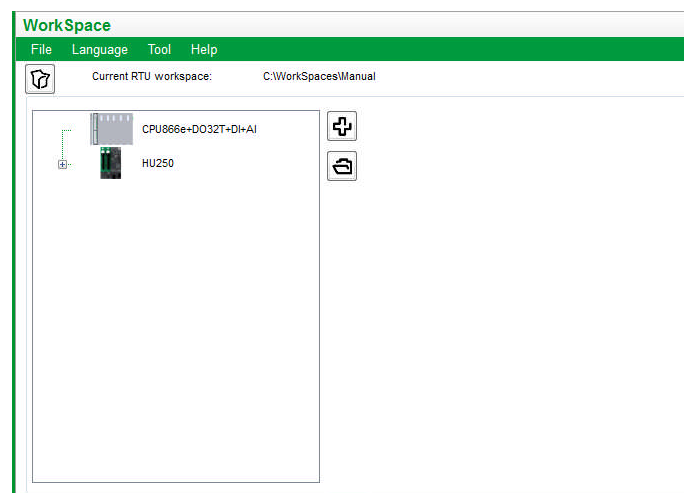


Figure 6-1 – Initial environment.

Create an RTU →  and define the ITB:

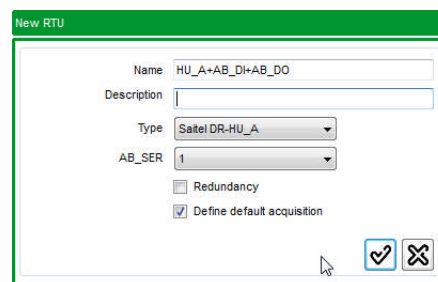


Figure 6-2 – A new RTU.

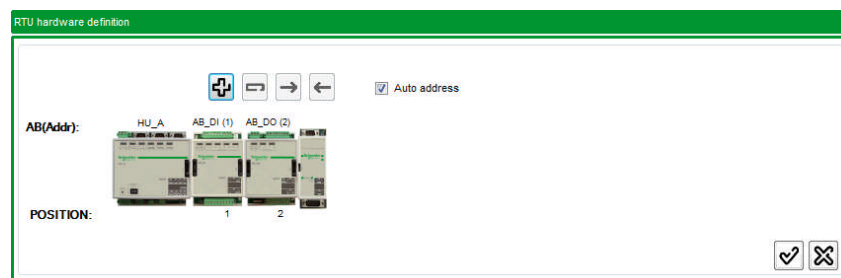


Figure 6-3 – Including slave modules.

Define users and interfaces:

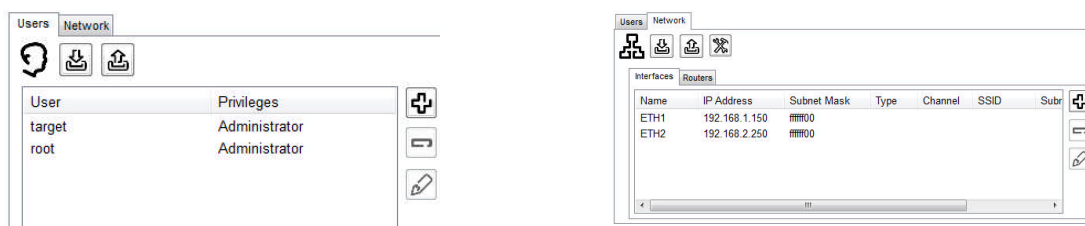


Figure 6-4 – Users and interfaces.

Define the connection of Easergy Builder to the RTU:



Figure 6-5 – Configuring the connection for Easergy Builder.

Transfer the general parameters to the RTU (only users and interfaces) →

Please, be sure that you can execute a ping command and the answer from the RTU is correct.

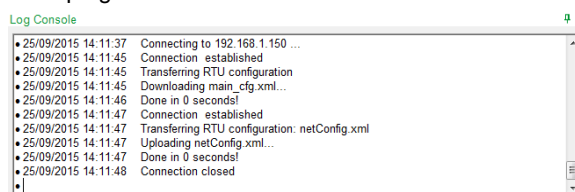


Figure 6-6 – Showing messages on the Log Console.

Create a Configuration →

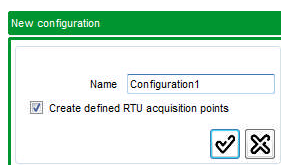


Figure 6-7 – Creating a configuration.

Double click on the configuration:

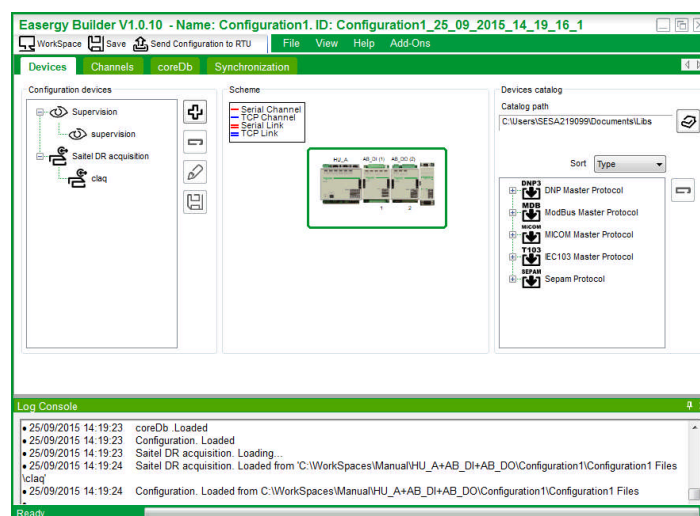


Figure 6-8 – Editing the new configuration.

Double click on the supervision device to configure the supervision points:

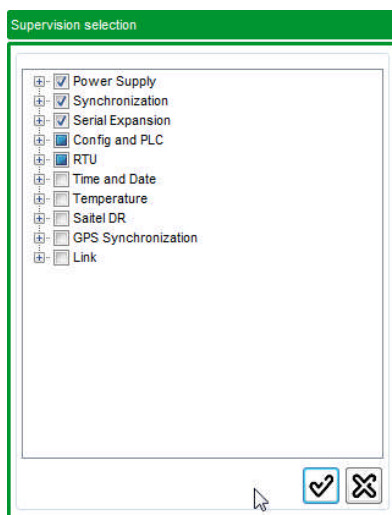


Figure 6-9 – Selecting supervision points to be included in coreDb.

Double click on the claq device to configure local acquisition.

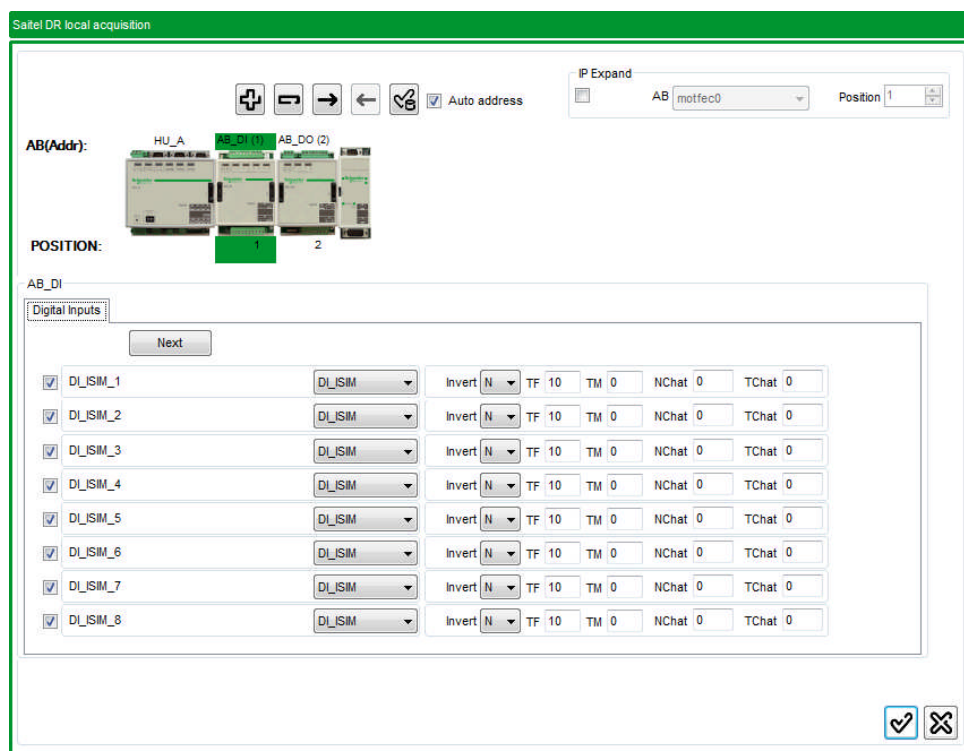


Figure 6-10 – Configuring local acquisition.

Press button  to insert in coreDb all points for this ITB. Selecting tab "coreDb" you can see these points into each table.

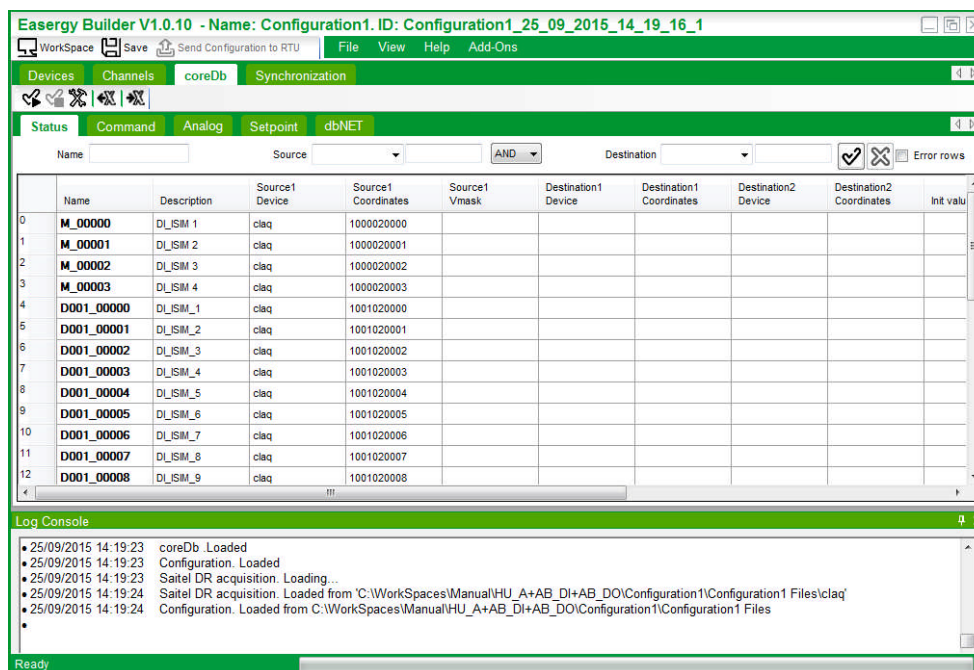


Figure 6-11 – Points defined in coreDb.

You can now define Channels, Synchronization and any other devices depending on your system architecture.

For more information about the configuration of the local acquisition in Saitel DR consult the manual “Configuration & Startup of Saitel DR”. For more information about each Device, please, consult its manual.

Transfer the configuration files to the RTU (all information except users and interfaces) → Send Configuration to RTU

# Chapter 7 - Working with Easergy T300

## 7.1 Introduction

Easergy Builder allows creating an RTU step by step, but this way is not recommended for this version.

You should import a configuration generated by the “T300 Generator” tool or from a RTU configuration file (tar.gz as explained in this chapter).

Using Easergy Builder you can after modify all parameters associated to this FRTU or its configurations:

- Network and modem parameters : See xx
- Synchronisation parameters : See xx
- System of events : See xx
- Create formula : See xx
- 

For the following items please refer to:

- Master and Slave protocols
- ISaGRAF program: See xx

With this initial configuration, you can after customize your T300 (add slave or master protocol, create your data mapping, manage your system of events....)

## 7.2 Sending Configuration Files to the FRTU

If you need to modify an FRTU configuration, select it in the workspace and double-click on it.

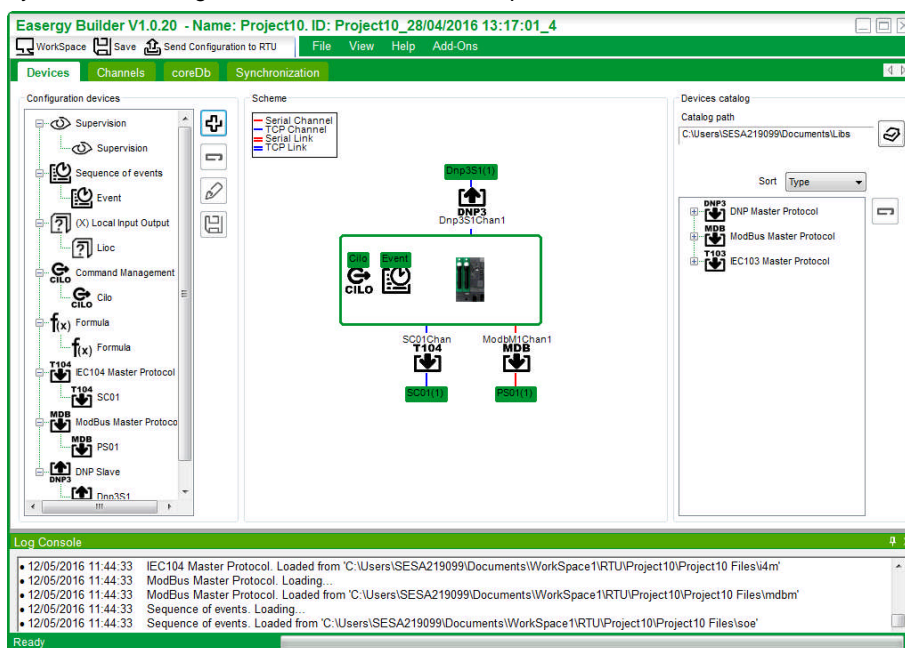




Figure 7-1 – Editing and sending a HU250 configuration

Make all necessary changes and press button  Send Configuration to RTU

If you don't need to make any changes, and only wish to send information to an FRTU, on the WorkSpace mode, select appropriate RTU and press button .



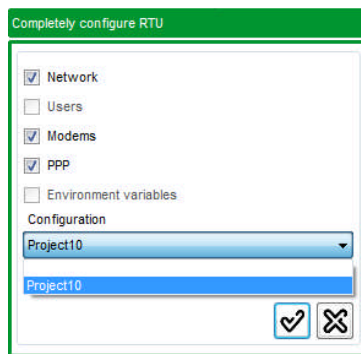


Figure 7-2 – Select data to be sent to the HU250

Select all elements to be sent to the RTU, including a configuration.

## 7.3 Configuring Cilo

### NOTICE

Only available for Easergy T300.

This Plugin allows configure the following parameters:

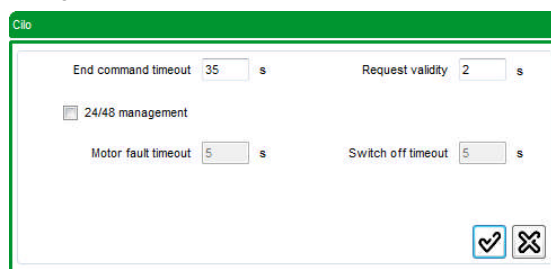


Figure 7-3 – Configuring Cilo

- **End command timeout** (in seconds): Timeout for completion of the command in progress (from 1 to 40). Default value: 35 s.
- **Request validity** (in seconds): Timeout to accept a request for execution of a command (from 1 to 10). Default value: 2 s.
- **24/48 management**: Check this box to enable management of the PS50 power supply.
- **Motor fault timeout**: Timeout for detecting the change from 1 to 0 at the point of failure of the motor output (from 1 to 10). Default value: 5 s.
- **Switch off timeout**: Timeout for sending the power-off command to the PS50 (from 1 to 10). Default value: 5 s.

## 7.4 Configuring Lioc

### NOTICE

Only available for Easergy T300.

The LIOC controller is used to manage HMI and digital inputs outputs of HU250 device. It manages:

- HMI buttons and LEDs.
- Digital Inputs / Outputs.
- PT100 probe.

You can configure the following parameters:

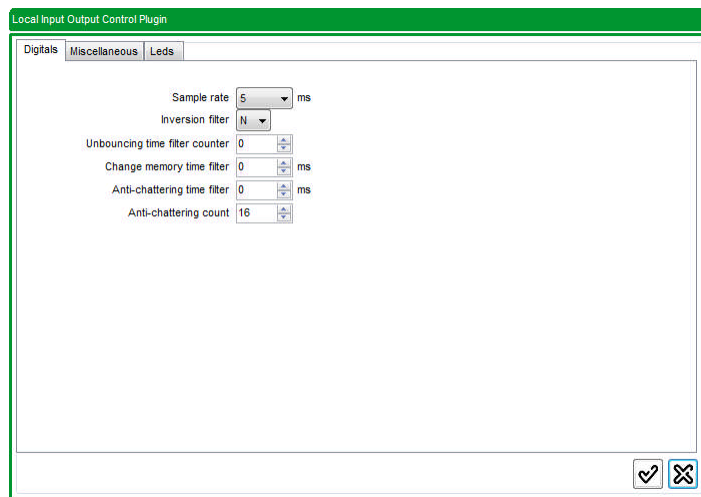


Figure 7-4 – Configuring Lioc

Selecting **Digital**s tab, following parameters can be configured:

- **Sample rate** (in milliseconds): Possible values: 1, 5 or 10 ms. Default value: 5 ms.
- **Inversion filter**: Possible values: Yes or No. Default value: No (disabled).
- **Unbouncing time filter counter**: Possible values: 0 and from 2 to 30. Default value: 0 (disabled).
- **Change memory time filter** (in milliseconds): Possible values: 0 and from 5 to 2000. Default value: 0 ms (disabled).
- **Anti-chattering time filter** (in milliseconds): Possible values: 0 and from 5 to 2000. Default value: 0 ms (disabled).
- **Anti-chattering count**: Possible values: From 1 to 255. Default value: 16.

Selecting **Miscellaneous** tab, the parameter “External Local/Remote Key” can set to TRUE or FALSE.

Using **Leds** tab, each front led can be configured with a different color depending on the status to be shown.

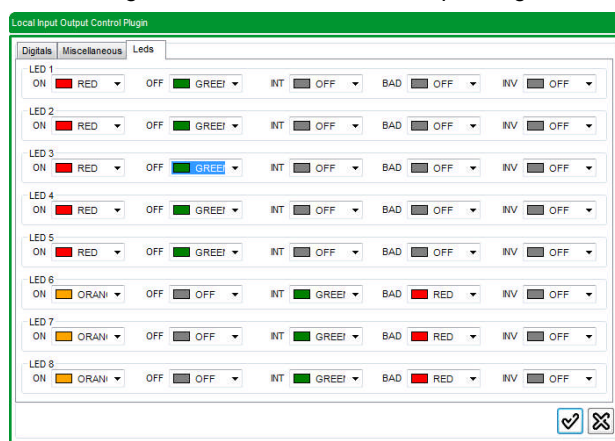


Figure 7-5 – Configuring front leds using Lioc

Where:

- **Led 1**: AC supply indicator.
- **Led 2**: 24/48V supply indicator.
- **Led 3**: 12V Telecom indicator.
- **Led 4**: Ext. 12V supply failure indicator.
- **Led 5**: Battery fault indicator.
- **Led 6**: Generic indicator 1.

- **Led 7:** Generic indicator 2.
- **Led 8:** Generic indicator 3.

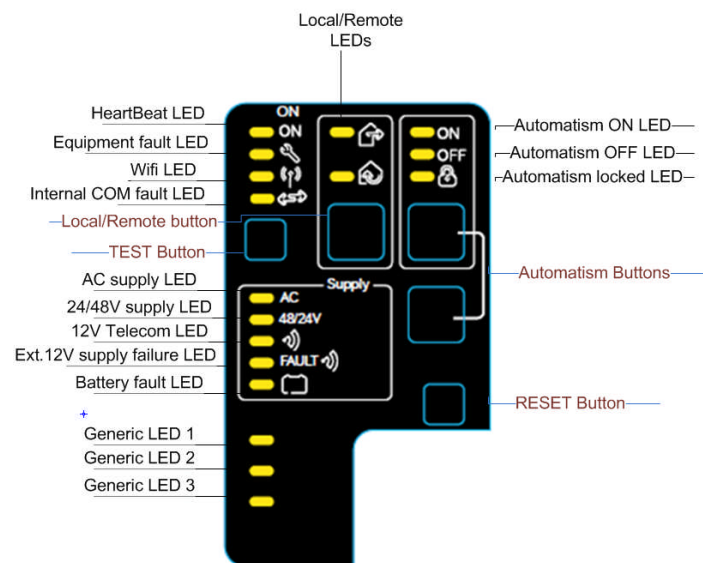


Figure 7-6 – Leds on the front panel

---

# Appendix A - Expressions

---

## Introduction

This appendix shows all expressions allowed to be calculated. In this section, following terms can be used:

- **dbValue:** Is an structure which contains a value, quality flag and timestamp.
- **Expression:** Any of the defined functions, operations, coreDb point name or constant value.
- **Invalid quality flag:** Quality flag with the bit IV\_LQF active. If a coreDb point has NT\_LQF, NT\_RQF, IV\_RQF active, IV\_LQF will make active internally by formula Controller.
- **Good quality flag:** Not Invalid Quality flag.
- **Controller start time:** Timestamp corresponding with the moment when the controller started.
- **NAN:** Not a Number (0x7FFFFFFF at HU\_A platform). It is used when a result cannot be calculated.

The coordinates are fixed and are always mapped as status, command, analog or setpoint source and its coordinate name must follow the structure of the implemented triggers or expressions, described on the previous section

Examples of valid expressions (mapped as source coordinate names) are:

```
SCALE(NOT(input2)+2,TEMPO(DPSTOSPS(input3),2*MIN(4-SPSTODPS(input5),input4)),input1)
MAX(8*NOT(input1), (input2+input3)/2) * (TEMPO(input4, 10)+SPSTODPS(input5)) / 2
OR(input1, NOT(input2)-1, MIN(0, input2)) – (-MAX(0,input1) / DPSTOSPS(input3))
-(input1 * NOT(input2) / (MAX(2, input2) + 1))
AND(input1 > input2, MAX(1, FORM_CYCLETIME) == FORM_PERIOD/2, IF(input2, input1+1, 2))
```

If any of the points mapped for formula as source is wrong introduced, it will be set as *INVALID* and will not be considered during calculation process at entry, since this point will not be included in the list of formulas.

Examples of valid triggers (mapped as destination coordinate names) are:

```
input1:EVENT
input2:VAL:-2.34
input3:VAL:1
```

In this case, we are supposing that *input1*, *input2* and *input3* are coreDb point names defined on status, analog, command or setpoint tables.

These coreDb points also have a formula source coordinate expression to be calculated. If not, the trigger will not be taken into account and deleted from memory.

---

## NOT (*input*) → return: *output*

Where:

- *input* and *output* are dbValues.

Where *input* is a dbValue defined by an expression. The function is a NOT of this input so its return, another dbValue called as *output*, could be:

Input variable value ( <i>input.value</i> )	Calculated value ( <i>output.value</i> )	Quality returned ( <i>output.qFlag</i> )	Timestamp returned ( <i>output.tSpec</i> )
0	1	Same as input	Same as input
Other case	0	Same as input	Same as input

This function is implemented according to the following pseudo-code:

```
If (input.value != !previousResult)
    → output.value = !input.value
```

Output timestamp and quality flag values will be the same as input.

If input value is equal to NAN, the result value will be NAN (0 at status or command) with quality flag sets to invalid and timestamp equal to writing time.

### Example

---

#### NOT (*input1*)

Given that *input1* is a coreDb point name, NOT function will return:

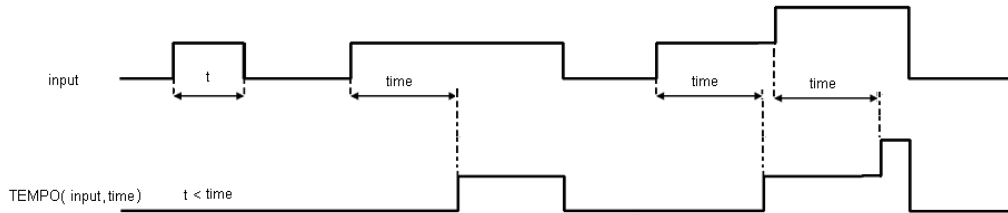
- 0 if *input1* is different to 0.
- 1 if *input1* is equal to 0.
- Quality flag will be always the same as *input1* quality flag.
- Output timestamp will be the same as *input1* timestamp.

## TEMPO (*input*, *time*) → return: *output*

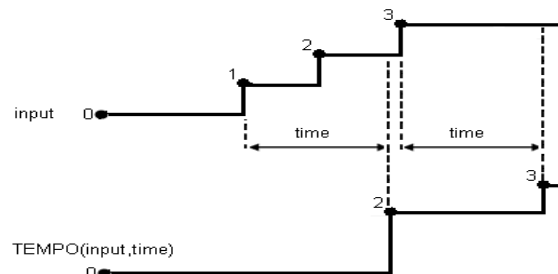
Where:

- Input*, *time* and *output* are dbValues defined by expressions (*time.value* indicates a number of seconds).

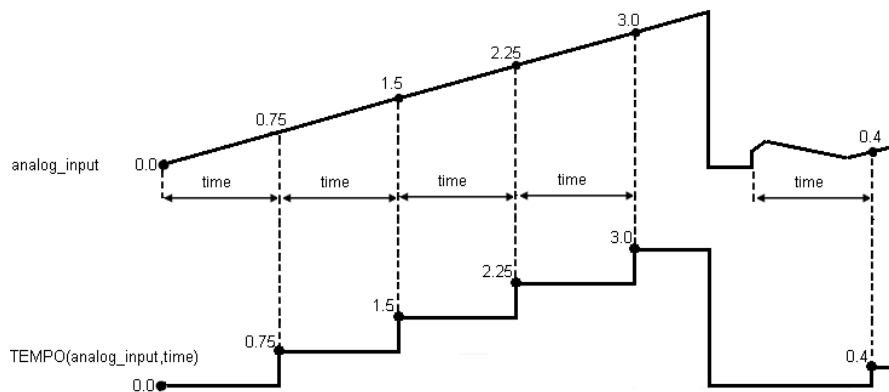
The function returns *input.value* if it moves to 1, or another value different to 0, for at least *time.value* seconds. After this time, the function returns this value of *input.value* as long as it is at this value. If *input.value* moves to 0, its returns 0 immediately:



TEMPO function basic diagram



TEMPO function with multiple changes diagram



TEMPO function with analog input diagram

The returned quality is the same as *input.qFlag*.

The returned timestamp is the moment which tempo signal changes, so *time* seconds after *input* if its value is different to 0 or the moment in which *input* goes equal to 0, both in formula Controller local time.

The maximum *time* value allowed is the size of an unsigned long divided by 1000 (4294967295/1000 in our systems). If its value is negative, the expression will get the absolute value.

If *input* value or quality changes, when the value is different to 0, the TEMPO formula will wait *time* seconds to change both on return. If many changes are done on this time, the TEMPO counter will start with the first one and *time* seconds after, this formula will return with the last changes.

If *input* value is equal to 0, TEMPO formula will return any new change at quality flag immediately.

If input or time value is equal to NAN, the result value will be equal to NAN (0 at status or command) and its quality flag equal to IV\_LQF.

This function is implemented according to the following pseudo-code:

```
If (input.value == 0 && (previousValue!=input.value || previousQFlag!=input.qFlag))
    → output.value = 0; output.qFlag = input.qFlag;
    → output.tSpec = LOCAL_NOW();
    → notWaitTime = 1;
Else if (input.value!=0 && (previousResult!=input.value || previousQFlag!=input.qFlag) &&
notWaitTime)
    → cntTime = sysCnt1mlsg();
    → notWaitTime = 0
If (notWaitTime == 0)
    → if (sysCnt1mlsg() - cntTime >= time.value_ms)
        → output.value = input.value; output.qFlag = input.qFlag
        → output.tSpec = LOCAL_NOW();
        → notWaitTime = 1
```

Where:

- *notWaitTime* is a flag that indicates if the algorithm is not waiting (when it is equal to 1) the seconds indicated by *time.value* to change the *la\_outputVal* to a value different to 0.
- *cntTime* stores the time when *input.value* changed to a value different to 0.
- *time.value\_ms* is the value of *time.value* in milliseconds, converted to be compared with the system counter (*sysCnt1mlsg()*) which value is returned in ms.
- *LOCAL\_NOW()* is a macro which returns the local time of the RTU at this moment.

## Example

### TEMPO (input1, 10)

Given that *input1* is a coreDb point name, TEMPO function will return:

- 0 if *input1* is equal to 0.
- A value equal to *input1.value* if it is different to 0, and this value has been written 10 seconds or more before.

## SPSTODPS (*input*) → return: *output*

Where:

- *input* and *output* are dbValues defined by expressions.

The function returns 1 if the input value last bit is 0, or 2 if this bit is 1.

Input variable last bit value ( <i>input.value</i> )	Input quality flag ( <i>input.qFlag</i> )	Calculated value ( <i>output.value</i> )	Quality returned ( <i>output.qFlag</i> )	Timestamp returned ( <i>output.tSpec</i> )
0	GOOD	1	Same as input	Same as input
1	GOOD	2	Same as input	Same as input
X	IV_LQF, IV_RQF, NT_LQF or NT_RQF	3	Same as input	Same as input

The input value is always casted as long integer.

This function is implemented according to the following pseudo-code:

```
If (If (input.qFlag == IV || input.qFlag == NT) → output.value = 3
Else if (input.value & 1 == 0) → output.value = 1
Else if (input.value & 1 == 1) → output.value = 2
```

The quality and timestamp returned are the same as *input.qFlag* and *input.tStamp* respectively.

If input value is equal to NAN, the result value will be NAN (0 at status or command) with quality flag as invalid and timestamp equal to writing time.

### Example

#### SPSTODPS (*input1*)

Given that *input1* is a coreDb point name, SPSTODPS function will return:

- 3 if *input1* quality flag has IV\_LQF, IV\_RQF, NT\_LQF or NT\_RQF set to 1.
- 1 if *input1* last bit is equal to 0.
- 2 if *input1* last bit is equal to 1.
- Output quality flag will be always the same as *input1* quality flag.
- Output timestamp will be the same as *input1* timestamp.



## DPSTOSPS (*input*) → return: *output*

Where:

- Input* and *output* are dbValues defined by expressions.

Input variable value ( <i>input.value</i> )	Calculated value ( <i>output.value</i> )	Quality returned ( <i>output.qFlag</i> )	Timestamp returned ( <i>output.tSpec</i> )
0	0	<i>input.qFlag</i>   IV_LQF	Same as input
1	0	<i>input.qFlag</i>	Same as input
2	1	<i>input.qFlag</i>	Same as input
3 or other	0	<i>input.qFlag</i>   IV_LQF	Same as input

When an *output* returns with a quality flag equal to *input.qFlag* | IV\_LQF, means the same quality as input adding, and making sure, the Invalid flag (IV\_LQF) set up at output quality.

The returned timestamp is the same as *input.tSpec*.

This function is implemented according to the following pseudo-code:

```
If (input.value == 0) → output.value = 0 && output.qFlag = input.qFlag | IV_LQF
Else if (input.value == 1) → output.value = 0 && output.qFlag = input.qFlag
Else if (input.value == 2) → output.value = 1 && output.qFlag = input.qFlag
Else → output.value = 0 && output.qFlag = input.qFlag | IV_LQF
```

If input value is equal to NAN, the output value will be set as NAN (0 at status or command) with quality flag equals to invalid and timestamp equal to writing time.

### Example

#### DPSTOSPS (*input1*)

Given that *input1* is a coreDb point name, DPSTOSPS function will return:

- 0 as *output.value* and *input.qFlag* | INVALID as *output.qFlag* if *input1* value is equal to 0 or equal or greater than 3.
- 0 as *output.value* and *input.qFlag* as *output.qFlag* if *input1* value is equal to 1.
- 1 as *output.value* and *input.qFlag* as *output.qFlag* if *input1* value is equal to 2.
- Output timestamp will be the same as *input1*.
- Output timestamp will be the same as *input2* timestamp when it got its minimum value.

## OR (*input\_1*, *input\_2*, ..., *input\_N*) → return: *output*

Where:

- Where *input\_i* is a dbValue defined by an expression, where  $\forall i \in \{1, 2, \dots, N\}$ .

The function is an OR of these inputs so its return, on another dbValue called as *output*, could be:

- 1 at *output* value if one or more of the input values are different to 0.
  - If one or more of these inputs, with value different to 0, has quality as good, it returns the output quality (*output.qFlag*) and timestamp (*output.tSpec*) of the most recent *input* with good quality flag.
  - If every input with value 1 has quality different to good, it returns the output quality (*output.qFlag*) and timestamp (*output.tSpec*) of the most recent *input*.
- 0 at *output* value if all input values are 0.
  - If every input has quality as good, it returns the output quality the output quality (*output.qFlag*) and timestamp (*output.tSpec*) of the most recent *input*.
  - If one or more of the inputs have quality different to good, it the output quality (*output.qFlag*) and timestamp (*output.tSpec*) of the most recent *input* with quality flag different to good.

This function is implemented according to the following pseudo-code:

```
Output.qFlag = input_1.qFlag; output.tSpec = input_1.tSpec;
Foreach input_i where i ∈ {1, 2, .. , N}
  If (input_i.value != 0) → output.value = 1
    → If(input_i.qFlag == GOOD)
      → if(output.qFlag != GOOD || input_i.tSpec > output.tSpec)
        → output.qFlag = input_i.qFlag; output.tSpec = input_i.qFlag;
    → Else
      → if(output.qFlag != GOOD && input_i.tSpec > output.tSpec)
        → output.qFlag = input_i.qFlag; output.tSpec = input_i.qFlag;
Else if (output.value == 0)
  → If(input_i.qFlag != GOOD)
    → if(output.qFlag == GOOD || input_i.tSpec > output.tSpec)
      → output.qFlag = input_i.qFlag; output.tSpec = input_i.qFlag;
    → Else
      → if(output.qFlag == GOOD && input_i.tSpec > output.tSpec)
        → output.qFlag = input_i.qFlag; output.tSpec = input_i.qFlag;
End foreach
```

Output timestamp will be equal to the timestamp value of the input which set the quality flag.

If any of the input values are equal to NAN, OR function result will be NAN (0 at status or command) with invalid as quality flag when none of the rest of inputs are equal to 1, or a value different to 0.

### Example

#### OR (input1, 1, input2)

- Given that *input1* and *input2* are coreDb point names and their values are equal to 0, OR function will return 1 as *output.value*. The quality flag and timestamp will come from the constant input equal to 1 (GOOD quality flag and Controller start time as timestamp):
- If *input1* value changes to a value different to 0, AND function will return 1 as *output.value*. The quality flag and timestamp will come from *input1* since is more recent than the constant input.
- If *input2* value also moves to a value different to 0, AND function will return 1 as *output.value*. The quality flag and timestamp will depend on:
  - If *input1* and *input2* quality flags are GOOD type, the output quality flag and timestamp come from the most recent of the inputs.

- 
- If *input1* or *input2* quality flags are not GOOD type, the output quality flag and timestamp come from the input with not GOOD quality flag type.
  - If *input1* and *input2* quality flags are not GOOD type, the output quality flag and timestamp come from the most recent quality flag.

## AND (*input\_1*, *input\_2*, ..., *input\_N*) → return: *output*

Where:

- Where *input\_i* is a dbValue defined by an expression, where  $\forall i \in \{1, 2, \dots, N\}$ .

The function is an AND of these inputs so its return, on another dbValue called as *output*, could be:

- 0 at *output* value if one or more of the input values are equal to 0
  - If one or more of these inputs, with value equal to 0, has quality as good, it returns the output quality (*output.qFlag*) and timestamp (*output.tSpec*) of the most recent *input* with good quality flag.
  - If every input with value 0 has quality different to good, it returns the output quality (*output.qFlag*) and timestamp (*output.tSpec*) of the most recent *input*.
- 1 at *output* value if all input values are different to 0
  - If every input has quality as good, it returns the output quality the output quality (*output.qFlag*) and timestamp (*output.tSpec*) of the most recent *input*
  - If one or more of the inputs have quality different to good, it the output quality (*output.qFlag*) and timestamp (*output.tSpec*) of the most recent *input* with quality flag different to good.

This function is implemented according to the following pseudo-code:

```
Output.qFlag = input_1.qFlag; output.tSpec = input_1.tSpec;
Foreach input_i where i ∈ {1, 2, .. , N}
  If (input_i.value == 0) → output.value = 0
    → If(input_i.qFlag == GOOD)
      → if(output.qFlag != GOOD || input_i.tSpec > output.tSpec)
        → output.qFlag = input_i.qFlag; output.tSpec = input_i.qFlag;
      → Else
        → if(output.qFlag != GOOD && input_i.tSpec > output.tSpec)
          → output.qFlag = input_i.qFlag; output.tSpec = input_i.qFlag;
  Else if (output.value == 1)
    → If(input_i.qFlag != GOOD)
      → if(output.qFlag == GOOD || input_i.tSpec > output.tSpec)
        → output.qFlag = input_i.qFlag; output.tSpec = input_i.qFlag;
      → Else
        → if(output.qFlag == GOOD && input_i.tSpec > output.tSpec)
          → output.qFlag = input_i.qFlag; output.tSpec = input_i.qFlag;
End foreach
```

Output timestamp will be equal to the timestamp value of the input which set the quality flag.

If any of the input values are equal to NAN, AND function result will be NAN (0 at status or command) with invalid as quality flag when none of them are equal to 0.

### Example

#### AND (input1, 1, input2)

- Given that *input1* and *input2* are coreDb point names and their values are equal to 0, AND function will return 0 as *output.value*. The quality flag and timestamp will depend on:
  - If *input1* and *input2* quality flags are GOOD type, the output quality flag and timestamp come from the most recent of the inputs.
  - If *input1* or *input2* quality flags are not GOOD type, the output quality flag and timestamp come from the input with GOOD quality flag type.
  - If *input1* and *input2* quality flags are not GOOD type, the output quality flag and timestamp come from the most recent quality flag.

- 
- If *input1* value changes to a value different to 0, AND function will return 0 as *output.value*. The quality flag and timestamp will come from *input2*.
  - If *input2* value also moves to a value different to 0, AND function will return 1 as *output.value*. The quality flag and timestamp will depend on:
    - If *input1* and *input2* quality flags are GOOD type, the output quality flag and timestamp come from the most recent of the inputs.
    - If *input1* or *input2* quality flags are not GOOD type, the output quality flag and timestamp come from the input with not GOOD quality flag type.
    - If *input1* and *input2* quality flags are not GOOD type, the output quality flag and timestamp come from the most recent quality flag.

---

## MIN (*numDays*, *input*) → return: *output*

Where:

- Where *numDays* and *input* is a dbValue defined by an expressions.

This function returns the minimum value of *input.value* during the number of days, indicated by *numDays.value* (between 0 and 30), at the end of this number of days. Once the new value is set, the function starts again.

Output timestamp and quality flag are set up equals to timestamp and quality flag from *input*, respectively, when the minimum value was read.

If the same minimum value is reached several times, the returned quality flag and timestamp are from the last one.

Each period starts and ends at midnight of the day. The writing is only done once the period ends.

If the number of days (*numDays.value*) is equal to 0, the function returns the value of the second parameter (*input*) each time it is updated.

If the number of days is greater than 30 or less than 0, the function will be updated with 0 as value and the output quality flag will be equal to invalid. The output timestamp, in this case, is equal to calculation moment.

If *numDays* or *input* values are equal to NAN, the result value will be equal to NAN (0 at status or command) with invalid output quality flag.

This function is implemented according to the following pseudo-code:

```
If(numDays.value == 0)
    → output = input
Else if(numDays.value > 0) && (la_fstNumDaysVal <= 30)
    → if(input.value <= previousResult)
        → output = input
    → Else if(fstVal)
        → output = input;
        → fstVal = 0;
    → if(acumTime < currentTSpec)
        → krunchData; startTSpec = currentTSpec;
        → fstVal = 1;
Else → output.value = 0; output.qFlag = IV_LQF; output.tSpec = LOCAL_NOW();
```

---

### Example

#### MIN (3, *input1*)

- Given that *input1* is a coreDb point names, MIN function will return the minimum value of *input1.value* during 3 days. Once the 3 days are reached, the days counter will start again. The output quality flag and timestamp will be equal to input quality and timestamp, respectively, when the minimum value was reached.

---

## MAX (*numDays*, *input*) → return: *output*

Where:

- Where *numDays* and *input* is a dbValue defined by an expressions.

This function returns the maximum value of *input.value* during the number of days, indicated by *numDays.value* (between 0 and 30), at the end of this number of days. Once the new value is set, the function starts again.

Output timestamp and quality flag are set up equals to timestamp and quality flag from *input*, respectively, when the maximum value was read.

If the same maximum value is reached several times, the returned quality flag and timestamp are from the last one.

Each period starts and ends at midnight of the day. The writing is only done once the period ends.

If the number of days (*numDays.value*) is equal to 0, the function returns the value of the second parameter (*input*) each time it is updated.

If the number of days is greater than 30 or less than 0, the function will be updated with 0 as value and the output quality flag will be equal to invalid. The output timestamp, in this case, is equal to calculation local moment.

If *numDays* or *input* values are equal to NAN, the result value will be equal to NAN (0 at status or command) with invalid output quality flag.

This function is implemented according to the following pseudo-code:

```
If(numDays.value == 0)
    → output = input
Else if(numDays.value > 0) && (la_fstNumDaysVal <= 30)
    → if(input.value >= previousResult)
        → output = input
    → else if(fstVal)
        → output = input;
        → fstVal = 0;
    → if(acumTime < currentTSpec)
        → krunchData; startTSpec = currentTSpec;
        → fstVal = 1;
Else → output.value = 0; output.qFlag = IV_LQF; output.tSpec = LOCAL_NOW();
```

---

### Example

#### MAX (15, input1)

- Given that *input1* is a coreDb point names, MAX function will return the maximum value of *input1.value* during 15 days. Once the 15 days are reached, the days counter will start again. The output quality flag and timestamp will be equal to input quality and timestamp, respectively, when the maximum value was reached.

---

## SCALE (*factor*, *offset*, *input*) → return: *output*

Where:

- Where *factor*, *offset* and *input* is a dbValue defined by an expressions.

This function returns the value of *input.value* multiplied by *factor.value*, adding the value of *offset.value*:

$$\text{output.value} = \text{input.value} * \text{factor.value} + \text{offset.value}$$

Output quality flag and timestamp is equal to *input* quality flag and timestamp, respectively.

If *factor*, *offset* or *input* value is equal to NAN, the result value will be set as NAN (0 at status or command) with invalid output quality flag.

This function is implemented according to the following pseudo-code:

```
output.value = input.value * factor.value + offset.value
output.qFlag = input.qFlag
output.tSpec = input.tSpec
```

### Example

---

#### SCALE (2, 0.5, input1)

- Given that *input1* is a coreDb point names, SCALE function will return the result of:

$$\text{output.value} = 2 * \text{input1.value} + 0.5$$

$$\text{output.qFlag} = \text{input1.qFlag}$$

$$\text{output.tSpec} = \text{input1.tSpec}$$



---

## IF (*cond*, *ifVar*) → return: *output*

Where:

- Where *cond* and *ifVar* are dbValues defined by expressions.

Condition value ( <i>cond.value</i> )	Returned value ( <i>output.value</i> )	Quality returned ( <i>output.qFlag</i> )	Timestamp returned ( <i>output.tSpec</i> )
0	No change	No change	No change
Other case	Same as <i>ifVar</i>	Same as <i>ifVar</i>	Same as <i>ifVar</i>

If *cond* or *ifVar* value is equal to NAN, the result value will be set as NAN (0 at status or command) with invalid output quality flag.

This function (with two parameters) only can be as used as main expression:

IF(*cond*1, *input*1\**input*2) → VALID

IF(NOT(*cond*1), *input*1) \* *input*2 → INVALID (the main expression is the Multiplication Operation (\*))

SCALE(*input*2, IF(*cond*1, *input*3\**input*1), *input*4) → INVALID (the main expression is the SCALE function).

This function is implemented according to the following pseudo-code:

```
If(cond.value != 0) → output = ifVar
```

---

### Example

#### IF(1, *input*1)

- Given that *input*1 is a coreDb point names, IF with two parameters will return always the value, quality Flag and timestamp from *input*1.

---

## IF (*cond*, *ifVar*, *elseVar*) → return: *output*

Where:

- Where *cond*, *ifVar* and *elseVar* are dbValues defined by expressions.

This function returns *ifVar* if the value of *cond* is different to 0. Otherwise, *elseVar* is returned as result:

Condition value ( <i>cond.value</i> )	Returned value ( <i>output.value</i> )	Quality returned ( <i>output.qFlag</i> )	Timestamp returned ( <i>output.tSpec</i> )
0	Same as <i>elseVar</i>	Same as <i>elseVar</i>	Same as <i>elseVar</i>
Other case	Same as <i>ifVar</i>	Same as <i>ifVar</i>	Same as <i>ifVar</i>

If *cond*, *ifVar* or *elseVar* value is equal to NAN, the result value will be set as NAN (0 at status or command) with invalid output quality flag.

This function (with three parameters) can be used as an expression as a sub-expression.

This function is implemented according to the following pseudo-code:

```
If(cond.value != 0) → output = ifVar  
Else → output = elseVar
```

---

### Example

#### IF(0, input1,input2)

- Given that *input1* and *input2* are coreDb point names, IF with three parameters will return always the value, quality Flag and timestamp from *input2*.

---

## PLUS OPERATION (*input1* + *input2*) → return: *output*

Where:

- Where *input1* and *input2* are dbValues defined by expressions.

This function returns the value of *input1.value* plus *input2.value*.

*output.value* = *input1.value* + *input2.value*

If both inputs are not constant values, the returned quality flag is *input1.qFlag* | *input2.qFlag*, and the output timestamp is equal to the most recent of both inputs.

If only one of the inputs is not a constant value, the returned quality flag and timestamp are equal to this input values for quality and timestamp.

If both inputs are constants, the returned quality and timestamp are equal to the second input values for quality and timestamp.

This function is implemented according to the following pseudo-code:

```
output.value = input1.value + input2.value
If (input1.type != CONST)
    → output.qFlag = input1.qFlag && output.tSpec = input1.tSpec
If (input2.type != CONST) && input2.timestamp > input1.timestamp)
    → output.qFlag |= input2.qFlag
    → if (input2.tSpec > input1.tSpec)
        → output.tSpec = input2.tSpec
Else
    → output.qFlag = input2.qFlag && output.tSpec = input2.tSpec
```

If any of the input values are equal to NAN, the result value will be set as NAN (0 at status or command) with invalid output quality flag

### Example

---

#### input1 + input2

- Given that *input1* is a coreDb point name, this function will return the result of:

*output.value* = *input1.value* + 3

*output.qFlag* = *input1.qFlag*

*output.tSpec* = *input1.tSpec*

---

## MINUS OPERATION (*input1* - *input2*) → return: *output*

Where:

- Where *input1* and *input2* are dbValues defined by expressions.

Where *input1* and *input2* are dbValue defined by expressions. This function returns the value of *input1.value* plus *input2.value*.

*output.value* = *input1.value* - *input2.value*

If both inputs are not constant values, the returned quality flag is *input1.qFlag* | *input2.qFlag*, and the output timestamp is equal to the most recent of both inputs.

If only one of the inputs is not a constant value, the returned quality flag and timestamp are equal to this input values for quality and timestamp.

If both inputs are constants, the returned quality and timestamp are equal to the second input values for quality and timestamp.

This function is implemented according to the following pseudo-code:

```
output.value = input1.value - input2.value
If (input1.type != CONST)
    → output.qFlag = input1.qFlag && output.tSpec = input1.tSpec
If (input2.type != CONST) && input2.timestamp > input1.timestamp)
    → output.qFlag |= input2.qFlag
    → if (input2.tSpec > input1.tSpec)
        → output.tSpec = input2.tSpec
Else
    → output.qFlag = input2.qFlag && output.tSpec = input2.tSpec
```

If any of the input values are equal to NAN, the result value will be set as NAN (0 at status or command) with invalid output quality flag.

### Example

#### **-(-input1 – input2 – MIN(4, input3))**

- Given that *input1*, *input2* and *input3* are coreDb point names, the function will return, once each parameter gets its values:

*output.value* = - (((*input1.value* - *input2.value*) - MIN(4, *input3*)))

Calculating it from left to right.

*output.qFlag* = *input1.qFlag* | *input2.qFlag* | MIN(4, *input3*).qFlag

In this case, the output quality flag is adding the quality flag values from each inputs, with type different to constant. The last one, is the returned value of MIN(4, *input3*) quality flag returned.

The *output.tSpec* will be equal to the most recent between *input1.tSpec*, *input2.tSpec* and the returned timestamp from MIN(4, *input3*) calculation.

## MULTIPLICATION OPERATION (*input1\*input2*) → return: *output*

Where:

- Where *input1* and *input2* are dbValues defined by expressions.

This function returns the value of *input1.value* plus *input2.value*.

*output.value* = *input1.value* \* *input2.value*

If both inputs are not constant values, the returned quality flag is *input1.qFlag* | *input2.qFlag*, and the output timestamp is equal to the most recent of both inputs.

If only one of the inputs is not a constant value, the returned quality flag and timestamp are equal to this input values for quality and timestamp.

If both inputs are constants, the returned quality and timestamp are equal to the second input values for quality and timestamp.

This function is implemented according to the following pseudo-code:

```
output.value = input1.value * input2.value
If (input1.type != CONST)
    → output.qFlag = input1.qFlag && output.tSpec = input1.tSpec
If (input2.type != CONST) && input2.timestamp > input1.timestamp)
    → output.qFlag |= input2.qFlag
    → if (input2.tSpec > input1.tSpec)
        → output.tSpec = input2.tSpec
Else
    → output.qFlag = input2.qFlag && output.tSpec = input2.tSpec
```

If any of the input values are equal to NAN, the result value will be set as NAN (0 at status or command) with invalid output quality flag.

### Example

#### (*input1 + input2*) \* 4 + MIN(4,*input3*)

- Given that *input1*, *input2* and *input3* are coreDb point names, the function will return, once each parameter gets its values:

Firstly, it calculates the value of *input1.value+input2.value*, then this result is multiplied by 4 and, finally, the value of calculate *MIN(4,input3)* is added:

*output.value* = (((*input1.value* + *input2.value*)\*4) + *MIN(4,input3)*)

Calculating it from left to right, having into account that multiplication has higher precedence than plus operation, but less than brackets.

*output.qFlag* = *input1.qFlag* | *input2.qFlag* | *MIN(4, input3).qFlag*

In this case, the output quality flag is adding the quality flag values from each inputs, with type different to constant. The last one, is the returned value of *MIN(4, input3)* quality flag returned.

The *output.tSpec* will be equal to the most recent between *input1.tSpec*, *input2.tSpec* and the returned timestamp from *MIN(4, input3)* calculation.

---

## DIVISION OPERATION (*input1*/*input2*) → return: *output*

Where:

- Where *input1* and *input2* are dbValues defined by expressions.

This function returns the value of *input1.value* plus *input2.value*.

*output.value* = *input1.value* / *input2.value*

If both inputs are not constant values, the returned quality flag is *input1.qFlag* | *input2.qFlag*, and the output timestamp is equal to the most recent of both inputs.

If only one of the inputs is not a constant value, the returned quality flag and timestamp are equal to this input values for quality and timestamp.

If both inputs are constants, the returned quality and timestamp are equal to the second input values for quality and timestamp.

This function is implemented according to the following pseudo-code:

```
output.value = input1.value / input2.value
If (input1.type != CONST)
    → output.qFlag = input1.qFlag && output.tSpec = input1.tSpec
If (input2.type != CONST) && input2.timestamp > input1.timestamp)
    → output.qFlag |= input2.qFlag
    → if (input2.tSpec > input1.tSpec)
        → output.tSpec = input2.tSpec
Else
    → output.qFlag = input2.qFlag && output.tSpec = input2.tSpec
```

If any of the input values are equal to NAN, the result value will be set as NAN (0 at status or command) with invalid output quality flag. Besides, if the second input value is equal to 0, the returned value will be also NAN (0 at status or command) with invalid output quality flag.

---

### Example

#### 3 + input1 / NOT(input2)

- Given that *input1* and *input2* coreDb point names, the function will return, once each parameter gets its values:

Firstly, it calculates the value of *input1.value*/NOT(*input2*), then 3 is added to this previous result.

*output.value* = (*input1.value* / NOT(*input2*)) + 3

Calculating it from left to right. Besides, division calculation has higher precedence than plus or minus operation.

*output.qFlag* = *input1.qFlag* | NOT( *input2*).*qFlag*

In this case, the output quality flag is adding the quality flag values from each inputs, with type different to constant. The last one, is the returned value of NOT(*input2*) quality flag returned.

The *output.tSpec* will be equal to the most recent between *input1.tSpec* and the returned timestamp from NOT(*input2*) calculation.

If NOT(*input2*) result value is equal to 0 (so *input2.value* is different to 0), when the division operation is done, its result is equal to NaN (0 at status or command) and it is inherited to the complete formula.

---

## GREATER CONDITIONAL OPERATION (*input1*>*input2*) → return: *output*

Where:

- Where *input1* and *input2* are dbValues defined by expressions.

This function returns 1 if the value of *input1* is greater than *input2.value*. Otherwise the returned value is 0.

If both inputs are not constant values, the returned quality flag is *input1.qFlag* | *input2.qFlag*, and the output timestamp is equal to the most recent of both inputs.

If only one of the inputs is not a constant value, the returned quality flag and timestamp are equal to this input values for quality and timestamp.

If both inputs are constants, the returned quality and timestamp are equal to the second input values for quality and timestamp.

Conditional operators have the lowest precedence between the operators.

This function is implemented according to the following pseudo-code:

```
If(input1.value > input2.value)
    → output.value = 1;
Else
    → output.value = 0;
If (input1.type != CONST)
    → output.qFlag = input1.qFlag && output.tSpec = input1.tSpec
If(input2.type != CONST) && input2.timestamp > input1.timestamp)
    → output.qFlag |= input2.qFlag
    → if(input2.tSpec > input1.tSpec)
        → output.tSpec = input2.tSpec
Else
    → output.qFlag = input2.qFlag && output.tSpec = input2.tSpec
```

If any of the input values are equal to NAN, the result value will be set as NAN (0 at status or command) with invalid output quality flag.

### Example

#### *input1* > 2

- Given that *input1* is coreDb point name, this function will return 1 if *input1.value* is greater than 2. If not, the returned value is equal to 0:

*output.qFlag* = *input1.qFlag*

*output.tSpec* = *input1.tSpec*

---

## LESS CONDITIONAL OPERATION (*input1* < *input2*) → return: *output*

Where:

- Where *input1* and *input2* are dbValues defined by expressions.

This function returns 1 if the value of *input1* is LESS than *input2.value*. Otherwise the returned value is 0.

If both inputs are not constant values, the returned quality flag is *input1.qFlag* | *input2.qFlag*, and the output timestamp is equal to the most recent of both inputs.

If only one of the inputs is not a constant value, the returned quality flag and timestamp are equal to this input values for quality and timestamp.

If both inputs are constants, the returned quality and timestamp are equal to the second input values for quality and timestamp.

Conditional operators have the lowest precedence between the operators.

This function is implemented according to the following pseudo-code:

```
If(input1.value < input2.value)
    → output.value = 1;
Else
    → output.value = 0;
If (input1.type != CONST)
    → output.qFlag = input1.qFlag && output.tSpec = input1.tSpec
If(input2.type != CONST) && input2.timestamp > input1.timestamp)
    → output.qFlag |= input2.qFlag
    → if(input2.tSpec > input1.tSpec)
        → output.tSpec = input2.tSpec
Else
    → output.qFlag = input2.qFlag && output.tSpec = input2.tSpec
```

If any of the input values are equal to NAN, the result value will be set as NAN (0 at status or command) with invalid output quality flag.

---

### Example

**(*input1* + *input2*) \* 4 < MIN(4,*input3*)**

- Given that *input1*, *input2* and *input3* are coreDb point names, the function will return, once each parameter gets its values:

Firstly, on one side, it calculates the value of *input1.value*+*input2.value*, and then this result is multiplied by 4. On another side, *MIN(4,input3)* is calculated. Both sides are compared, and if the first one is the lowest, the *output.value* is equal to 1. If not, *output.value* is equal to 0.

*output.qFlag* = *input1.qFlag* | *input2.qFlag* | *MIN(4, input3).qFlag*

In this case, the output quality flag is adding the quality flag values from each input, with type different to constant. The last one, is the returned value of *MIN(4, input3)* quality flag returned.

The *output.tSpec* will be equal to the most recent between *input1.tSpec*, *input2.tSpec* and the returned timestamp from *MIN(4, input3)* calculation



---

## EQUAL CONDITIONAL OPERATION (*input1==input2*) → return: *output*

Where:

- Where *input1* and *input2* are dbValues defined by expressions.

This function returns 1 if the value of *input1* is equal to *input2.value*. Otherwise the returned value is 0.

If both inputs are not constant values, the returned quality flag is *input1.qFlag* | *input2.qFlag*, and the output timestamp is equal to the most recent of both inputs.

If only one of the inputs is not a constant value, the returned quality flag and timestamp are equal to this input values for quality and timestamp.

If both inputs are constants, the returned quality and timestamp are equal to the second input values for quality and timestamp.

Conditional operators have the lowest precedence between the operators.

This function is implemented according to the following pseudo-code:

```
If(input1.value == input2.value)
    → output.value = 1;
Else
    → output.value = 0;
If (input1.type != CONST)
    → output.qFlag = input1.qFlag && output.tSpec = input1.tSpec
If(input2.type != CONST) && input2.timestamp > input1.timestamp)
    → output.qFlag |= input2.qFlag
    → if(input2.tSpec > input1.tSpec)
        → output.tSpec = input2.tSpec
Else
    → output.qFlag = input2.qFlag && output.tSpec = input2.tSpec
```

If any of the input values are equal to NAN, the result value will be set as NAN (0 at status or command) with invalid output quality flag.

---

### Example

**input1 == 3**

- Given that *input1* is coreDb point name, this function will return 1 if *input1.value* is equal to 3. If not, the returned value is equal to 0:

*output.qFlag* = *input1.qFlag*

*output.tSpec* = *input1.tSpec*

---

## COREDB POINT NAME

Formula binController also accepts a coreDb point name as a valid coordinate. In this case, the calculation function consists on copy the value of this coreDb point and refresh it when is needed.

The binController firstly look for the coreDb point name at *Status*, secondly at *Command*, followed by *Analog* and finally *SetPoint* type.

If the quality flag from the coreDb point name has the *Not Topical* or *Invalid* (local or remote), the *Invalid Local* quality flag will be set to 1.

If its value is equal to NAN, the value copied to formula variable is NAN (0 at status or command) with invalid quality flag and current local timestamp.

## VARIABLES

There are two special variable names used in the Formula Controller. They are:

### FORM\_PERIOD

---

This variable name will get the value of the configured maximum entry time between cycles.

NOTICE
In this version, this time is fix to 0 and cannot be changed in Easergy Builder. This functionality will be implemented in following versions.

Its quality flag is equal to 0 (GOOD) and its timestamp is the date of the Formula Controller initialization.

### FORM\_CYCLETIME

---

This variable name gets the last duration of the Controller entry calculation. Its unit is in milliseconds.

Its quality flag is equal to 0 (GOOD) and its timestamp is the local date of the end of the last formula calculation cycle time.

If any of these variable names are set up as coreDb point names, the Formula Controller will get them as a normal coreDb point name variable.

## CONSTANT VALUE

Formula Controller accepts a constant value (integer or float, positive or negative) as coordinate. In this case, the coordinate numeric value will be copied as its value.

<b>AB_AC</b>	Direct input acquisition block.
<b>AB_AI</b>	Analog input acquisition block (Saitel DR)
<b>AB_DI</b>	Digital input acquisition block (Saitel DR)
<b>AB_DIDO</b>	Digital input and output acquisition block
<b>AB_DO</b>	Digital output acquisition block (Saitel DR)
<b>AB_MIO</b>	Multiple inputs and outputs acquisition block.
<b>BaseLine</b>	Schneider Electric software platform for Saitel.
<b>CATconfig Tool</b>	Previous configuration included in the BaseLine Software Platform.
<b>chan</b>	Channels module's physical name.
<b>claq</b>	Saitel DR local acquisition controller's physical name.
<b>coreDb</b>	Real Time Database.
<b>CPU</b>	Central Processing Unit.
<b>DB</b>	DataBase.
<b>DD</b>	Day.
<b>DI</b>	Digital input.
<b>DLL</b>	Dynamic Link Library.
<b>dnpe</b>	Slave DNP 3.0 controller's physical name.
<b>dnpm</b>	Master DNP 3.0 controller's physical name
<b>DO</b>	Digital output.
<b>DPI</b>	Dots Per Inch. Measure of spatial printing or video dot density.
<b>EN</b>	English language.
<b>FTP</b>	File Transfer Protocol
<b>GMT</b>	Greenwich Mean Time
<b>GPS</b>	Global Positioning System
<b>HH</b>	Hour.
<b>HU</b>	Head unit.
<b>HU 250</b>	Easergy T300 CPU.
<b>HU_A</b>	Advanced head unit.
<b>HU_AF</b>	Advanced head unit with acquisition.
<b>HU_B</b>	Basic head unit.
<b>HU_BI</b>	Basic head unit with acquisition.
<b>i1e</b>	Slave iec101 controller's physical name.
<b>i1m</b>	Master iec101 controller's physical name.

<b>i4e</b>	Slave iec104 controller's physical name.
<b>i4m</b>	Master iec104 controller's physical name.
<b>IED</b>	Intelligent Electronic Device.
<b>I/O</b>	Input / Output.
<b>IP</b>	Internet Protocol.
<b>isg</b>	ISaGRAF controller's physical name.
<b>LAN</b>	Local Area Network.
<b>laq</b>	Saitel DP local acquisition controller's physical name.
<b>Login</b>	Operation which allows a user through a name and password, access a session on the tool.
<b>mdbe</b>	Slave modbus controller's physical name.
<b>mdbm</b>	Master Modbus controller's physical name.
<b>mm</b>	Millimeter.
<b>MM</b>	Month.
<b>NN</b>	Minute.
<b>NVRAM</b>	Non Volatile Random Access Memory.
<b>OS</b>	Operating System.
<b>PC</b>	Personal Computer.
<b>PLC</b>	Programmable Logic Controller.
<b>PTP</b>	Precise Time Protocol.
<b>RAM</b>	Random Access Memory.
<b>Rev</b>	Revision.
<b>RTDB</b>	Real Time DataBase.
<b>RTC</b>	Real Time Clock.
<b>RTOS</b>	Real Time Operating System. An operating system is considered to be in real time if it can guarantee that the time of execute a given task always lies within a specified constraint.
<b>RTU</b>	Remote Terminal Unit
<b>SCADA</b>	Supervisory Control and Data Acquisition
<b>SM_AI16</b>	16 analog input module.
<b>SM_AI8AO4</b>	8 analog input and 4 analog output module.
<b>SM_CPU866</b>	Saitel DP CPU
<b>SM_CPU866e</b>	Saitel DP High-Performance CPU.
<b>SM_DI32</b>	32 digital input module.
<b>SM_DO16R</b>	16 digital output to relay module.
<b>SM_DO32T</b>	32 digital output to transistor module.
<b>SNTP</b>	Simple Network Time Protocol.
<b>soe</b>	SOE module's physical name.

<b>SP</b>	Spanish language.
<b>SS</b>	Second.
<b>sup</b>	Supervision module's physical name.
<b>TCP</b>	Transport Control Protocol.
<b>TFTP</b>	Trivial File Transfer Protocol.
<b>thm</b>	Synchronization module's physical name.
<b>UDP</b>	User Datagram Protocol.
<b>UTC</b>	Universal Time Coordinated.
<b>vX.X</b>	X.X version.
<b>Webtool</b>	Maintenance and monitoring tool of the BaseLine Software Platform. Evolution of CATweb Tool.
<b>Windows</b>	PC operating system with a powerful graphic interface.
<b>XML</b>	eXtensible Markup Language.
<b>YY</b>	Year.



## BaseLine Software Platform

[www.schneider-electric.com](http://www.schneider-electric.com)

### Schneider Electric

C/ Charles Darwin s/n  
Parque Científico y Tecnológico de la Cartuja  
Seville, Spain

Phone: +34 95 492 09 92  
Fax: +34 95 541 33 75  
E-mail: [es-infoSaitel@schneider-electric.com](mailto:es-infoSaitel@schneider-electric.com)

© 2016 All rights reserved. The information contained in this document is confidential and is owned by Schneider Electric. It cannot be copied or distributed in any way, unless there is express written authorization by Schneider Electric. Although this information was verified at the time of publication, may be subject to change without notice.

Rev. 1.3 - July 2016