Hadron Xtorm
User Manual

MU223600 Rev. H

November 7, 2024

COPYRIGHTS

Nexto, MasterTool, Grano and WebPLC are the registered trademarks of Altus Sistemas de Automação S.A.

Windows, Windows NT and Windows Vista are registered trademarks of Microsoft Corporation.

OPEN SOURCE SOFTWARE NOTICE

To obtain the source code under GPL, LGPL, MPL and other open source licenses, that is contained in this product, please contact opensource@altus.com.br. In addition to the source code, all referred license terms, warranty disclaimers and copyright notices may be disclosed under request.

Contents

altus

altus

altus

altus

altus

altus

altus

altus

1. Introduction

1.1. Hadron Xtorm Series

The automation of electric power systems is characterized by the use of robust, reliable, and high-tech equipment and devices with the ability to operate in hostile environments, where there are significant levels of electromagnetic interference and exposure to higher operating temperatures. This is the reality of applications in hydroelectric power plants (HPPs), electricity substations, and wind farms, among others.

In this context, the Hadron Xtorm Series is an innovative Remote Terminal Unit (RTU), perfect for applications in electricity generation, transmission, and distribution. The Series has an ideal set of features with high performance and facilities for the various stages in the life cycle of an application, to reduce engineering, installation, and commissioning costs and minimize downtime and system maintenance when in operation. With intuitive and user-friendly interfaces, precise and intelligent diagnostics, a modern and robust design, and several innovative features, Hadron Xtorm exceeds the requirements of applications in this market.

The Series has an intelligent and versatile architecture, offering modularity in input and output (I/O) points, redundancy options, hot-swapping of modules, high-speed communication protocols such as IEC 61850 and DNP3, implementation of logic in compliance with the IEC 61131-3 standard and time synchronization.

Figure 1: Hadron Xtorm Series – Overview

1.1.1. Modules List

Below is the complete list of modules. For more information, please refer to the product documentation for each module.

1.1.1.1. CPUs - Central Processing Units

- **HX3040:** High-speed CPU, 6 Ethernet ports, 2 serial channels, memory card interface, bus expansion support and redundancy support

1.1.1.2. Input Modules

- **HX1100:** 32 DI 24 Vdc Module w/ Time Stamping
- **HX1120:** 32 DI 125 Vdc Module w/ Time Stamping
- **HX6000:** 16 AI Voltage/Current Module
- **HX6020:** 8 AI Temperature (RTD) Module

1.1.1.3. Mixed I/O Modules

- **HX6065:** AC Measurement / 4O Voltage/Current Mixed Module

1.1.1.4. Output Modules

- **HX2200:** 16 DO Relay Module
- **HX2300:** 16 DO 24 Vdc Relay Module w/ 8 CBO
- **HX2320:** 16 DO 125 Vdc Relay Module w/ 8 CBO

1.1.1.5. Power Supply Modules

- **HX8300:** 60 W 24 Vdc Redundant Power Supply
- **HX8320:** 60 W 125 Vdc Redundant Power Supply

1.1.1.6. Backplane Racks

- **HX9001:** 9-position Rack
- **HX9003:** 18-position Rack

1.1.1.7. Software

- **HD8500/ADV:** MasterTool Xtorm

1.1.1.8. Accessories

- **HX9102:** Backplane Connector Cover
- **HX9405:** 04-terminal Connector
- **HX9401:** 06-terminal Connector
- **HX9402:** 10-terminal Connector
- **NX9202:** RJ45-RJ45 2 m Cable
- **NX9205:** RJ45-RJ45 5 m Cable
- **NX9210:** RJ45-RJ45 10 m Cable
- **NX9101:** 32 GB microSD memory card with miniSD and SD adapters

1.1.2. Innovative Features

Hadron Xtorm Series brings several innovations in system usage, supervision and maintenance. These features were developed focusing on a new concept in the automation of hydroelectric power plants, substations and other applications of this segment. The list below shows some new features that users will find in the Hadron Xtorm Series:

Battery Free Operation: Hadron Xtorm Series does not require any kind of battery for memory maintenance and real time clock operation. This feature is extremely important because it reduces the system maintenance needs and allows the use in remote locations where maintenance can be difficult to be performed. Besides, this feature is environmentally friendly.

Multiple Block Storage: Several kinds of memories are available to the user in Hadron Xtorm Series CPUs, offering the best option for any user needs. These memories are divided in volatile memories and non-volatile memories. For volatile memories, Hadron Xtorm Series CPUs offer addressable input (%I), addressable output (%Q), addressable memory (%M), data memory and redundant data memory. For applications that require non-volatile functionality, Hadron Xtorm Series CPUs bring retain addressable memory (%Q), retain data memory, persistent addressable memory (%Q), persistent data memory, program memory, source code memory, CPU file system (doc, PDF, data) and memory card interface.

One Touch Diag: One Touch Diag is an exclusive feature that Hadron Xtorm Series brings to PLCs. With this new concept, the user can check diagnostic information of any module present in the system directly on CPU's graphic display with one single press in the diagnostic switch of the respective module. OTD is a powerful diagnostic tool that can be used offline (without supervisor or programmer), reducing maintenance and commissioning times.

OFD – On Board Full Documentation: Hadron Xtorm Series CPUs are capable of storing the complete project documentation in its own memory. This feature can be very convenient for backup purposes and maintenance, since the complete information is stored in a single and reliable place.

ETD – Electronic Tag on Display: Another exclusive feature that Hadron Xtorm Series brings to PLCs is the Electronic Tag on Display. This new functionality brings the process of checking the tag names of any I/O pin or module used in the system directly to the CPU's graphic display. Along with this information, the user can check the description, as well. This feature is extremely useful during maintenance and troubleshooting procedures.

1.1.3. Main Features

1.1.3.1. CPUs

The CPU features many integrated functions, online programming, high memory capacity, six Ethernet ports, two serial channels, one input port and one output port for the time synchronization signal. The six Ethernet ports are available for configuration and programming, use on IEC 61850, DNP3, IEC 60870-5-104 networks, MODBUS TCP, PROFINET and embedded web server. In addition, the CPUs have two serial interfaces for connecting local HMIs and use in MODBUS RTU networks, a memory card slot for storing application source code and program updates, a port for receiving the IRIG-B time synchronization signal and another output port for this signal, used so that the CPU can perform time synchronization of other equipment.

1.1.3.2. Modules

The modules present high density I/Os. Each I/O module has a display for local diagnostics, which shows the status of I/O each point. There are also multi-functional diagnostics on the status of the modules. All diagnostic information can also be accessed remotely by the CPU, communication protocols or through the MasterTool Xtorm configuration tool.

1.1.3.3. High Speed Bus

Hadron Xtorm Series architecture features a state-of-the art bus based on Ethernet 100 Mbps. The high throughput allows the updating of large amounts of inputs in a short period of time. The modules are automatically addressed and identified avoiding eventual errors during the application configuration and field maintenance. The bus provides special features that allow CPU redundancy in the same rack, among other features.

- Automatic addressing and identification module
- Hot swapping
- Serial Bus based on Ethernet 100 Mbps
- Time synchronization for I/O updating or precise time stamp
- Single chip hardware solution

1.1.3.4. Hot Swapping

The hot-swap feature allows the modules replacement without the system powering off. The CPU tracks the whole process and the modules can be replaced whenever necessary.

1.1.3.5. High Availability

Hadron Xtorm Series offers several different redundancy architectures, where CPUs, Power Supplies and Network Interfaces can be mounted in a redundant application. Through this flexibility, the system can be adjusted from simple systems with no redundancy to complex and critical applications where high availability is essential.

1.1.3.6. Enhanced Diagnostics

Each module contains its own diagnostics. CPUs, Network Interfaces, Power Supplies and I/O modules show several diagnostics. Each module has a multifunctional display to report its status. In addition, each module offers a button on its front to provide different diagnostic information for maintenance staff. These diagnostics can be monitored through displays or via the configuration tool. Examples:

- Module placed in the wrong position in the rack
- No power supply
- Short circuit in the outputs
- No need for configuration for modules in normal operation
- Tag view and I/O descriptions of the IO
- IP address view

1.1.3.7. Capacities

In Hadron Xtorm Series, the largest rack can hold up to 18 modules. The combination between the chosen modules must not exceed the current limit of the rack power supply. The current consumed by each Hadron Xtorm Series module from the bus is found in the Technical Characteristics document. The user may check the MasterTool Xtorm "Configuration and Consumption" functionality in order to get information about the following items: architecture assembly with the desired modules, current consumption of each module, total current required for the selected modules and the value provided by the power supply. With this architecture, a single CPU can control up to 512 I/O points through only one rack. The user can expand up to 16 racks (main rack + 15 expansion racks) using the bus expansion functionality. In this case, the maximum limit of modules, counted among all the expansion racks and the main rack, is 100.

1.1.3.8. Robustness

The Hadron Xtorm Series design is extremely robust and allows the use in applications with harsh environments. Able to be installed in environments with presence of mechanical vibration and extended operation temperature, the Series is qualified for applications in power plants, in powerhouses or near large gates. Finally, it has high requirements for immunity to electrostatic discharges and electromagnetic noise commonly present in these applications. Its design offers these possibilities, without compromising installation and maintenance procedures.

1.1.3.9. CPU Programming & Firmware Updating

Hadron Xtorm Series allows CPU programming and firmware updating via CPU's Ethernet port. This approach offers some features such as:

- Multifunctional Ethernet port used for program share, peer to peer data exchange, third-party device protocol at the application layer, network variable data exchange, etc.
- Direct access to CPU local variables
- Remote access via Ethernet interface
- Firmware update via Ethernet interface

1.1.4. Software Features

1.1.4.1. HD8500 – MasterTool Xtorm

MasterTool Xtorm is the software for configuration, programming, simulation, monitoring and debugging of the Hadron Xtorm Series. Based on the concept of integrated tool, MasterTool Xtorm provides flexibility and ease of use, allowing users to import data from electronic spreadsheets for module parameterization or variable mapping in the communication protocols available in the CPU.

Among the integrated protocols and services, there are MODBUS RTU, MODBUS TCP, DNP3, IEC 60870-5-104, IEC 61850 (MMS and GOOSE Server), time synchronization, and event grouping that can be graphically performed.

MasterTool Xtorm also offers all the editors defined in the IEC 61131-3 standard for application development: Structured Text (ST), Sequence Function Chart (SFC), Functional Block Diagram (FBD), Ladder Diagram (LD), Instruction List (IL) and Continuous Functional Chart (CFC). All editors have been specially developed to ensure the best choice for users depending on the application and their automation profile and technical culture.

Main Features:

- Programming languages based on IEC 61131-3
- Graphical editors for project and hardware configuration
- Object-oriented programming
- Simulation
- Integrated user documentation and help files
- Advanced diagnostics
- Visualization using the concept of tabs (Docking View technology)
- Integration of Logical Nodes (IEC 61850) with the IEC 61131-3 language
- Function library for hydropower plants and substations

1.1.4.1.1. IEC 61131-3 Programming Languages

MasterTool IEC XE offers all editors defined in the IEC standard for application development: Structured Text (ST), Sequential Function Chart (SFC), Function Block Diagram (FBD), Ladder Diagram (LD), Instruction List (IL) and Continuous Function Chart (CFC).

All editors were specially designed to ensure optimal handling. Ideas and suggestions from experienced users are incorporated into the development process.
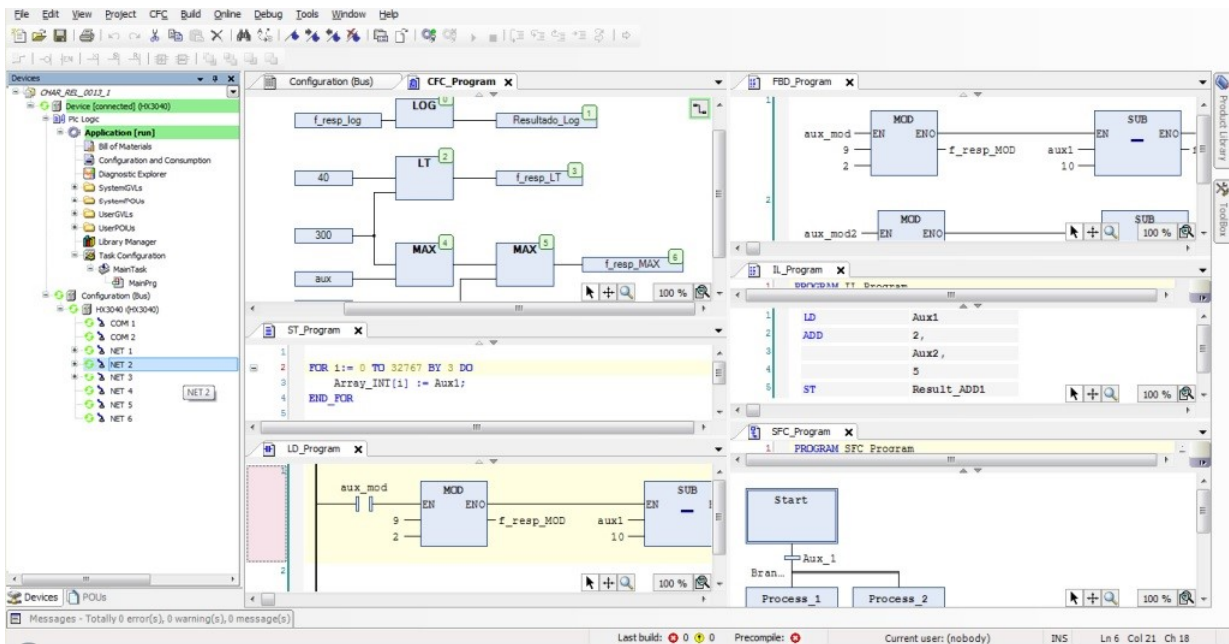
Figure 2: IEC 61131-3 Programming Languages

Some examples:

- When working in FBD, LD or IL you can freely switch between these editors
- Language elements can either be entered directly or dragged into the editor from a tool box
- HD8500 offers an intelligent input assistance and an extended IntelliSense functionality
- Standard language constructs (IF statements, FOR loops, variable classes, etc.) can be folded and unfolded in the text editors
- Language constructs are automatically created (IF ... END_IF)
- The SFC editor can either be used as defined in the standard or in a simplified version
- A comfortable time monitoring for steps as well as online diagnosis functionality is also available in the SFC editor

1.1.4.1.2.   Editors for Design Configuration and Hardware Configuration

With the help of special editors, a project can be easily configured in MasterTool Xtorm. The graphical tool allows a fast and user-friendly way to configure the system. Additionally, the user has a complete view of the application architecture with the physical position and information of each module.

The configuration of network interfaces and standard communication protocols such as MODBUS, DNP3, IEC 60870-5-104, and the IEC 61850 protocol are integrated into the programming tool. This feature allows the user to define all configuration parameters in one place, without having to use different software tools.

If necessary, the user can use the ability to import data from spreadsheets for parameterization, retrieval, or comparison of project mapping tables by simply selecting and dragging the spreadsheet tables into the editors' configuration tabs.
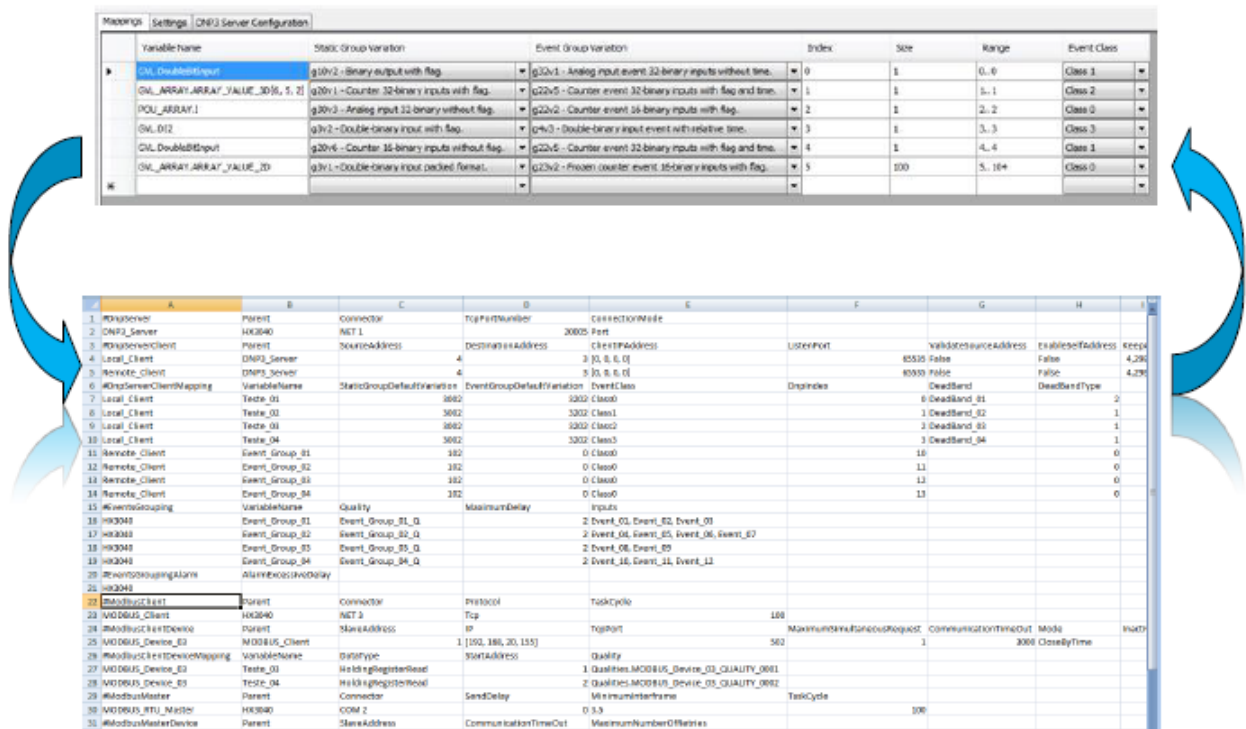
Figure 3: Editors for Project Configuration and Hardware Configuration

1.1.4.1.3.   Object Oriented Programming

MasterTool Xtorm offers object-oriented programming with the known advantages of modern high-level languages such as JAVA or C++: classes, interfaces, methods, inheritance, polymorphism, etc. Object-oriented programming offers great advantages to the user, for example when you want to reuse existing parts of an application, or when working on an application with several developers.

1.1.4.1.4. Online, Debugging and Commissioning Features

The code generated from the application is sent to the device with a single mouse click. Once MasterTool Xtorm is online, several important functions are available for fast and efficient debugging, as well as for testing and commissioning.
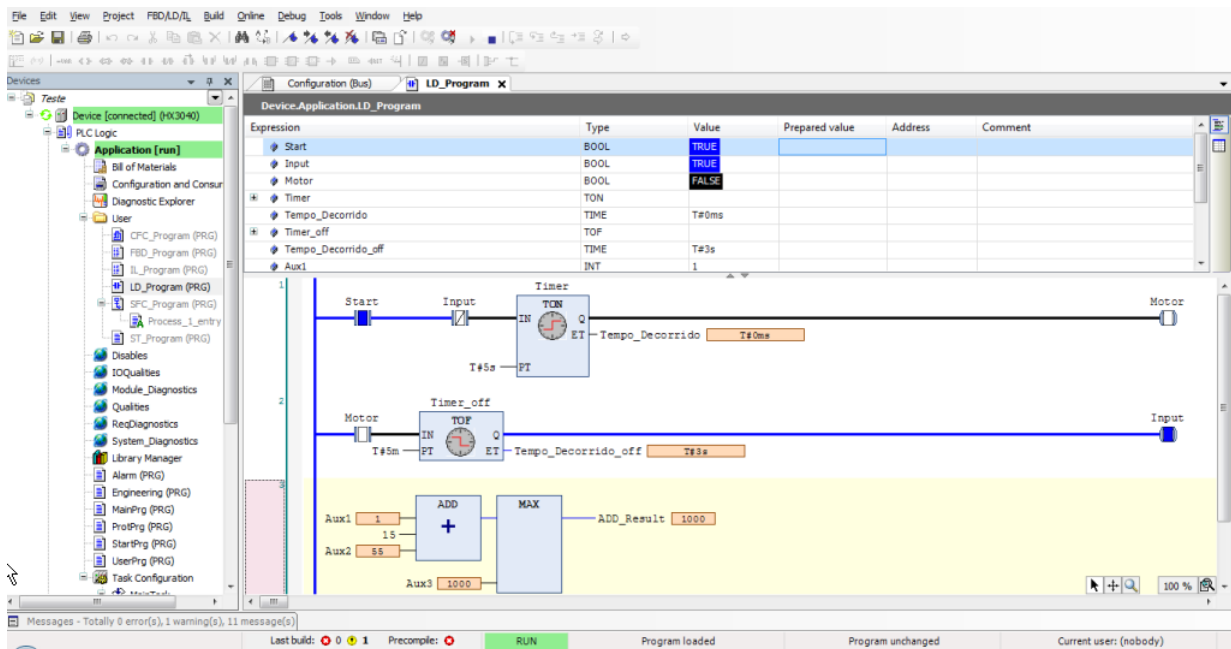
Figure 4: Online, Debugging and Commissioning Features

The values of declared variables, for example, are displayed in the program code. These values can be changed or forced without any difficulty. By defining breakpoints and traversing them through the line of code (step by step) errors can be easily detected. Breakpoints, in MasterTool Xtorm, can also be assigned to certain conditions to give more precision to the debugging process. In a single cycle operation, the user can follow the execution of the application through one complete cycle.

If the application is modified, only the changes are compiled, loaded and then activated, without the need to stop the controller and risk losing variable values. Changes to multiple POUs (Program Organization Units), variables or data types are also possible. This functionality is called Online Change.

Trace is a powerful feature that allows the user to analyze application data graphically. Thus, making it possible to easily obtain information such as trends and temporal dynamics of the application data.

1.1.4.1.5. Simulation

One feature that allows the user to evaluate and test various logics and algorithms is the simulation tool. This feature allows user applications to be designed and tested without the need for a connected RTU. This is also interesting for training, documentation, and evaluation of test cases. Since this is a simulator, naturally there may be some limitations in application development compared to the real RTU.

1.1.4.1.6. User Documentation & Help Files

Since programming the Remote Terminal Unit according to IEC 61131-3 languages is a complex task, MasterTool Xtorm offers an extensive help file with various tips and descriptions to guide and serve as a first troubleshooting database for the user in creating logic code or using the software features. This help file is available in different languages, depending on the installation options.

MasterTool Xtorm also offers multiple language support, allowing the user to select their preferred language from the available options.

As part as user documentation, HD8500 can print out user application documents, like bill of materials (BOM), POUs and configuration parameters.

1.1.4.1.7.   Enhanced Diagnostics

One of the innovations of the Hadron Xtorm Series is extensive diagnostic support. This idea comes from the demands of extensive and complex applications, where the correct use of each piece of information is fundamental to maintain, solve and prevent potential problems. This feature is also present in MasterTool Xtorm where the user, while connected to a running CPU, may access complete diagnostic structures through monitoring windows and web pages.

1.1.4.1.8.   Docking View

Docking View technology allows the user to customize the MasterTool Xtorm environment, according to their personal needs. This feature provides a user-friendly user interface to maximize the experience with the software tool.

1.1.4.1.9.   Integration of IEC 61850 with the IEC 61131-3 programming environment

In Mastertool Xtorm, for each IED configuration logic node (LN) assigned to the user project, the corresponding function block is automatically declared in the user logics development environment.

More details on configuration, descriptions and operation of the LNs are described in the Hadron Xtorm Series User Manual. However, it is recommended to read the IEC 61850 standard for better use of the presented logics.

1.1.5.   Examples of Applications
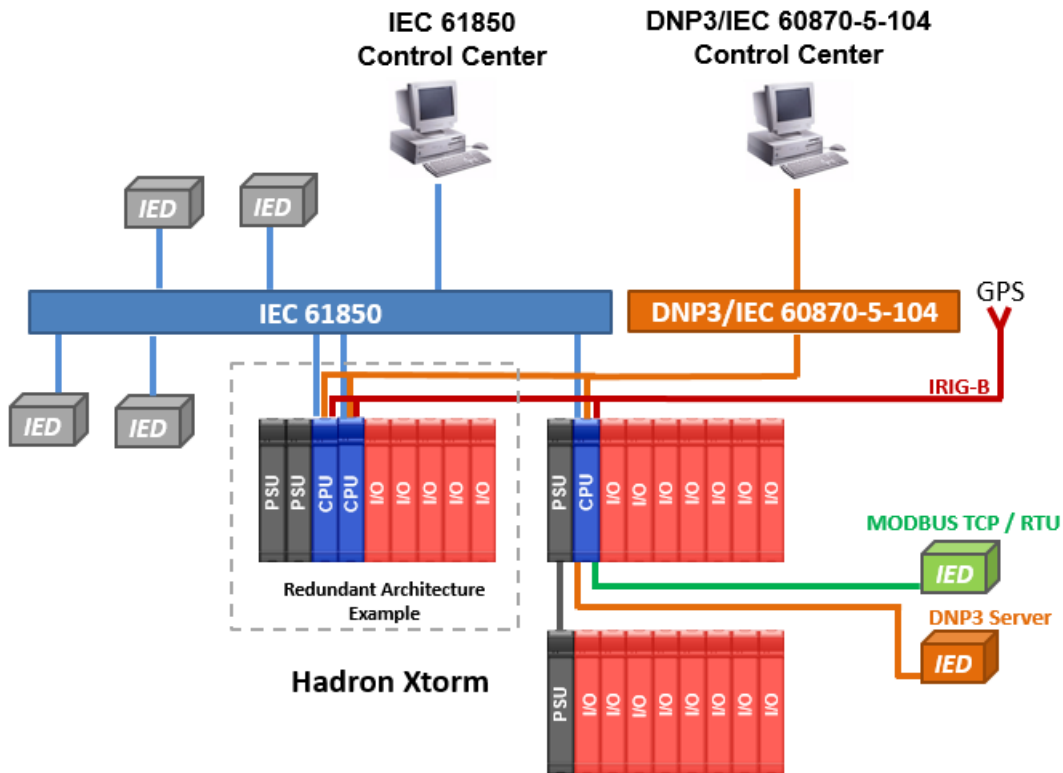
Figure 5: Application Example

Below we can see an example of a typical architecture using the Hadron Xtorm Remote Terminal Unit (RTU).

1.1.5.1.   CPU with Local I/O

This architecture is based on a single rack, called local rack. This rack consists of a CPU, a power supply module and the I/O modules required for the application, as shown below. The order of the modules must follow the configuration rules presented in the configuration tool.

This architecture is intended for small applications, where there are a low number of I/O points.

Figure 6: CPU with Local I/O

1.1.5.2. CPU with Remote I/O (Bus Expansion)

This architecture is based on a main rack (where the CPU is located) and remote racks. The communication between the local rack and the remote racks is done through the expansion ports, located on the HX8300 and HX8320 module. Each remote rack requires its own power supply module. The HX8300 and HX8320 module has two RJ45 ports, one of which is used for input data and the other for output data.

In this application example, only the output port of the local HX8300 or HX8320 module is connected, leaving the input data port open. On the other hand, in the last remote rack, it is the output data port that is left open. The remote racks in between have both ports connected: one port connected to the previous rack and the other to the next.

This architecture is intended for medium and large applications, where there are a high number of I/O points.
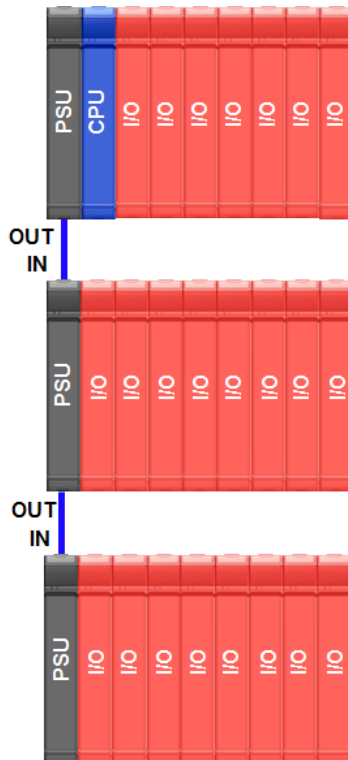
Figure 7: CPU with Remote I/O (Bus Expansion)

1.1.5.3. CPU with Remote I/O (Bus Expansion with Loopback)

Similar to the previous one, this architecture is based on a local rack (where the CPU is located) and remote racks. The communication between the local rack and the remote racks is done through the expansion ports, located on the HX8300 or

HX8320 module. The only difference with the previous architecture, is that the output data port of the last remote rack is connected to the input data port of the main rack.

This architecture allows the system to maintain access to remote rack information even in the event of a failure in the expansion cables. The CPU will detect the single failure in one of the cables and redirect the internal data paths to support this failure. In this case, a diagnostic alarm will also be generated for the user. This feature has advantages in the maintenance of the cables with the system powered up, as well as increasing the availability of the overall system. In the figure below it is possible to visualize this proposed architecture.
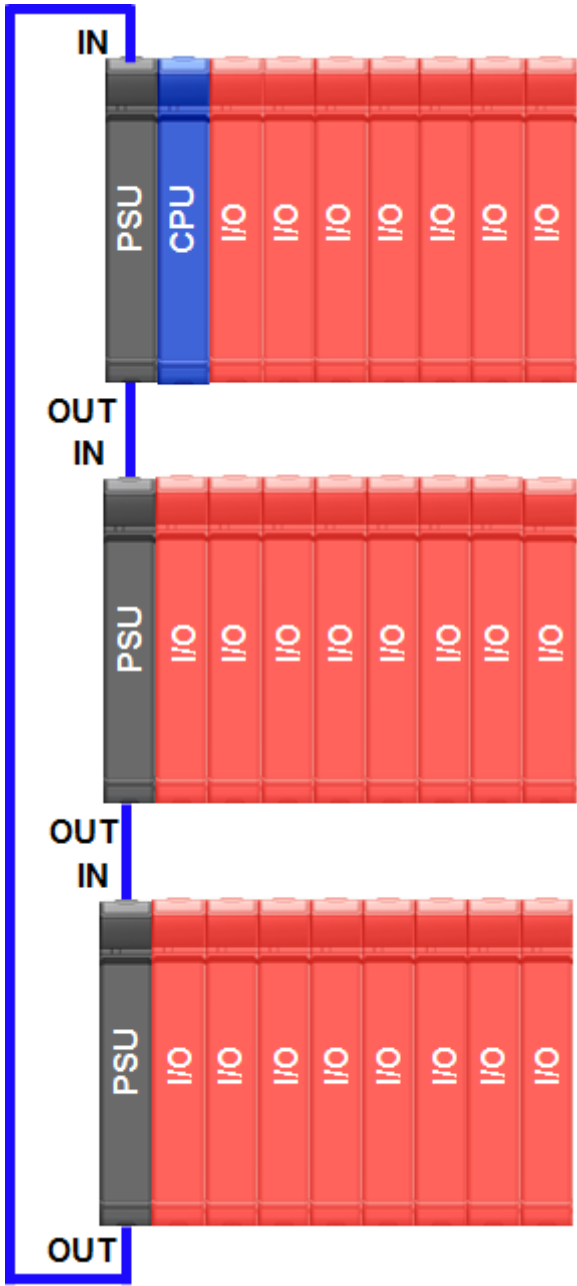
Figure 8: CPU with Remote I/O (Bus Expansion with Loopback)

1.1.5.4. CPU with Remote I/O with High Availability (Bus Expansion Redundancy with Loopback)

This architecture is based on the use of two HX8300 or HX8320 modules per rack. With two bus expansion modules, the system has a high availability, as it supports failure in the bus expansion cables or in the HX8300 or HX8320 module itself.

Similar to the previous architecture, this architecture is intended for systems where maintenance is critical and the system needs to be available for long periods. In this architecture, the racks should be assembled according to the diagram below, with the HX8300 or HX8320 modules located side-by-side in the first positions of the rack.

Note that there are unused bus expansion module ports, which should be left disconnected.
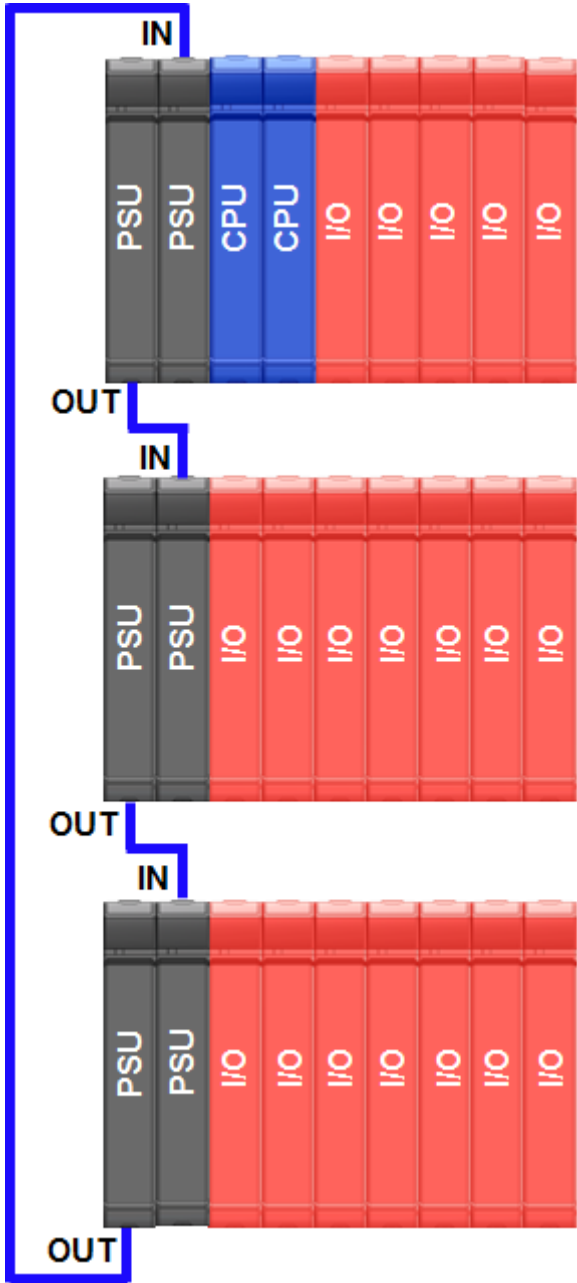
Figure 9: CPU with Remote I/O with High Availability (Bus Expansion Redundancy with Loopback)

1.1.5.5. CPU Redundancy and Power Supply Module

For critical applications, the Hadron Xtorm Series offers CPUs and Power Supply Module redundancy in the same rack, and when the user chooses to use CPU redundancy in its architecture, it must also use Power Supply Module redundancy.

In this architecture, the system will have one CPU performing the control task (active) and another remaining in reserve, in a hot-standby topology. In case of failure of the active CPU, the system will automatically perform a switchover (event where the spare CPU becomes active). This means that critical applications are no longer affected by eventual control system failures,

ensuring high availability for these applications. The results are increased efficiency, productivity, minimized downtime, and reduced maintenance time.

The configuration of the two CPUs must be identical. The active and the standby CPU must be in rack positions that support such functionality (rack positions 3 and 4). This application is easy to configure and requires no special programming or parameterization.

For Power Supply Module redundancy, if one of the modules fails, automatically the second one will take over. On each module there will be an indication of its operating status and input power supply voltage. This status can be used to detect and replace the failed module. Such status can be read by the CPU via the bus and reported to monitoring devices such as an operator terminal or Human Machine Interface (HMI) or a SCADA supervisory and data acquisition system. A failed module can be replaced during normal system operation, without requiring a power down or interruption of the application.

Figure 10: CPU Redundancy and Power Supply Module

1.1.5.6. Time Synchronization via IRIG-B

For applications where time synchronization with other equipment is required, the Hadron Xtorm Series CPU has one input port and one output port for the IRIG-B signal. Through the input port, the CPU will receive the time synchronization data and synchronize with its internal clock. And through the output port, the CPU can synchronize other equipment, retransmitting at the output port, the signal received at the input port.
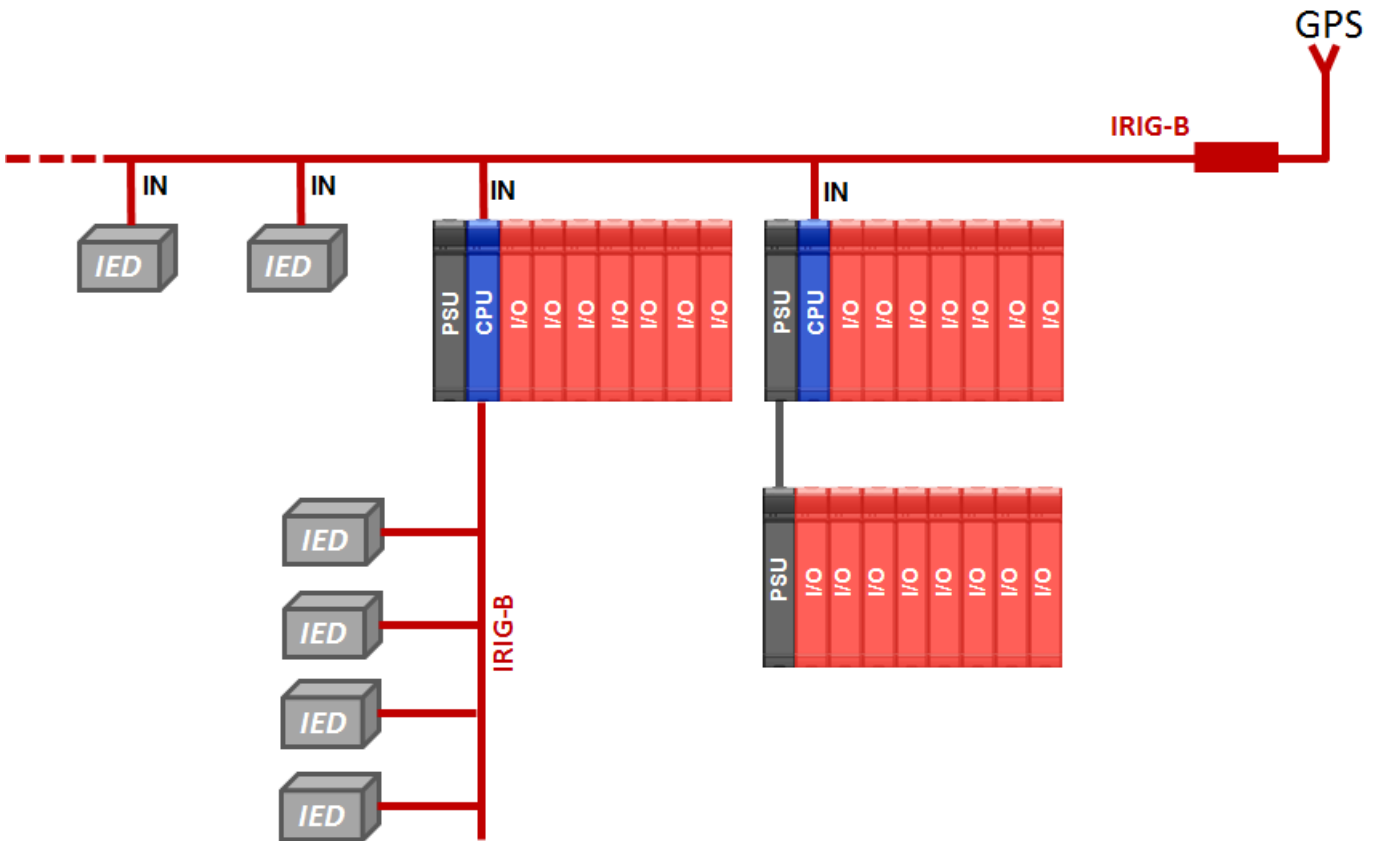
Figure 11: Time Synchronization via IRIG-B - Simple CPU

For cases of applications where CPU redundancy is used, it is convenient that the time synchronization signal be interconnected to each of the CPUs, regardless of which CPU is in active mode and which is in standby mode, thus ensuring system time synchronism. In the figure below is an example of this type of architecture.
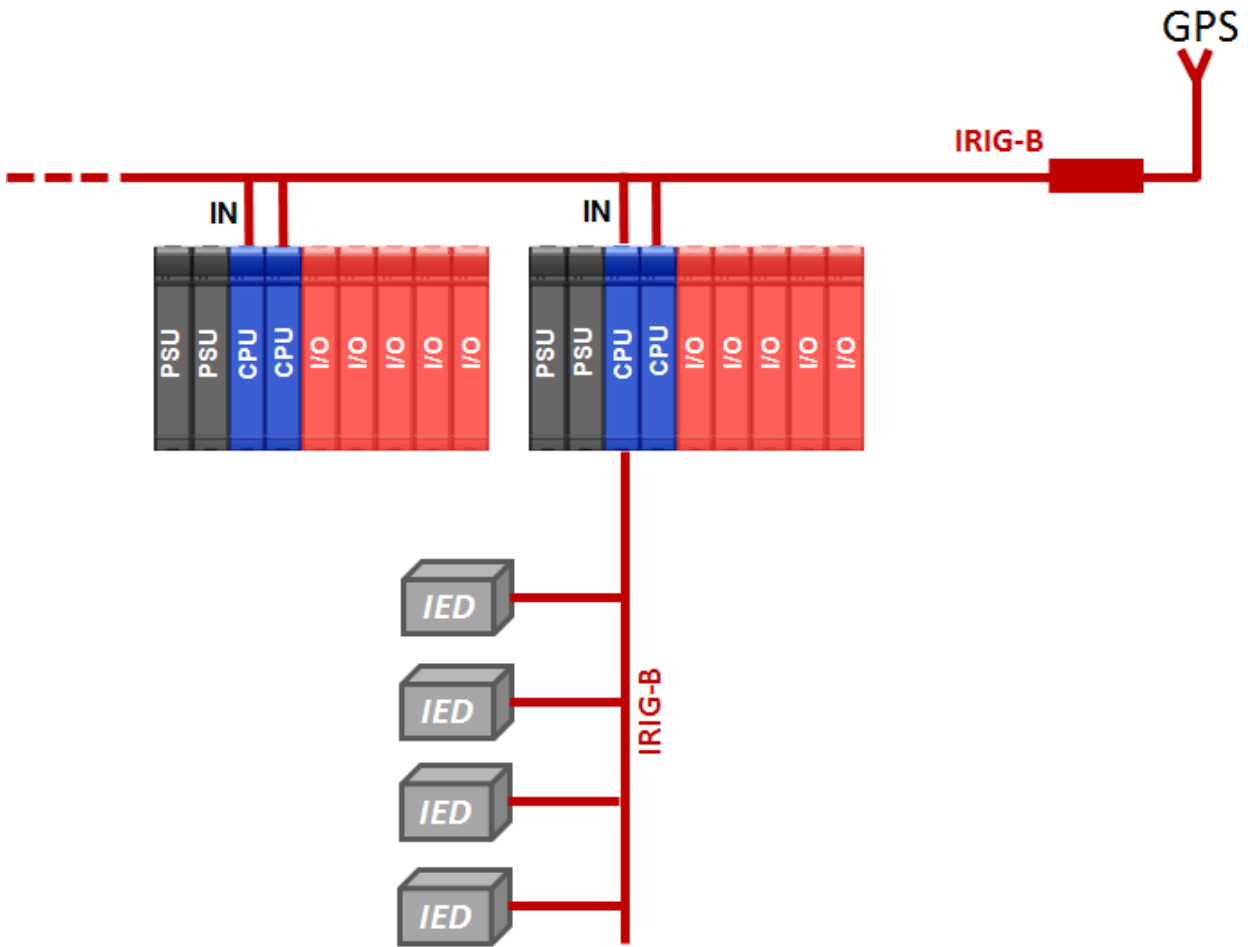
Figure 12: Time Synchronization via IRIG-B - Redundant CPU

1.1.5.7. Redundant Ethernet Networks with NIC Teaming

Each CPU can have one or more network protocols configured for communication with the control center or other field devices or equipment. For Ethernet network redundancy with NIC Teaming, two Ethernet ports of the CPU must be configured to form a redundant pair. A set of two Ethernet ports forming a NIC Teaming pair has a single IP address bound to the port pair. In this way, the control center does not have to worry about changing the IP if any of the ports in the NIC Teaming pair fails. Each of the Ethernet ports must be interconnected to different switches. If one of the ports fails, automatically the data packages will be redirected to the other port.

This Ethernet architecture enables high availability of system communication and is strongly suited for bridging faults on Ethernet ports, cables, and switches.
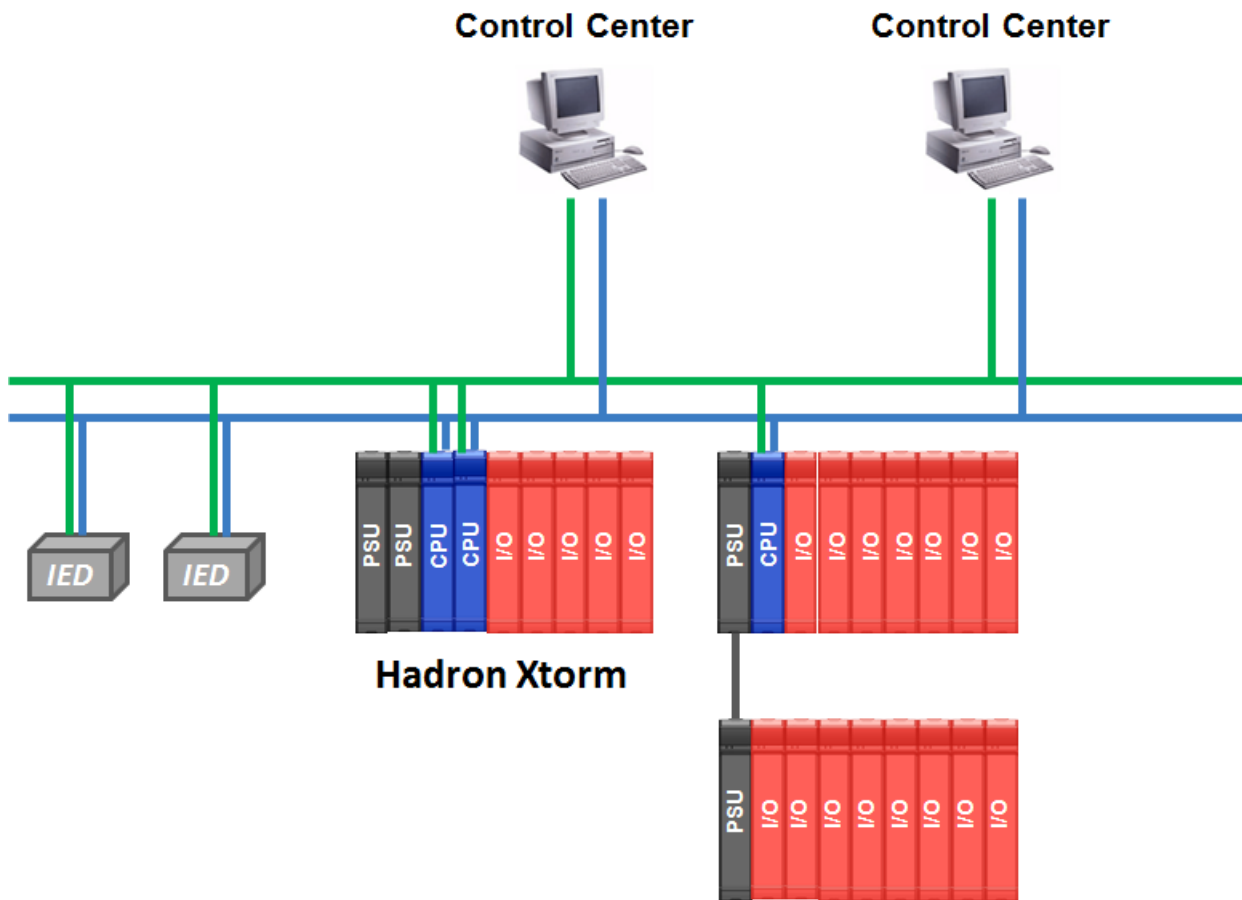
Figure 13: Redundant Ethernet Networks with NIC Teaming

1.1.5.8. Ring Mode Ethernet Networks

There is a mode of operation that turns a pair of Ethernet interfaces into a switch, where communication can be done over both ports. This makes it possible to implement a ring network topology.

In this ring, it is necessary to include an external switch that can manage it, to avoid loops that degrade network performance.

All Ethernet interfaces have individual diagnostics, making it easy to debug any problems.
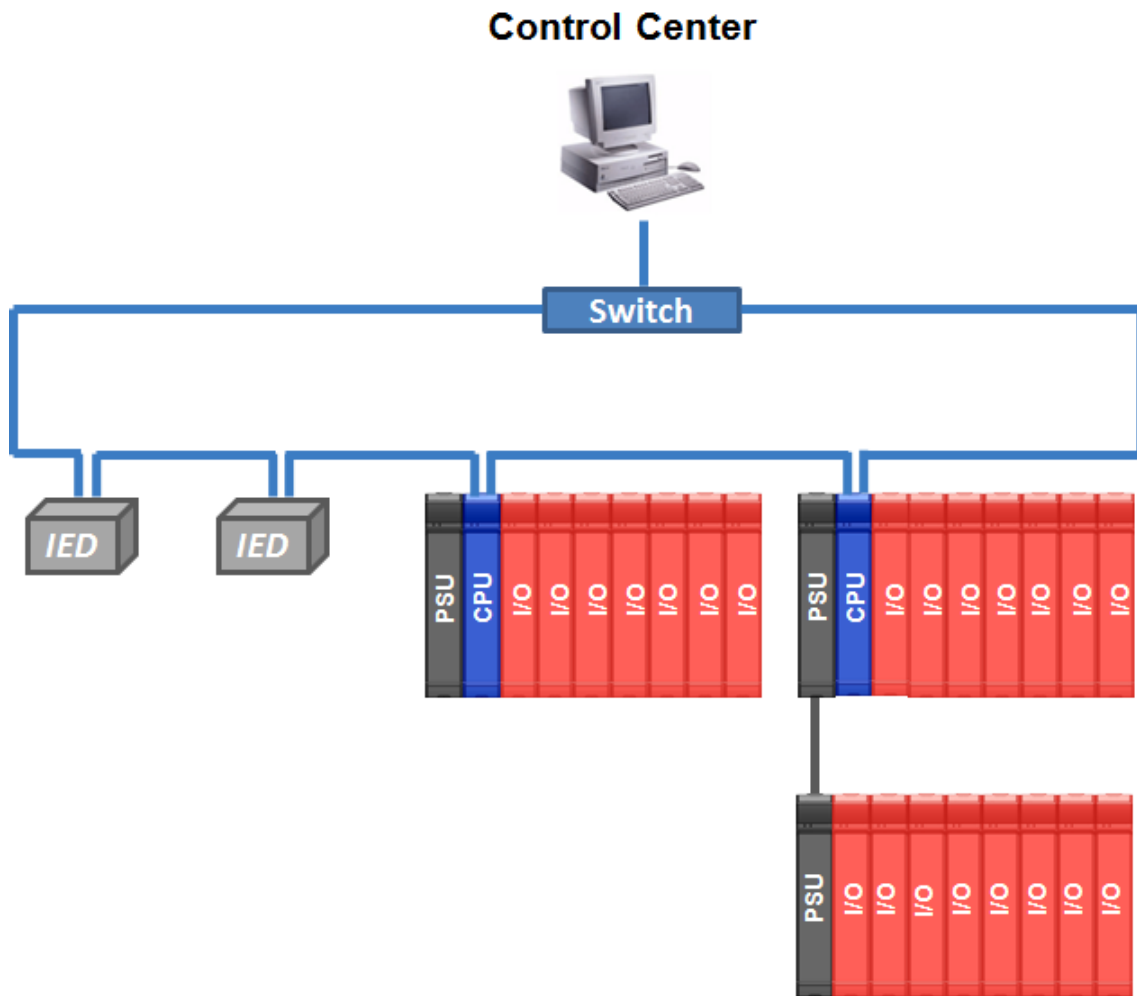
Figure 14: Ring Mode Ethernet Networks

1.1.5.9. Bus Expansion with Xtorm and Nexto

The Hadron Xtorm Series is compatible with all versions of MasterTool Xtorm, supporting bus expansion through the HX8300 or HX8320 module. With the UCP HX3040 model, it is possible to expand the architecture up to 16 racks (1 main rack and 15 expansion racks) using the bus expansion functionality. In this case, the maximum limit of modules counted among all expansion racks cannot exceed 100 modules.

In addition, it is also possible to build a hybrid architecture using the Hadron Xtorm Series bus expansion to interconnect with a Nexto or Nexto Jet Series rack. In the following figure, this proposed architecture can be visualized.
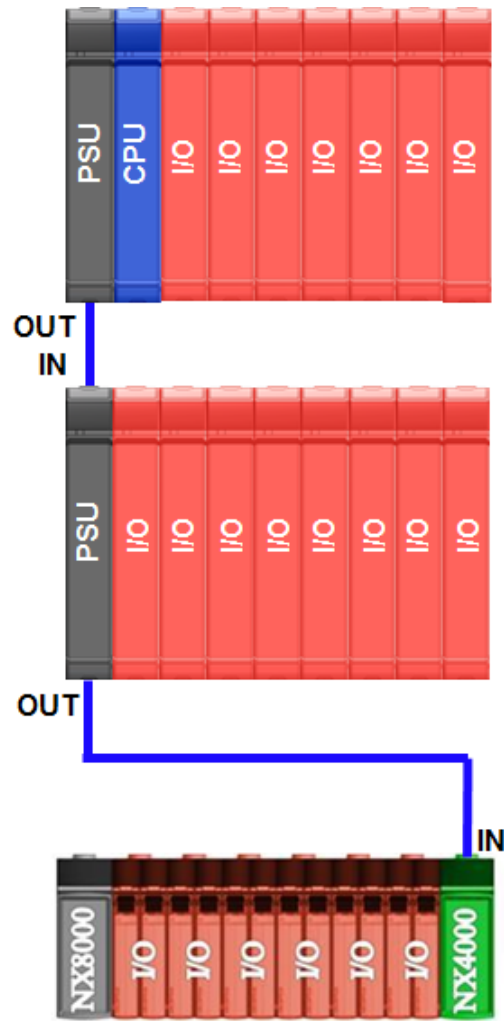
Figure 15: Bus Expansion with Xtorm and Nexto

> **ATTENTION**
>
> For mixed architecture, only Nexto Series I/O modules are allowed. Other Nexto Series modules cannot be installed on Hadron Xtorm Series mixed expansion buses.

1.2. Documents Related to this Manual

For additional information about the Hadron Xtorm Series, other documents (manuals and technical characteristics), in addition to this one, may be consulted. These documents are available in their latest revision at www.altus.com.br.

Each product has a document called Technical Characteristics where you can find the characteristics of the product.

The following documents are recommended as a source of additional information:

Code	Description	Language
CE108804	MasterTool Xtorm Technical Characteristics	English
CT108804	Características Técnicas MasterTool Xtorm	Portuguese
CE123000	Hadron Xtorm Series Technical Characteristics	English
CT123000	Características Técnicas Série Hadron Xtorm	Portuguese
CE123100	CPU 6 ETH, 2 SERIALS, IRIG-B, RED Module Technical Characteristics	English
CT123100	Características Técnicas do Módulo UCP 6 ETH, 2 SERIAIS, IRIG-B, RED.	Portuguese
CE123701	Hadron Xtorm Series Backplane Racks Technical Characteristics	English
CT123701	Características Técnicas dos Bastidores da Série Hadron Xtorm	Portuguese
CE123200	Redundant Power Supply 60 W Modules Technical Characteristics	English
CT123200	Características Técnicas dos Módulos Fonte de Alimentação Redundante 60 W	Portuguese
CE123300	Módulo 32 DI 125 Vdc w/ event log Module Technical Characteristics	English
CT123300	Características Técnicas do Módulo 32 ED 125 Vdc c/ registro de eventos	Portuguese
CE123400	16 SD Relay 125 Vdc w/ CBO Module Technical Characteristics	English
CT123400	Características Técnicas do Módulo 16 SD Relé 125 Vdc c/ CBO	Portuguese
CE123310	16 AI Voltage/Current Module Technical Characteristics	English
CT123310	Características Técnicas do Módulo 16 EA Tensão/Corrente	Portuguese
CE123313	AC Measurement / 4O Voltage/Current Mixed Module Technical Characteristics	English
CT123313	Características Técnicas do Módulo Misto Medição AC / 4S Tensão/Corrente	Portuguese
CE123311	8 AI RTD Module Technical Characteristics	English
CT123311	Características Técnicas do Módulo 8 EA RTD	Portuguese
CE123901	Hadron Xtorm Series Connectors Technical Characteristics	English
CT123901	Características Técnicas dos Conectores da Série Hadron Xtorm	Portuguese
CE123900	Rack Connector Cover Technical Characteristics	English
CT123900	Características Técnicas da Tampa para conector de bastidor	Portuguese
MU223600	Hadron Xtorm Utilization Manual	English
MU223000	Manual de Utilização Hadron Xtorm	Portuguese
MU223601	Hadron Xtorm DNP3 Server Device Profile Document	English
MU223602	Hadron Xtorm DNP3 Client Device Profile Document	English
MU223603	IEC 60870-5-104 Server Device Profile Document	English
MU223604	Hadron Xtorm IEC 60870-5-104 Client Device Profile Document	English
MU223605	Hadron Xtorm IEC 61850 Server Device Profile Document	English
MP399609	MasterTool IEC XE Programming Manual	English
MP399048	Manual de Programação MasterTool IEC XE	Portuguese
MU299609	MasterTool IEC XE User Manual	English
MU299048	Manual de Utilização MasterTool IEC XE	Portuguese

Table 1: Related Documents

altus

1.3.   Technical Support

For Altus Technical Support contact in São Leopoldo, RS, call +55 51 3589-9500. For further information regarding the Altus Technical Support existent on other places, see https://www.altus.com.br/en/ or send an email to altus@altus.com.br.

If the equipment is already installed, you must have the following information at the moment of support requesting:

- The model from the used equipments and the installed system configuration
- The product serial number
- The equipment revision and the executive software version, written on the tag fixed on the product's side
- CPU operation mode information, acquired through MasterTool IEC XE
- The application software content, acquired through MasterTool IEC XE
- Used programmer version

1.4.   Warning Messages Used in this Manual

In this manual, the warning messages will be presented in the following formats and meanings:

> **DANGER**
>
> Reports potential hazard that, if not detected, may be harmful to people, materials, environment and production.

> **CAUTION**
>
> Reports configuration, application or installation details that must be taken into consideration to avoid any instance that may cause system failure and consequent impact.

> **ATTENTION**
>
> Identifies configuration, application and installation details aimed at achieving maximum operational performance of the system.

2. Technical Description

This chapter presents all the technical features of the Hadron Xtorm HX3040 Series CPU.

2.1. Panels and Connections

The following figure shows the front panel of the HX3040 CPU.

Figure 16: CPU HX3040

As shown in the figure, at the top of the front panel is the graphical display used to show the status and diagnostics of the entire system, including module-specific diagnostics. The graphical display also offers an easy-to-use menu that gives the user a quick way to read or set some parameters such as: internal temperature (read only), contrast of the graphical display, IP address for each network interface (read only) and local time (read only).

The Hadron Xtorm Series CPU provides a graphic display containing status and useful information for the user, such as: application states (running and stopped), miniSD card status, serial interface activity (RX and TX) among others. Additionally, Hadron Xtorm Series CPUs also provide a bicolor LED used to indicate status and diagnostics. For more information about the meaning of the diagnostics LED status, please refer to the CPU technical characteristics document CE123100 - CPU 6 ETH, 2 SERIALS, IRIG-B, RED Module Technical Characteristics .

On the front panel are the interfaces for connecting the Hadron Xtorm Xtorm Series CPU. These interfaces are Ethernet communication, serial communication, memory card interface, and time synchronization interface. The table below gives a brief description of these interfaces.

altus

Interfaces	Descrição
NET 1 .. NET 6	RJ45 communication connector in the 10/100Base-TX standard. Allows point-to-point or network communication in the open protocols, MODBUS TCP client and server, MODBUS RTU via TCP Client and Server, DNP3 Client and Server, IEC 60870 5 104 Client and Server, IEC 61850 MMS Server, IEC 61850 GOOSE publish and subscriber. For more information on use, see section Ethernet Interfaces Configuration.
COM 1	DB9 female connector type for communication in RS-232C and RS-485 standard. Allows point-to-point or network communication in the open portocols, MODBUS RTU Slave or MODBUS RTU Master. For more information on use, see section Serial Interfaces Configuration.
COM 2	DB9 female connector type for communication in RS-422 and RS-485 standard. Allows point-to-point or network communication in the open protocols, MODBUS RTU Slave or MODBUS RTU Master. For more information on use, see section Serial Interfaces Configuration.
MEMORY SLOT	Connector for memory card. Allows to use a memory card for different types of data storage as: user logs, web pages, project documentation and source files. For more information on use, see section Memory Card.
IRIG-B	4-terminal block connector for communication in IRIG-B standard. For more information on use, see section IRIG-B.

Table 2: Connection Interfaces

2.2. Product Features

2.2.1. General Features

	HX3040
Programming languages	Instruction List (IL) Structured Text (ST) Ladder Diagram (LD) Sequential Function Chart (SFC) Function Block Diagram (FBD) Continuous Function Chart (CFC)
Types of tasks	Cyclic (periodic)) Event (software interruption)) External Event (hardware interruption) Continuous (free run) Status (software interruption)
Online changes	Yes
Hot swap support	Yes
Bus expansion redundancy support	Yes

	HX3040
Serial Interfaces	1 x RS-232C / RS-485 (COM 1) 1 x RS-485 / RS-422 (COM 2)
Maximum number of tasks	32
Maximum number of expansion racks	15
Ethernet TCP/IP local interface	6
Ethernet TCP/IP interfaces redundancy support	Yes
CPU redundancy support (same rack)	Yes
Real time clock (RTC)	Yes Resolution of 1 ms and maximum variance of 2 s per day
Clock synchronization (IRIG-B, SNTP and PTP)	Yes

Event queue
     Size
     Overflow policy
     Retentivity
	
	 4.500 events
	 Keeps the latest
	 Yes

Watchdog	Yes
Status and diagnostic indication	Display, LED, web page and CPU's internal memory
One Touch Diag (OTD)	Yes
Eletronic Tag on Display (ETD)	Yes

Isolation
     Logic to protective earth ⏚
     Logic to Ethernet interfaces
     Logic to serial port (COM 2)
     Logic to IRIG-B port
     Ethernet interfaces to protective earth ⏚
     Ethernet Interface to serial port (COM 2)
     Ethernet Interface to Ethernet Interface
     Serial Port (COM 2) to protective earth ⏚
	
	 2500 Vac/ 1 minute
	 1500 Vac/ 1 minute
	 2000 Vac/ 1 minute
	 2500 Vac/ 1 minute
	 1500 Vac/ 1 minute
	 2500 Vac/ 1 minute
	 1500 Vac/ 1 minute
	 2500 Vac/ 1 minute

Current consumption from backplane rack	1500 mA
Dissipation	7.5 W
Operation temperature	-5 to 60°C
Storage temperature	-25 to 75°C
Operating and storage relative humidity	5 to 96 %, no condensation
Eletronic circuit coating	Yes
Protection Leve	IP 20
Module dimensions (L x A x P)	38,0 x 235,3 x 184,2 mm
Module dimensions (L x A x P)	55,0 x 308,0 x 266,0 mm
Weight	1000 g
Weight with package	1300 g

Table 3: Main Features

Notes:

    Types of tasks: Task is an object used to call POUs. A Task can be triggered by period, events or can run in freewheeling mode. Each task can call one or more POUs.

    Real Time Clock (RTC): The retention time, which is the time that the real time clock keeps updating the date and time after the CPU goes off, is 15 days considering environments with temperature of 25 °C. Up to the maximum product operation temperature, retain time is reduced to 10 days.

altus

One Touch Diag (OTD): This option is available to the user only when the module is in operational mode.

Isolation: Logic is the name for the internal circuits such as processors, memories and interfaces with backplane rack.

Eletronic circuit coating: The eletronic circuit coating protects the electronic components inside the product from humidity, dust and other harsh elements to electronic circuits.

2.2.2. Memory

	HX3040
Addressable input variables memory (%I)	96 Kbytes
Addressable output variables memory (%Q)	96 Kbytes
Symbolic variable memory	6 Mbytes
Retain symbolic variables memory	8 Kbytes
Persistent symbolic variables memory	4 Kbytes
Redundant variable memory	512 Kbytes
Program memory	12 Mbytes
Source code memory (backup)	100 Mbytes
User files memory	32 Mbytes

Table 4: Memory

Notes:

Addressable input variables memory (%I): It is the area where all the direct addressable input variables are assigned. An addressable variable means that the variable can be accessed directly in the memory using the desired address. For example: %IB0, %IW100. Addressable input variables can be used for mapping analogic or digital input points. As a reference, 8 digital input points can be represented by one byte and 1 analogic input point can be represented by two bytes.

The Hadron Xtorm Series HX3040 CPU defines all the area of addressable input variables memory (%I) as redundant variables, which means that the user does not need to select this area.

Addressable output variables memory (%Q): : It is the area where all the addressable output variables are assigned. An addressable variable means that the variable can be accessed directly in the memory using the desired address. For example: %QB0, %QW100. Addressable output variables can be used for mapping analogic or digital output points. As a reference, 8 digital output points can be represented by one byte and 1 analogic output point can be represented by two bytes.

The Hadron Xtorm Series HX3040 CPU defines all the addressable output variables memory (%Q) as redundant variables, which means that the user does not need to select this area.

Symbolic variables memory: It is the area where the symbolic variables are assigned. Symbolic variables are IEC variables created in POUs and GVLs during the application development, not addressed directly in memory. Symbolic variables can be defined as retain or persistent. In this case the retain symbolic variables memory or the persistent symbolic variables memory area will be used, respectively. The RTU system allocates system variables in this area, so that the available space for the allocation of variables user-created is lower than those reported in the table. The amount of memory occupied by these system variables depends on the project characteristics (number of modules, drivers, etc. ...), so it is recommended to observe the available space in the MasterTool Xtorm compilation messages.

Retain symbolic variables memory: It is the area where the retain symbolic variables are assigned. Retain data keep their respective values even after a power on/power off cycle of the CPU. The complete list of when retain variables keep their values and when the value is lost can be found on the table below.

Persistent symbolic variable memory: It is the area where the persistent symbolic variables are assigned. Persistent data keep their respective values even after a download of a new application into CPU.

> **ATTENTION**
>
> The declaration and use of persistent variables should be performed exclusively through the Persistent Vars object, which may be included in the project through the treeview in Application -> Add Object -> Persistent Variables. Do not use the expression VAR PERSISTENT in the declaration field of POUs variables.

In addition to the persistent area size given in the table above, these 44 bytes are reserved for storing information about the persistent variables (not available for use). The complete list of when persistent variables keep their values and when the value is lost can be found in the following table.

altus

The table shows the behavior of the symbolic, retentive and persistent variables for different situations, where "-" means that the value is lost and "X" means that the value is kept.

Command	Symbolic Variable	Retain Variable	Persistent Variable
Reset Warm / Power on Cycle	-	X	X
Reset Cold	-	-	X
Reset Origin	-	-	-
Remove CPU or Power Supply from Rack when powered	-	-	-
Download	-	-	X
Online change	X	X	X
Reboot CPU	-	X	X
Clean All	-	-	X
Reset Process (IEC 60870-5-104	-	X	X

Table 5: Non-volatile Variables Behavior

In the case of the Clean All command, if the application has been modified in such a way that persistent variables have been removed, inserted at the beginning of the list or have had their type changed, the value of these variables will be lost (alerted by the MasterTool when downloading). It is therefore recommended that changes in the GVL of persistent variables only involve adding new variables at the end of the list.

Memory of redundant variables: Only applied to designs with CPU redundancy. These are the variables synchronized between redundant CPUs, which includes direct representation variables (%I e %Q), used by I/O modules, and user program symbolic variables declared in redundant GVLs or POUs.

Program memory: Memory area that corresponds to the maximum size allowed for the user application. This area is shared with source code memory, with the total area being the sum of "program memory" and "source code memory".

Source code memory (backup): Memory area used as a backup of the project, that is, if the user wants to import its project, the MasterTool Xtorm software will find the necessary information in this area. It is important to make sure that the project saved as a backup is up to date to avoid losing critical information. This area is shared with the program memory with the total area being the sum of "program memory" and "source code memory".

User files memory: This area of the memory is destined to the storage of files, such as: doc, pdf, images, among others, that is, it allows the recording of data as if it were a memory card.

Redundancy support (same rack): The HX3040 CPU supports redundancy of CPUs placed in the same rack.

2.2.3. Protocols

	HX3040	Interface
Communication with programming software	Yes	NET 1 .. NET 2
Open Protocol	Yes	COM 1 / COM 2
MODBUS RTU Master	Yes	COM 1 / COM 2
MODBUS RTU Slave	Yes	COM 1 / COM 2
MODBUS TCP Client	Yes	NET 1 .. NET 6
MODBUS TCP Server	Yes	NET 1 .. NET 6
MODBUS RTU over TCP Client	Yes	NET 1 .. NET 6
MODBUS RTU over TCP Server	Yes	NET 1 .. NET 6
IEC 60870-5-104 Client	Yes	NET 1 .. NET 6
IEC 60870-5-104 Server	Yes	NET 1 .. NET 6
DNP3 Client	Yes	NET 1 .. NET 6
DNP3 Server	Yes	NET 1 .. NET 6
IEC 61850 MMS Server	Yes	NET 1 .. NET 6
IEC 61850 GOOSE Publisher	Yes	NET 1 .. NET 6

altus

	HX3040	**Interface**
IEC 61850 GOOSE Subscriber	Yes	NET 1 .. NET 6
OPC DA Server	Yes	NET 1 .. NET 6
OPC UA Server	Yes	NET 1 .. NET 6
EtherCAT Master	No	-
SNMP Agent	No	-
EtherNet/IP Scanner	No	-
EtherNet/IP Adapter	No	-
MQTT Client	Yes	NET 1 .. NET 6
SNTP Client (for clock synchronism)	Yes	NET 1 .. NET 6
PTP (Precision Time Protocol) Slave	Yes	NET 1 .. NET 6
PROFINET Controller	Yes	NET 1 .. NET 6
PROFINET Device	No	-
OpenVPN Client	No	-
OpenVPN Server	No	-
FTP Server	Yes	NET 1 .. NET 6
MRP	No	-

Table 6: Protocols

Notes:

PROFINET Controller: Enabled for use without CPU redundancy and in a simple network (without a ring) with up to 8 devices. For larger applications, consult technical support.

OPC UA Server: Enabled for use without CPU redundancy and in a simple network (without a ring).

2.2.4. Serial Interfaces

2.2.4.1. COM 1

	COM 1
Connector	DB9 shielded female
Physical interface	RS-232C or RS-485 (depending on the connected cable)
Communication direction	RS-232C: full duplex RS-485: half duplex
Maximum number of RS-485 transceivers	32
RS-485 termination	No (allows the use of external active termination)
Modem signals	RTS, CTS, DCD
Baud rate	600, 1200, 1800, 2400, 4800, 9600, 19200, 38400, 57600, 115200 bps

Table 7: Serial Interface COM 1

altus

2.2.4.2. COM 2

	COM 2
Connector	DB9 shielded female
Physical interface	RS-422 or RS-485 (depending on the selected cable)
Communication direction	RS-422: full duplex RS-485: half duplex
Maximum number of RS-422 transceivers	11 (1 driver e 10 receivers)
Maximum number of RS-485 transceivers	32
Termination	Yes (optional via cable selection)
Baud rate	600, 1200, 1800, 2400, 4800, 9600, 19200, 38400, 57600, 115200 bps

Table 8: Serial Interface COM 2

Notes:

Physical interface: Depending on the cable configuration, it is possible to choose the type of physical interface: RS-232C or RS-485 to COM1, and RS-422 or RS-485 to COM2.

Maximum number of RS-422 transceivers: The maximum number of RS-422 interfaces that can be used on the same bus.

Maximum number of RS-485 transceivers: The maximum number of RS-485 interfaces that can be used on the same bus.

2.2.5. Ethernet Interfaces

2.2.5.1. NET 1 .. NET 6

	NET 1 .. NET 6
Connector	RJ45 shielded female
Auto crossover	Yes
Maximum cable length	100 m
Cable type	UTP or ScTP, category 5
Baud rate	10/100 Mbps
Physical layer	10/100 Base-TX
Data link layer	LLC (logic link control)
Network layer	IP (internet protocol)
Transport layer	TCP (transmission control protocol) UDP (user datagram protocol)

Table 9: Ethernet Interfaces NET 1 .. NET 6

2.2.6. IRIG-B

	IRIG-B
Connector type	Removable terminal connector with 4 terminals (HX9405)
Maximum cable length	5 m
Wire gauge	0,5 mm^2
Input and output level	TTL
Input impedance	>100 kΩ
Delay between input and output	< 10 ns
Maximum output current	10 mA
Maximum output load	500 Ω
Protection against short circuit	Sim
Voltage levels	0 a 1,5 Vdc to logic level 0 3,5 a 5 Vdc to logic level 1

Table 10: IRIG-B Interface

Note:

Maximum output load: The resulting total load of all devices connected to the output should not exceed this value. There is no maximum predefined limit of devices. The value should be calculated regarding the minimum input impedance of each device connected to the IRIG-B Xtorm output.

2.2.7. Graphic Display

The Hadron Xtorm Series CPUs have a graphic display used to show the status and diagnostics of the entire system, including the specific diagnostics of each additional module. The display also offers a user-friendly menu that gives the user a quick path to read or set some parameters such as: internal temperature (read-only); graphic display contrast and IP address for each NET interface (read-only).

2.2.8. Memory Card Interface

Memory cards can be used for different types of data storage such as: user logs, web pages, project documentation, and source files.

	Memory Card
Maximum capacity	32 Gbytes
Minimum capacity	2 Gbytes
Type	SD
File system	FAT32
Remove card safely	Yes, through specific menu for this function.

Table 11: Memory Card Interface Features

Notes:

Maximum capacity: The memory card capacity should be equal or below this limit for the Hadron Xtorm CPU correct functioning. The CPU may not recognize the card or data loss may occur during the transfer processes.

Maximum capacity: The memory card capacity should be equal or above this limit for the Hadron Xtorm CPU correct functioning. The CPU may not recognize the card or data loss may occur during the transfer processes.

File system: It is recommended to format the memory using the Hadron Xtorm CPU itself, otherwise performance loss may occur when accessing the memory card interface.

altus

2.3.   Compatibility with Other Products

The following table provides information regarding the compatibility of the Hadron Xtorm Series CPU versions with the Mastertool versions:

Xtorm Series CPUs	HD8500	MT8500	Firmware version
HX3040	1.00 to 1.09	-	1.0.0.0 to 1.7.58.0
HX3040	2.01 or later	-	1.9.4.0 or later
HX3040	2.15 or later	-	1.12.27.0 or later
HX3040	-	3.70 or later	1.14.36.0 or later

Table 12: Compatibility between CPUs and Mastertool

The following table brings information regarding the compatibility of Nexto Series products with Hadron Xtorm:

Product	Firmware version
NX1001	1.2.0.2 or later
NX1005	1.2.0.3 or later
NX2001	1.2.0.2 or later
NX2020	1.2.0.2 or later
NX6000	1.2.0.2 or later
NX6010	1.0.0.0 or later
NX6020	1.0.0.0 or later
NX6100	1.2.0.1 or later

Table 13: Compatibility with Other Products

2.3.1.   Migration of Projects from HD8500 to MT8500

When migrating projects from HD8500 to MT8500, it may be necessary to make minor adjustments to the application code due to differences between the libraries of the two tools. If compilation errors occur, the user should consider the following adaptations:

- Replace the reference (namespace) to the *LibPlcStandard* library from HD8500 with *NextoStandard* from MT8500.
- Replace the enumerated data type *RTC_CMD_STATUS* in HD8500 with *RTC_STATUS* in MT8500.
- Replace the struct data type *TYPE_RESULT* in HD8500 with *ERRORCODE* in MT8500 in the function GetTaskInfo.

2.4.   Performance

The performance of the Hadron Xtorm Series CPU depends on a few factors, among them:

- User Application Time
- Application Interval
- Operating System Time
- Number of modules (process data, inputs/outputs, and others)

2.4.1.   Application Times

The execution time of a Hadron Xtorm Series CPU application depends on the following variables:

- Input reading time (local and remote)
- Task execution time
- Output writing time (local and remote)

It is important to note that the execution time of the *MainTask* will be directly influenced by the *Configuration* system task, a high-priority task that is periodically executed by the system. The *Configuration* task can interrupt the *MainTask*.

altus

2.4.2.   Time for Instructions Execution

In the following table are the times required to perform different instructions in the Hadron Xtorm Series CPU:

Instruction	Language	Variables	Execution Times (μs)
1000 Contacts	LD	BOOL	3
1000 Divisions	ST	INT	22
		REAL	41
	LD	INT	22
		REAL	41
1000 Multiplications	ST	INT	8
		REAL	12
	LD	INT	8
		REAL	12
1000 Sums	ST	INT	8
		REAL	12
	LD	INT	8
		REAL	12

Table 14: Instruction Times

2.4.3.   Initialization Times

Hadron Xtorm Series CPUs have initialization times of 25 seconds, and the initial screen with the HADRON XTORM logo (Splash) is presented after 7 seconds from the power on.

The initialization times of the I/O modules depend on several factors, such as, number of bus modules, "MainTask interval", etc.

2.5.   Related Products

The following products must be purchased separately when needed:

Code	Description
HD8500	MasterTool Xtorm
MT8500	MasterTool IEC XE
AL-2600	RS-485 network branch and terminator
AL-2306	RS-485 cable for MODBUS or CAN network
AL-1729	RJ45-CMDB9 Cable
AL-1748	CMDB9-CFDB9 Cable
AL-1752	CMDB9-CMDB9 Cable
AL-1753	CMDB9-CMDB25 Cable
AL-1754	CMDB9-CFDB9 Cable
AL-1762	CMDB9-CMDB9 Cable
AL-1763	CMDB9-Terminal Block Cable
NX9202	RJ45-RJ45 2 m Cable
NX9205	RJ45-RJ45 5 m Cable
NX9210	RJ45-RJ45 10 m Cable
NX9101	32 GB microSD memory card with miniSD and SD adapters
HX9401	06-terminal Connector

altus

Code	Description
HX9402	10-terminal Connector
HX9405	04-terminal Connector
HX9102	Backplane Connector Cover

Table 15: Related Products

Notes:

HD8500: MasterTool Xtorm is the programming tool used for the Hadron Xtorm Series.

MT8500: MasterTool IEC XE is available in four different versions: LITE, BASIC, PROFESSIONAL and ADVANCED. For more details, please check MasterTool IEC XE User Manual - MU299609.

AL-2600: This module is used for derivation and termination of RS-422/485 networks. For each network node there must be an AL-2600. The AL-2600 modules that are in the ends of the network must be configured as termination, except when there is a device with active internal termination. The other modules must be configured as derivation.

AL-2306: Two shielded twisted pairs cable without connectors, used for networks based on RS-485 or CAN.

AL-1729: RS-232C standard cable with one RJ45 connector and one DB9 male connector for communication between CPUs of Hadron Xtorm Series and the other Altus products (DUO, Piccolo and Ponto Series).

AL-1748: RS-232C standard cable with one DB9 male connector and one DB9 female connector for communication between CPUs of Hadron Xtorm Series and other products of the Altus Cimrex Series.

AL-1752: RS-232C standard cable with two DB9 male connectors for communication between CPUs of Hadron Xtorm Series and the Altus products of the H Series and HMIs of IX Series.

AL-1753: RS-232C standard cable with one DB9 male connector and one DB25 male connector for communication between CPUs of Hadron Xtorm Series and the Altus H Series products.

AL-1754: RS-232C standard cable with one DB9 male connector and one DB9 female connector for communication between CPUs of Hadron Xtorm Series and the Altus Exter Series products or a microcomputer serial port, RS-232C standard.

AL-1762: RS-232C standard cable with two DB9 male connectors for communication between CPUs of Hadron Xtorm Series and for communication between CPUs of Nexto Series.

AL-1763: Cable with one DB9 male connector and terminal blocks for communication between CPUs of Hadron Xtorm Series and the products with RS-232C/RS-485/RS-422 standard terminal blocks.

NX9202, NX9205 e NX9210: Ethernet CAT5 cable, shielded, twisted pair, with RJ45 male connectors in the ends, supports temperature of -5 °C to 70 °C, used for Ethernet networks with 2, 5 and 10 m of maximum length respectively.

HX9401, HX9402 e HX9405: 6, 10 and 4-terminal connectors, respectively. HX9401 (6-terminals) is exclusively used in HX8300 and HX8320 power supplies. HX9402 (10-terminals), on the other hand, can be used in any input/output modules of the Hadron Xtorm Series. HX9405 (4-terminals) refers to the exclusive use of the CPU HX3040 IRIG-B. These products support temperatures ranging from -5 °C to 70 °C.

HX9102: It is a cover to protect the Hadron Xtorm Series rack connectors. This cover was designed to provide a high protection to the unused rack connectors. It is recommended that all unused positions be protected with the HX9102 product, which supports temperatures of -5 °C to 70 °C.

3. Installation

This chapter introduces the necessary procedures for the physical installation of the Hadron Xtorm Series products, as well as the cautions that must be taken with other existing installations in the electrical cabinet occupied by the CPU.

3.1. Visual Inspection

Before proceeding with the installation, it is recommended to make a careful visual inspection of the equipment, verifying that there is no damage caused by transport on the equipment. Check that all components of your order are in perfect condition and any problem detected must be reported to the shipping company and to the nearest Altus representative or distributor.

> **CAUTION**
>
> Before removing the modules from the packaging, it is important to discharge any static potentials accumulated in the body. To do this, touch (with bare hands) any grounded metal surface before handling the modules. This procedure ensures that the static electricity levels supported by the module will not be exceeded.

It is important to register the serial number of each piece of equipment you receive, as well as the software revisions, if applicable. This information will be needed if you need to contact Altus Support.

3.2. Mechanical Installation

3.2.1. Rack Clamping

3.2.1.1. Drilling for 9 Slots Rack

The 9-position rack must be fixed using five M4 screws as shown in the following figure.
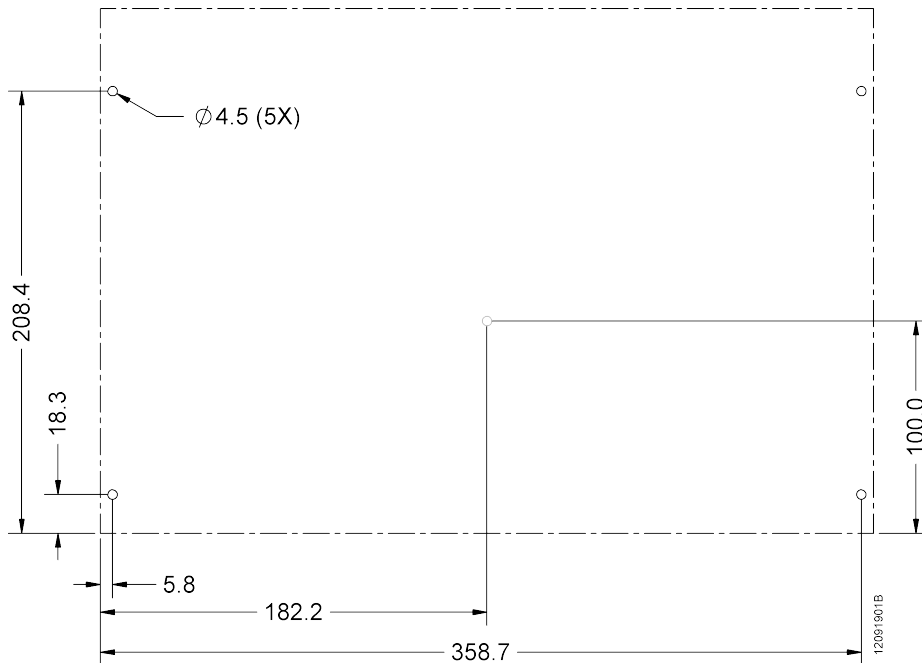
Figure 17: Drilling for 9 Slots Rack

3.2.1.2. Drilling for 18 Slots Rack

The 18-position rack must be fixed using six M4 screws as shown in the following figure.
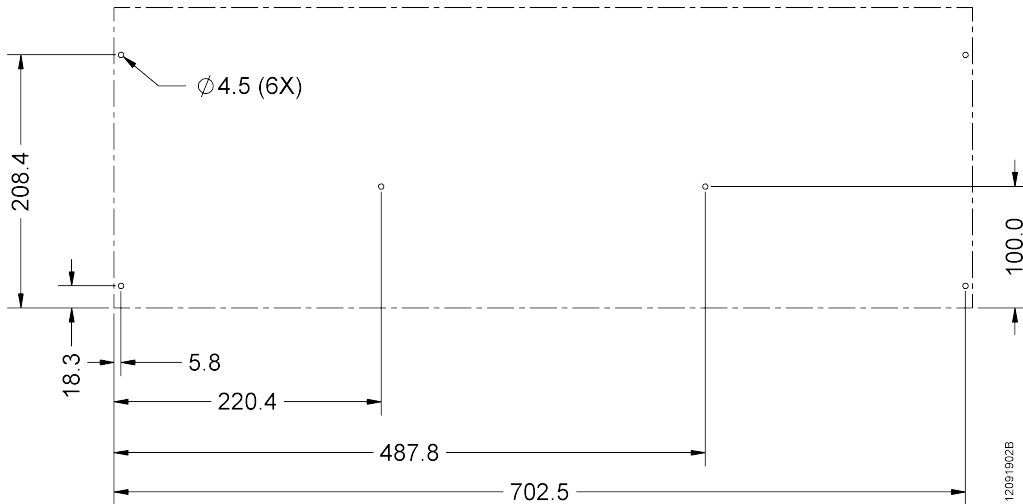
Figure 18: Drilling for 18 Slots Rack

DIN 7985 M4 cross head screws must be used in all holes. These screws can be fixed directly to the panel or in nuts, when the thickness of the panel is insufficient to have thread. When using nuts, we recommend using a self-locking type nut in order to avoid loosening.

> **ATTENTION**
>
> Section Spacing between modules and other equipment in the panel should be consulted to verify the minimum distances that must exist between the Hadron Xtorm Series rack and the other components located in the electrical panel.

3.2.1.3. Assembly

Before the rack is inserted into the panel, the screws in the hole type 1, as shown in figure 17, must be partially inserted.

Align the rack to the two type 1 screws and place it against the bottom of the electrical panel. Figure 19 indicates how this procedure should be done.

Note: Some figures used in this item do not show the rack Printed Circuit Board for reasons of simplifying the understanding of the procedure.
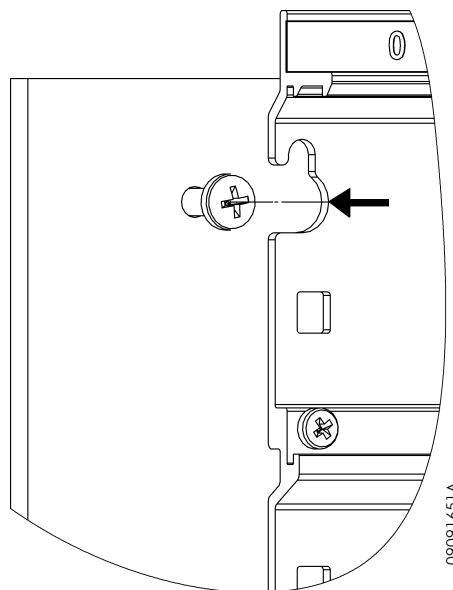
Figure 19: Rack Clamping – Alignment

Then, a movement must be made in such a way that the screws of the type 1 holes fit into the smallest part of the rack opening, as shown in the following figure.
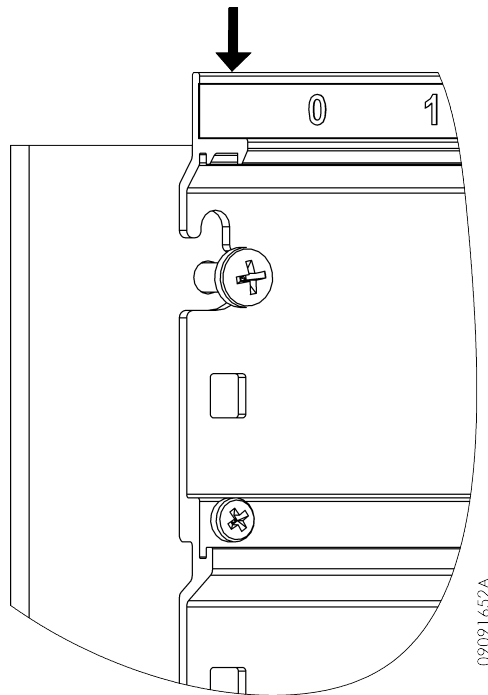
Figure 20: Rack Clamping – Fitting in the Screws

After the rack has been fully inserted, all the screws used to secure the rack must be assembled.

3.2.1.4. Removal

To remove the rack, you must perform the reverse sequence indicated in section Assembly.

3.2.2. Module Insertion

The following example shows a generic Hadron Xtorm Series module so that the procedure should be followed for all modules of the Series.

First the bottom part of the module must be fitted, which serves as a guide for the correct insertion into the rack. When fitting the bottom part of the module, it must be checked that the guide pins are correctly fitted into the rack slots corresponding to a certain position. The figure below shows how the bottom part of the module must be positioned in relation to the rack for correct insertion. Only one module should be inserted at a time.
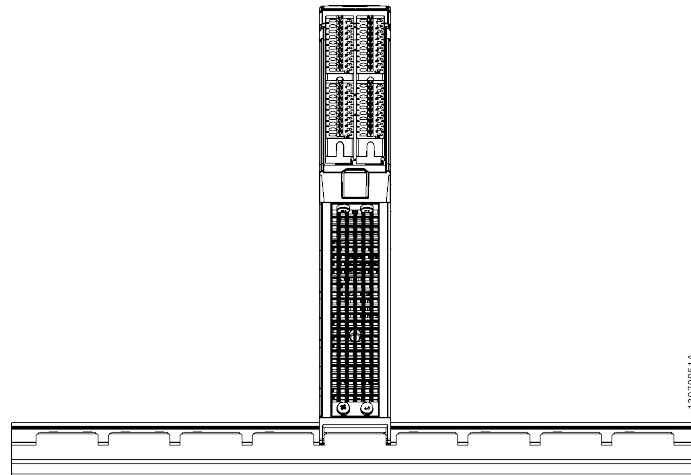
Figure 21: CPU connected on the Rack

After fitting the bottom of the module as described above, a single, continuous rotation movement must be made through the clamping cursor that must be in the locked position, so that the bottom of the module stays in place and the clamping lock fits into the top of the rack. After this procedure has been completed, you must wait at least two seconds before performing a new insertion/removal procedure. The following figure shows the movement that must be performed.
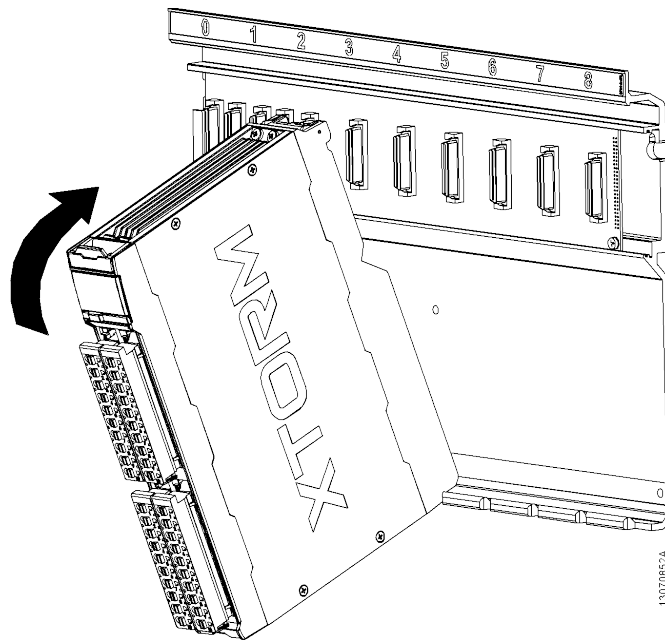
Figure 22: Movement for Rack Clamping

If the user follows the procedures described above correctly, the module is perfectly connected to the rack, as shown in the figure below.
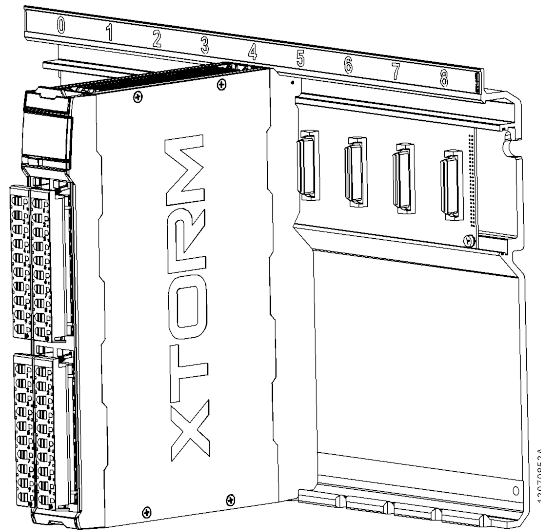
Figure 23: Docked Module in the Rack

The module cannot be connected into the rack in any other way. Attempting to insert the module the wrong way may cause irreparable damage to the module. The figure below shows one way NOT to insert Hadron Xtorm Series modules into the rack.
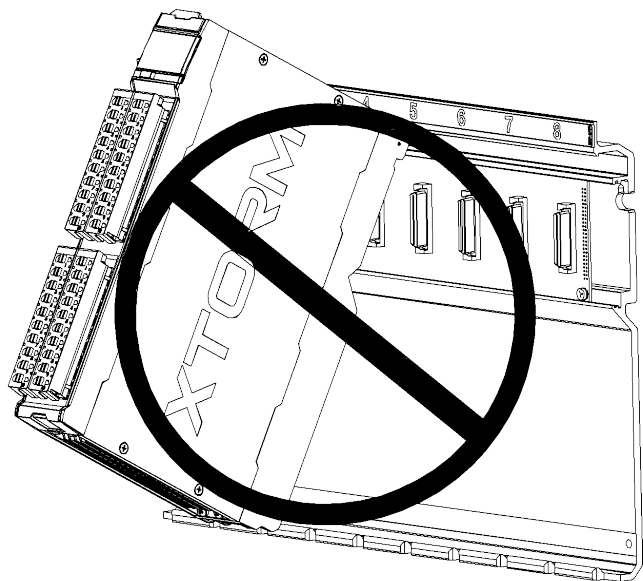
Figure 24: Incorrect Insertion Form

3.2.3.   Module Removal

The following example shows a generic Hadron Xtorm Series module so that the procedure should be followed for all modules of the Series.

First, the clamping cursor must be pulled, as indicated by number 1 in the figure below, in order to unlock it from the rack, and then rotated in a single continuous movement in the direction indicated by number 2, also in the figure below.
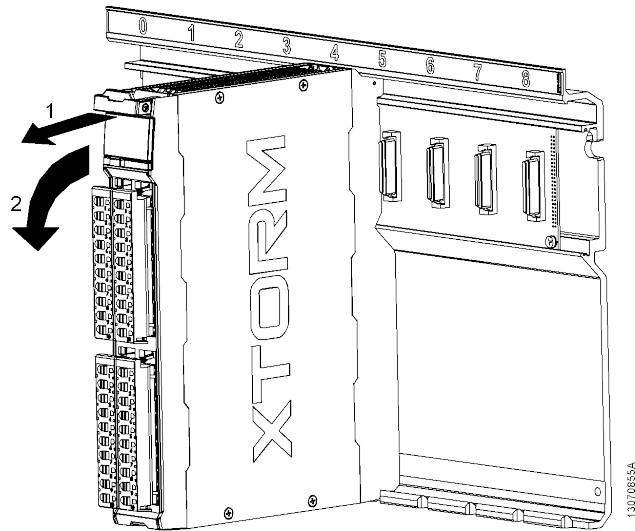
Figure 25: Bus Disconnection

If the user correctly follows the procedures described above, the module will be disconnected from the backplane and can be removed as shown in the figure below. Only one module should be removed at a time. After this procedure is completed, you must wait at least two seconds for a new insertion/removal procedure to be executed.
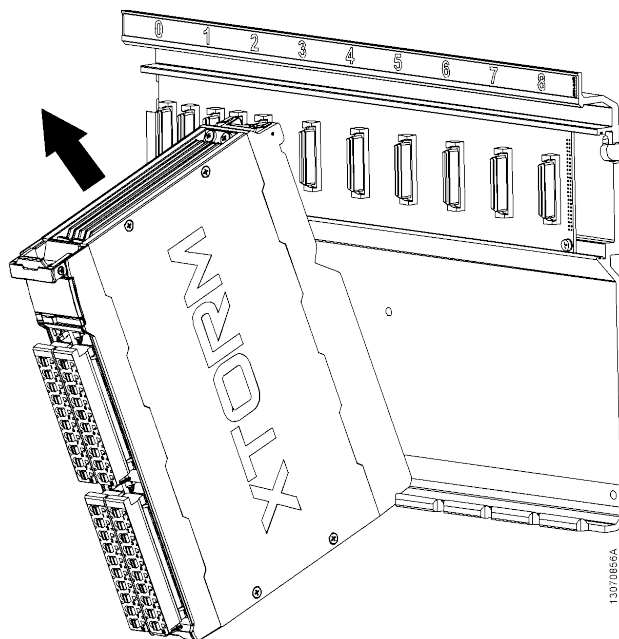
Figure 26: Removal of the Module

3.2.3.1. I/O Modules

The input and output modules of the Hadron Xtorm Series present some specific characteristics that will be addressed below.

3.2.3.2. I/O Terminal Blocks

The terminals blocks of Hadron Xtorm Series I/O modules use a clamping wire system with spring terminals, which do not require a screw for this purpose.

3.2.3.3. Identification

All pins of I/O terminals blocks have a serial number from 1 to 10. The relationship between the pin number and its respective functionality for a specific module is defined in the Technical Characteristics document of the respective module.

3.2.3.4. Installation Diagram

The installation diagram of each module is defined in its Technical Characteristics document.

3.2.3.5. Rack Connector Cover

The rack connector cover must be used in the rack positions that are not in use, so the rack gets protected against improper contacts and impurities.

3.2.3.6. Rack Connector Cover Insertion

Fit the Hadron Series Xtorm HX9102 connector cover from the bottom to the top, pressing it until it is completely settled, as shown in the figure below.

Figure 28 shows the connector cover fully settled.
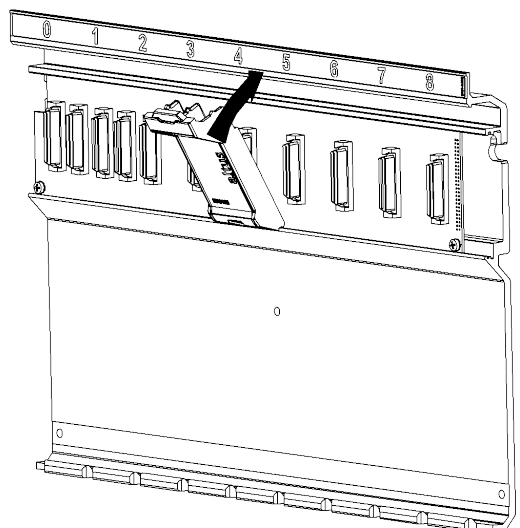
Figure 27: Rack Connector Cover Insertion

3.2.3.7. Rack Connector Cover Removal

In order to remove the Hadron Xtorm Series HX9102 connector cover, pull it from the upper side, as shown in the figure below.
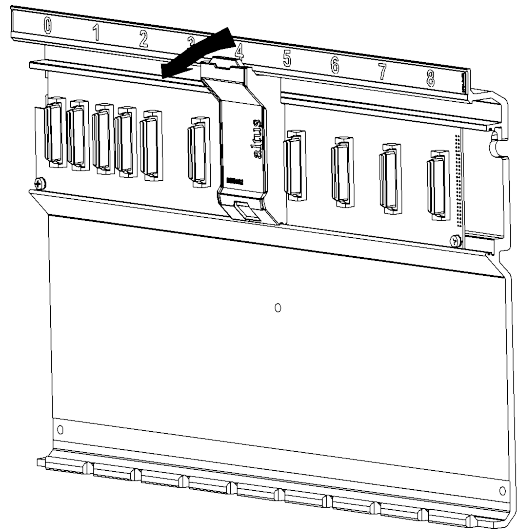
Figure 28: Rack Connector Cover Removal

3.3. Electrical Installation

> **DANGER**
>
> When executing any installation in an electric panel, certify that the main energy supply is OFF.

3.3.1. Electrical Safety

> **ATTENTION**
>
> The pin 5 of HX8300 and HX8320 modules terminal block is the ground protection and shall be connected to the local ground with a good connection, ensuring a maximum impedance of 0,1 Ω.

> **DANGER**
>
> The HX8320, HX1120 and HX2320 modules use voltages between 91 and 135 Vdc on power interfaces, digital inputs and outputs, respectively. Therefore, one should be careful when handling the cables and terminals of these modules during the stages of installation, use and maintenance, always checking if they are in good condition, without cuts or breaks. To ensure greater security, the modules terminal blocks are rated as IP20.

> **ATTENTION**
>
> The COM 1 serial interface (RS-232 / RS-485) of CPU HX3040 has a DB9 connector rated as IP20, and thus there is no risk of electric shock. However, be sure to not touch the cable pins, which are connected to this interface. Due to the danger presented by this interface in case of single failure, connect it exclusively to products with insulation higher than 820 Vac.

ATTENTION

The Printed Circuit Boards of the HX9001 and HX9003 racks, which may, under single fault conditions, present dangerous voltages, are represented in the regions highlighted in green in the figure below. Therefore, these regions should only be touched with the power supply turned off.
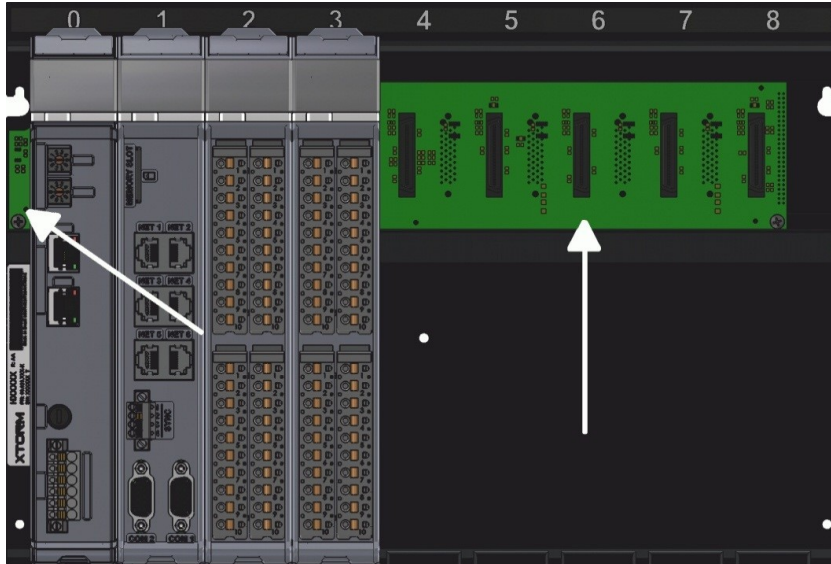
Figure 29: Printed Circuit Boards Area (HX9001 and HX9003)

3.3.2. Spring Terminal Blocks

This type of terminal block has a spring-based clamping system for high reliability, even in environments subject to vibration (Figures 30 and 31). For its assembly, it is recommended to use a 3.5 mm wide screwdriver and insulated cable (Figure 31). Through this system, the electrical cables are easily and quickly assembled.
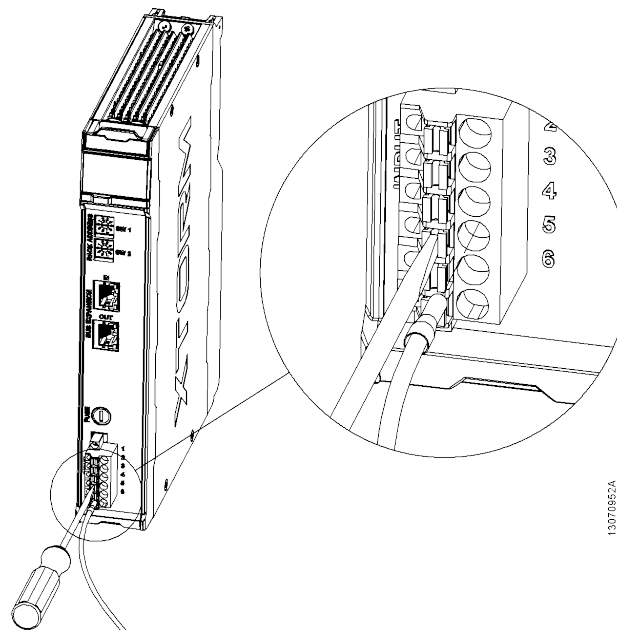
Figure 30: Source Module Spring Terminal Block

---

altus
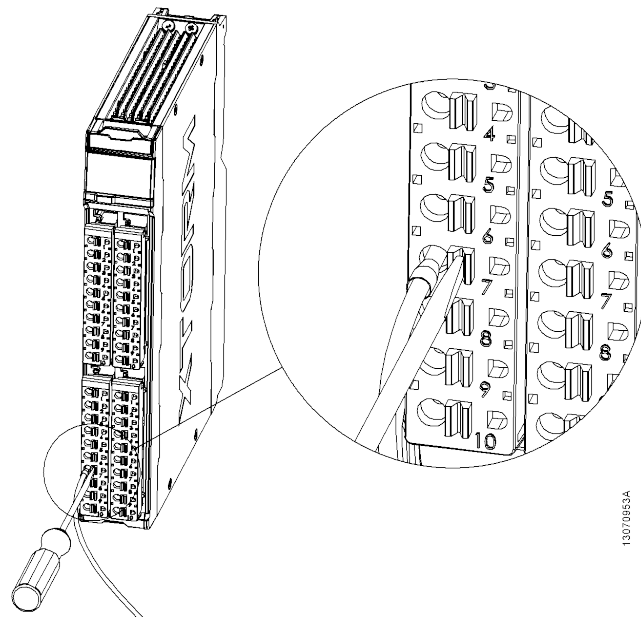
Figure 31: I/O Module Spring Terminal Block
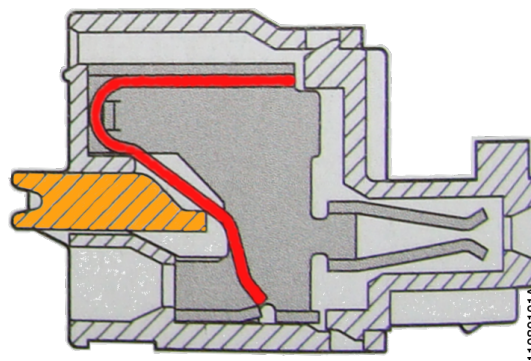
Figure 32: Spring Terminal Block

In order to mount the wire into the terminal block:

- Insert the screwdriver into the terminal block driver and open the terminal spring
- Insert the wire into the terminal block
- Remove the screwdriver and close the terminal block

3.3.2.1. Wiring Insertion

In order to get a similar length of all the wires connected to the same I/O terminal block, it is recommended to follow the following definition concerning the difference in length that each wire subsequent to the other must have before installation. It is important to emphasize that the longest wire should always be the one connected to pin 1 of the I/O terminal block.

3.3.2.2. 4-Route Pin

In this case, use a 0.5 mm² wire. Crimp the specified terminals to the given wire (0.5 mm²) in each route.

> **ATTENTION**
>
> Use terminals with length A = 8 mm to ensure effective contact (see figure 34).

3.3.2.3. 6-Route Pin

In this case, use a 2,5 mm² wire. Crimp the specified terminals to the given wire (2,5 mm²) in each route.

> **ATTENTION**
>
> Use terminals with length A = 12 mm to ensure effective contact (see figure 34).

3.3.2.4. 10-Route Pin

It is recommended to use 0.5 mm² to 1.5 mm² wire. Cut each wire with an 8 mm gap, as indicated in figure 33. Crimp the specified terminals to the given wire (0.5 mm² to 1.5 mm²) in each route.

> **ATTENTION**
>
> Use terminals with length A = 12 mm to ensure effective contact (see figure 34).
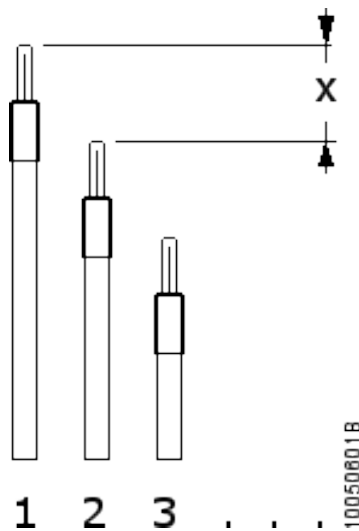
Figure 33: Wire Cutting
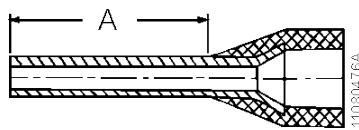
Figure 34: Terminal

3.3.2.5. Wiring Assembly

Place the terminals into the terminal block from pin number 10, to the 10-route terminal. Make sure that the terminals are fully inserted and properly connected into the terminal.

3.3.2.6. Wiring Cramping

Rotate the cables so that they are properly fit and settle them with a plastic wrapping as shown in the following figure.

altus
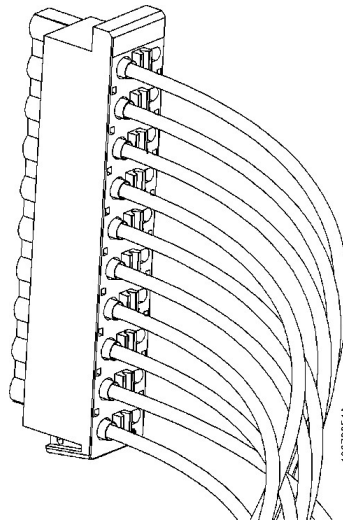
Figure 35: Cable Assembly (10-Route Connector)

3.3.2.7. Wiring Removal

To remove the wires from the terminal block it is recommended to use a 3.5 mm wide screwdriver with insulated cable. Insert the screwdriver into the drive next to the wire while pulling the wire out (figure 30 and figure 31).

3.3.3. Connections

The correct connection of the cables of the CPUs and the system modules ensures the safety of the equipment and its correct operation. For this, the following points should be checked:

- The cables near the connection terminals of the mounting panel must be connected securely and tightly.
- The power and grounding terminals of the system parts must be tight and well connected, ensuring good current flow.
- The connection of the equipment ground to the mounting panel ground must be tight and with the correct cable gauge to ensure good grounding and noise immunity.
- The connection between the connectors and the modules must be checked to ensure that it has been inserted completely, ensuring a correct connection.

3.3.4. Supply Voltages

Check that the supply voltages are within the values specified in the technical characteristics.

> **ATTENTION**
>
> Wherever there is high voltage, place a warning tag and protections in order to avoid the access.

3.3.5. Fuses

Check the system fuses, making sure that they are in good condition and with the proper value/type before powering up the system.

> **DANGER**
>
> Never replace a fuse for another with a higher current value. This procedure may cause serious damage to the equipment.

3.4. CPU Electrical Installation

> **DANGER**
>
> When executing any installation in an electric panel, certify that the main energy supply is OFF.

The Hadron Xtorm Series CPU power supply comes from the Power Supply Module, which provides voltage to the CPUs through the rack connection, with no need for external connections. The module grounding is accomplished through the contact between the grounding spring module and the rack.

The figure below shows the Hadron Xtorm Series CPU wiring diagram installed in a Hadron Xtorm Series rack. The same image can be seen on the left side of the CPU mechanics.

The arrangement of connectors and terminals in the figure is merely illustrative.
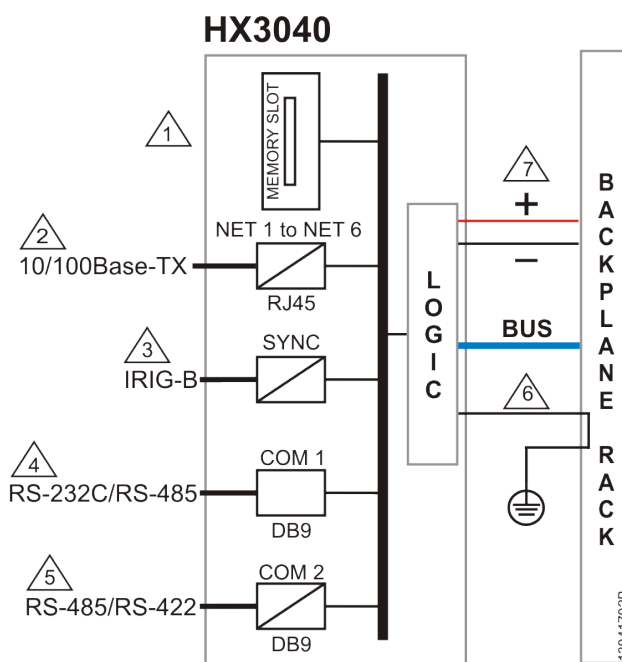
Figure 36: Hadron Xtorm Series CPU Electrical Diagram

Diagram Notes:

1   Interface for memory card.

2   Ethernet interface 10/100Base-TX standard for programming, debugging and connection to the supported protocols.

3   IRIG-B interface standard for connection to synchronism time signal. It is recommended to use a clabes section of $1,5mm^2$.

4   Serial interface RS-232C/RS-485 standard for connection with MODBUS RTU network or other protocols.

5   Serial interface RS-485C/RS-422 standard for connection with MODBUS RTU network or other protocols. The choice of the type of physical interface depends on the cable used.

6   The module is grounded through the Hadron Xtorm Series racks.

7   The module's power supply comes from the connection to the rack, without the need for external connections.

altus

3.5. Ethernet Network Connection

The isolated NET 1 to NET 6 communication interfaces make it possible to connect to an Ethernet network, where the NET 1 interface (or NET 2 when configured in a Teaming NIC pair with NET 1) must be used when communicating with MasterTool Xtorm.

The connection to the Ethernet network uses twisted-pair type cables (10/100Base-TX), and the speed detection is automatically performed by the Hadron Xtorm CPU. This cable must have one of its ends connected to the intended interface and the other to the HUB, switch, microcomputer or other Ethernet network point.

3.5.1. IP Address

The NET 1 Ethernet interface is used for Ethernet communication and CPU configuration. Default parameters:

	NET 1
IP Address	192.168.15.1
Subnet Mask	255.255.255.0
Gateway Address	192.168.15.253

Table 16: NET 1 Ethernet Interface Default Parameters

The parameters IP Address and Subnet Mask can be viewed on the CPU's graphical display through the parameters menu, as described in section Informative Menu and of CPU's Configuration.

Initially, the NET 1 interface of the CPU must be connected to a network or microcomputer with the same subnet to communicate with the MasterTool Xtorm, where the network parameters can be changed. For more details on configuring and changing network parameters, please check section Ethernet Interfaces Configuration.

The NET 2, NET 3, NET 4, NET 5, and NET 6 Ethernet interfaces are used for Ethernet communication. Default parameters:

	NET 2
IP Address	192.168.16.1
Subnet Mask	255.255.255.0
Gateway Address	192.168.16.253

Table 17: NET 2 Ethernet Interface Default Parameters

	NET 3
IP Address	192.168.17.1
Subnet Mask	255.255.255.0
Gateway Address	192.168.17.253

Table 18: NET 3 Ethernet Interface Default Parameters

	NET 4
IP Address	192.168.18.1
Subnet Mask	255.255.255.0
Gateway Address	192.168.18.253

Table 19: NET 4 Ethernet Interface Default Parameters

altus

	NET 5
IP Address	192.168.19.1
Subnet Mask	255.255.255.0
Gateway Address	192.168.19.253

Table 20: NET 5 Ethernet Interface Default Parameters

	NET 6
IP Address	192.168.20.1
Subnet Mask	255.255.255.0
Gateway Address	192.168.20.253

Table 21: NET 6 Ethernet Interface Default Parameters

The network parameters of NET 1 .. NET 6 interfaces can be changed through MasterTool Xtorm. For further information on configuration and network parameters changes, see Ethernet Interfaces Configuration.

3.5.2. Free ARP

The NETx Ethernet interface promptly sends ARP packets type in broadcast informing its IP and MAC address for all devices connected to the network. These packets are sent during a new application download by the MasterTool Xtorm software and in the CPU startup when the application goes into Run mode.

Five ARP commands are triggered within a 200 ms initial interval, doubling the interval every new triggered command, totalizing 3 s. Example: the first trigger occurs at time 0, the second one at 200 ms and the third one at 600 ms and so on until the fifth trigger at time 3 s.

3.5.3. Network Cable Installation

Hadron Xtorm Hadron Series CPU Ethernet ports, identified on the panel by NET 1 ... NET 6, have standard pinouts, the same used, for example, in personal computers. Connector type, cable type, physical level, and other details for connecting the CPU to the Ethernet network access device are defined in section Ethernet Interfaces.

The table and figure below show the RJ45 female connector of the Hadron Xtorm CPU, with the identification and pinout description valid for the 10Base T and 100Base TX type physical levels.
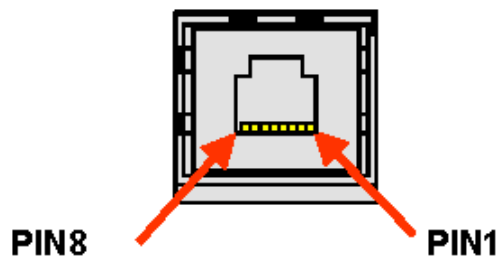
Figure 37: RJ45 Hadron Xtorm CPU Female Connector

Pin	Signal	Description
1	TXD +	Data transmission, positive
2	TXD -	Data transmission, negative
3	RXD +	Data reception, positive
4	NU	Not used
5	NU	Not used
6	RXD -	Data reception, negative
7	NU	Not used
8	NU	Not used

Table 22: RJ45 Hadron Xtorm Female Connector Pin Out

The interface can be connected to a communication network through a hub or switch, or directly to the communication equipment. In this last case, due to Hadron Xtorm CPU has the Auto Crossover feature, it is not necessary to use a network cable called Cross-over, the same used to connect two personal computers, point to point, through the Ethernet port.

It is important to emphasize that a network cable is understood as a pair of RJ45 male connectors interconnected to each other by a category 5 UTP or ScTP cable, under the direct or Cross-over configuration. The same is used to interconnect two devices with an Ethernet port.

Usually, these cables have a connection lock that ensures a perfect connection between the female connector of the interface and the male connector of the cable. At the time of installation, the cable's male connector must be inserted into the module's female connector until a characteristic sound ("click") is heard, ensuring the lock action. The lever present in the male connector must be used to disconnect them.

3.6.   Serial Network Connection (COM 1)

> **ATTENTION**
>
> For a correct and safe installation of the COM 1 Serial Network, please first check section Electrical Safety, which is part of this same chapter.

The non-isolated COM 1 communication interface enables connection to an RS-232C or RS-485 network. In the following sections, the Hadron Xtorm CPU DB9 female connector is presented with its identification and signals description.
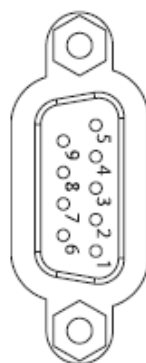
Figure 38: Hadron Xtorm CPU DB9 Female Connector

Pin	Signal	Interface	Description
1	DCD	RS-232C	Data Carrier Detect
2	TXT	RS-232C	Data transmission
3	RXD	RS-232C	Data reception
4	D-	RS-485	Data transmission/ reception, negative
5	GND	RS-232C RS-485	RS-232C: Ground RS-485: Negative reference for external termination
6	D+	RS-485	Data transmission/ reception, positive
7	CTS	RS-232C	Clear to Send
8	RTS	RS-232C	Request to Send
9	+5V	RS-485	Positive reference for external termination

Table 23: Hadron Xtorm CPU DB9 Female Connector Pinout

3.6.1. RS-232C Communication

For connection to a RS-232C device, use the appropriate cable as described in the section Compatibility with Other Products.

3.6.2. RS-485 Communication without Termination

For connection to an RS-485 network without termination on the COM 1 interface, you must connect the identified terminals of the AL-1763 cable to the respective terminals of the device, according to the table below.

AL-1763 Pins	CPU Signals
0	Shield
1	Not connected
2	Not connected
3	D-
4	Not connected
5	D+
6	Not connected
7	Not connected
8	Not connected

Table 24: Connections of COM 1 with RS-485 without Termination

The diagram in the figure below indicates how to connect the AL-1763 terminals to the RS-485 network.

altus
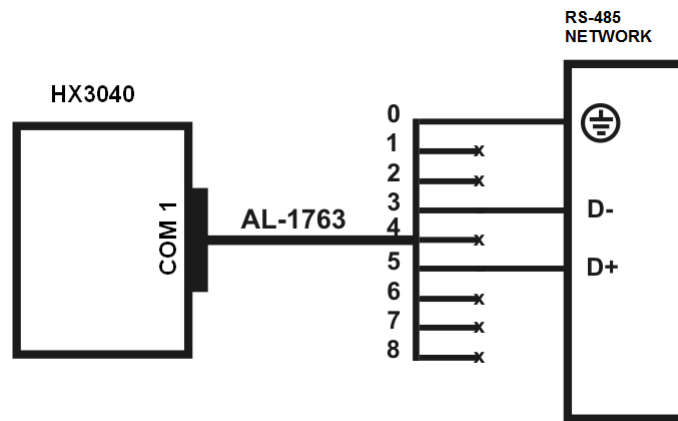
Figure 39: Connection Diagram of the COM 1 with RS-485 without Termination

Diagram Note:

The unconnected terminals must be insulated so that there is no contact between them.

3.6.3. RS-485 Communication with Internal Termination

The COM 1 interface does not allow internal termination. For connection to an RS-485 network using internal termination, the COM 2 interface must be used.

3.6.4. RS-485 Communication with External Termination

For connection to an RS-485 network, using the external termination of the COM 1 interface, the identified terminals of the AL-1763 cable must be connected to the respective terminals of the device, according to the table below.

AL-1763 Pins	CPU Signals
0	Shield
1	Not connected
2	Not connected
3	D-
4	0 V
5	D+
6	Not connected
7	Not connected
8	+ 5 V

Table 25: Connection of the COM 1 with RS-485 with External Termination

The diagram in the figure below indicates how to connect the AL-1763 terminals to the RS-485 network.
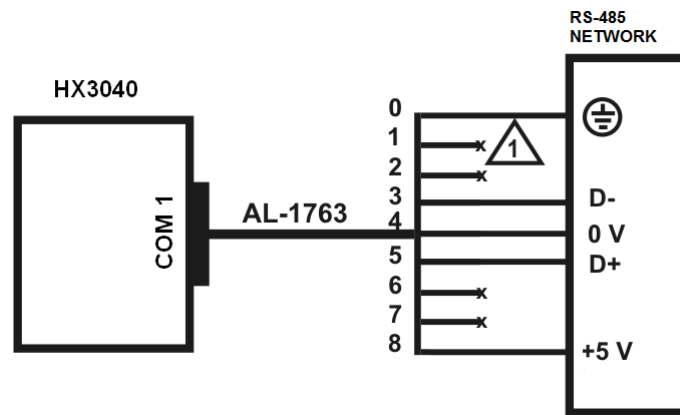
Figure 40: Connection Diagram of the COM 1 with RS-485 with External Termination

Diagram Note:

The unconnected terminals must be insulated so that there is no contact between them.

3.7. Serial Network Connection (COM 2)

The isolated COM 2 communication interface enables connection to an RS-485/422 network. The following shows the female DB9 connector of the Hadron Xtorm CPU, with signal identification and description.
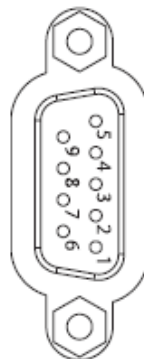
Figure 41: Hadron Xtorm CPU DB9 Female Connector

Pin	Signal	Description
1	-	Not used
2	Term+	Internal termination, positive
3	TXD+	Data transmission, positive
4	RXD+	Data reception, positive
5	GND	Negative reference to external termination
6	+5V	Positive reference to external termination
7	Term-	Internal termination, negative
8	TXD-	Data transmission, negative
9	RXD-	Data reception, negative

Table 26: Hadron Xtorm CPU DB9 Female Connector Pin Out

3.7.1. RS-485 Communication without Termination

For connection to an RS-485 network without termination on the COM 2 interface, the identified terminals of the AL-1763 cable must be connected to the respective terminals of the device, according to the table below.

AL-1763 Pins	CPU Signals
0	Shield
1	Not connected
2	D+
3	D+
4	Not connected
5	Not connected
6	Not connected
7	D-
8	D-

Table 27: Connection between COM 2 with RS-485 without Termination

The diagram in the figure below indicates how the AL-1763 terminals should be connected to the device's terminal block.
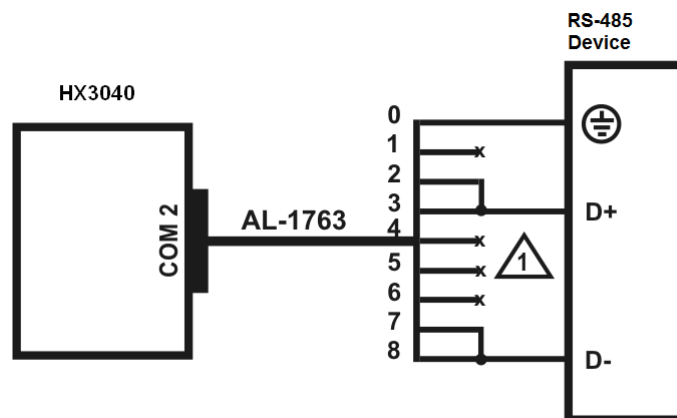
Figure 42: Connection Diagram of the COM 2 with RS-485 without Termination

Diagram Note:

The unconnected terminals must be insulated so that there is no contact between them.

3.7.2. RS-485 Communication with Internal Termination

For connection to an RS-485 network, using the internal termination of the COM 2 interface, the identified terminals of the AL-1763 cable must be connected to the respective terminals of the device, according to the table below.

AL-1763 Pins	CPU Signals
0	Shield
1	D+
2	D+
3	D+
4	Not connected
5	Not connected
6	D-
7	D-
8	D-

Table 28: Connection of the COM 2 with RS-485 with Internal Termination

Note: The internal termination available on COM 2 is of the safe state open mode type.

The diagram in the figure below indicates how the AL-1763 terminals should be connected to the device's terminal block.
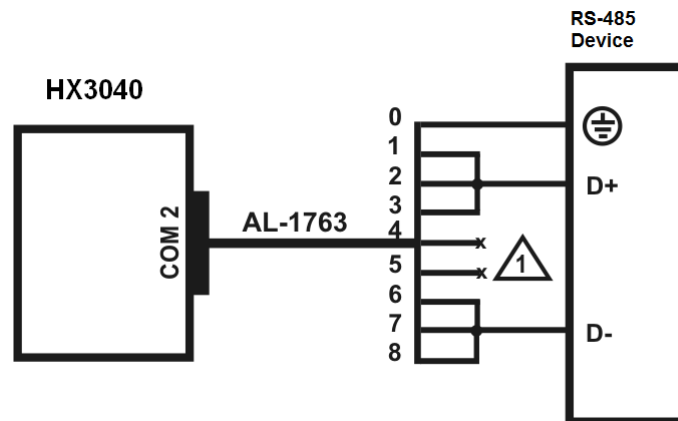
Figure 43: Connection Diagram of the COM 2 with RS-485 with Internal Termination

Diagram Note:

The unconnected terminals must be insulated so that there is no contact between them.

3.7.3. RS-485 Communication with External Termination

For connection to an RS-485 network, using the external termination of the COM 2 interface, the identified terminals of the AL-1763 cable must be connected to the respective terminals of the device, according to the table below.

AL-1763 Pins	CPU Signals
0	Shield
1	Not connected
2	D+
3	D+
4	0 V
5	+5 V
6	Not connected
7	D-
8	D-

Table 29: Connection of the COM 2 with RS-485 with External Termination

The diagram in the figure below indicates how the AL-1763 terminals should be connected to the device's terminal block.
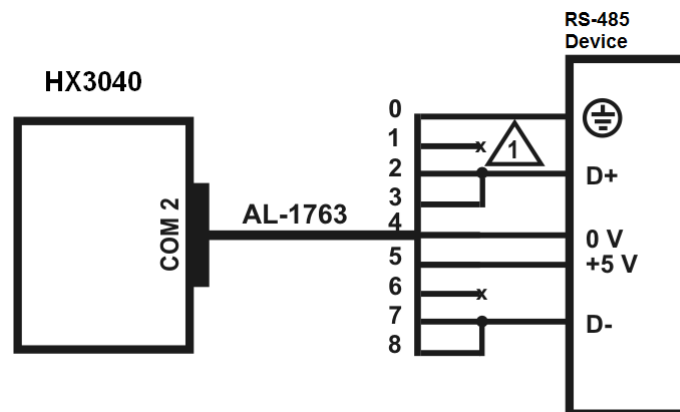
Figure 44: Connection Diagram of the COM 2 with RS-485 with External Termination

Diagram Note:

The unconnected terminals must be insulated so that there is no contact between them.

3.7.4. RS-422 Communication without Termination

For connection to an RS-422 network without termination on the COM 2 interface, the identified terminals of the AL-1763 cable must be connected to the respective terminals of the device, as shown in the table below.

AL-1763 Pins	CPU Signals
0	Shield
1	Not connected
2	TX+
3	RX+
4	Not connected
5	Not connected
6	Not connected
7	TX-
8	RX-

Table 30: Connection of the COM 2 with RS-422 without Termination

The diagram in the figure below indicates how the AL-1763 terminals should be connected to the device's terminal block.
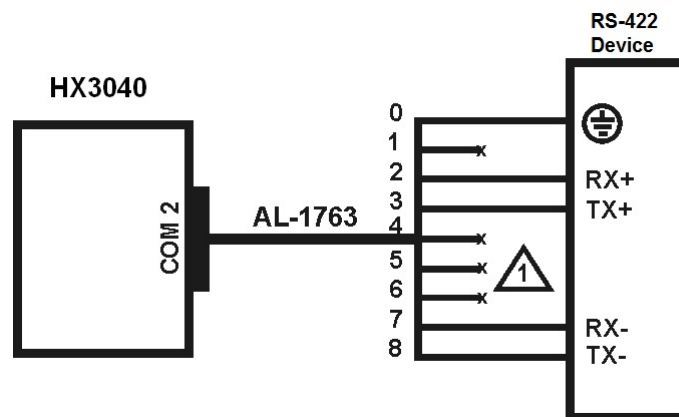
Figure 45: Connection Diagram of the COM 2 with RS-422 without Termination

Diagram Note:

The unconnected terminals must be insulated so that there is no contact between them.

3.7.5. RS-422 Communication with Internal Termination

For connection to an RS-422 network, using the internal termination of the COM 2 interface, the identified terminals of the AL-1763 cable must be connected to the respective terminals of the device, according to the table below.

AL-1763 Pins	CPU Signals
0	Shield
1	TERM+
2	TX+
3	RX+
4	Not connected
5	Not connected
6	TERM-
7	TX-
8	RX-

Table 31: Connection of the COM 2 with RS-422 with Internal Termination

Note: The internal termination available on COM 2 is of the safe state open mode type.

The diagram in the figure below indicates how the AL-1763 terminals should be connected to the device's terminal block.
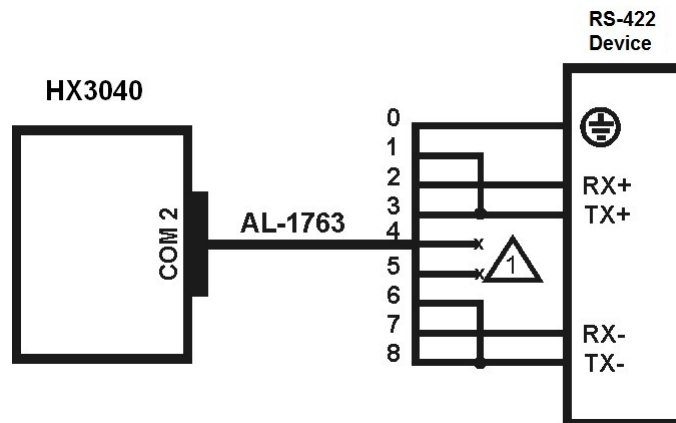
Figure 46: Connection Diagram of the COM 2 with RS-422 with Internal Termination

Diagram Note:

The unconnected terminals must be insulated so that there is no contact between them.

3.7.6. RS-422 Communication with External Termination

For connection to an RS-422 network, using the external termination of the COM 2 interface, the identified terminals of the AL-1763 cable must be connected to the respective terminals of the device, according to the table below.

AL-1763 Pins	CPU Signals
0	Shield
1	Not connected
2	TX+
3	RX+
4	0 V
5	+5 V
6	Not connected
7	TX-
8	RX-

Table 32: Connection of the COM 2 with RS-422 with External Termination

The diagram in the figure below indicates how the AL-1763 terminals should be connected to the device's terminal block.
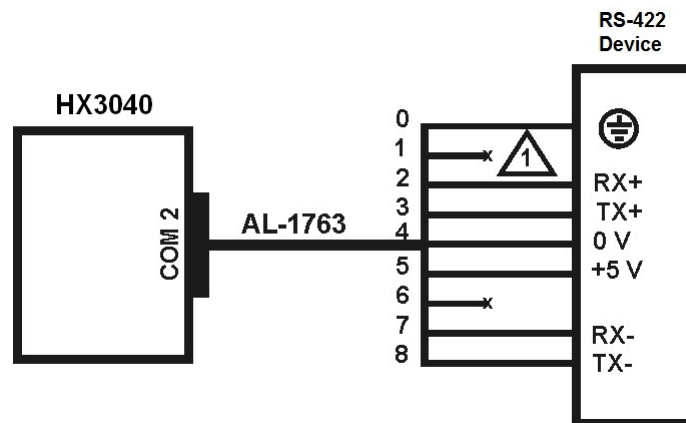
Figure 47: Connection Diagram of the COM 2 with RS-422 with External Termination

Diagram Note:

The unconnected terminals must be insulated so that there is no contact between them.

3.7.7. RS-422 Network Example

The figure below shows an example of using the RS-422 network, using the Xtorm CPU as master, slave devices with RS-422 interface, and Altus solution for derivators and terminators.
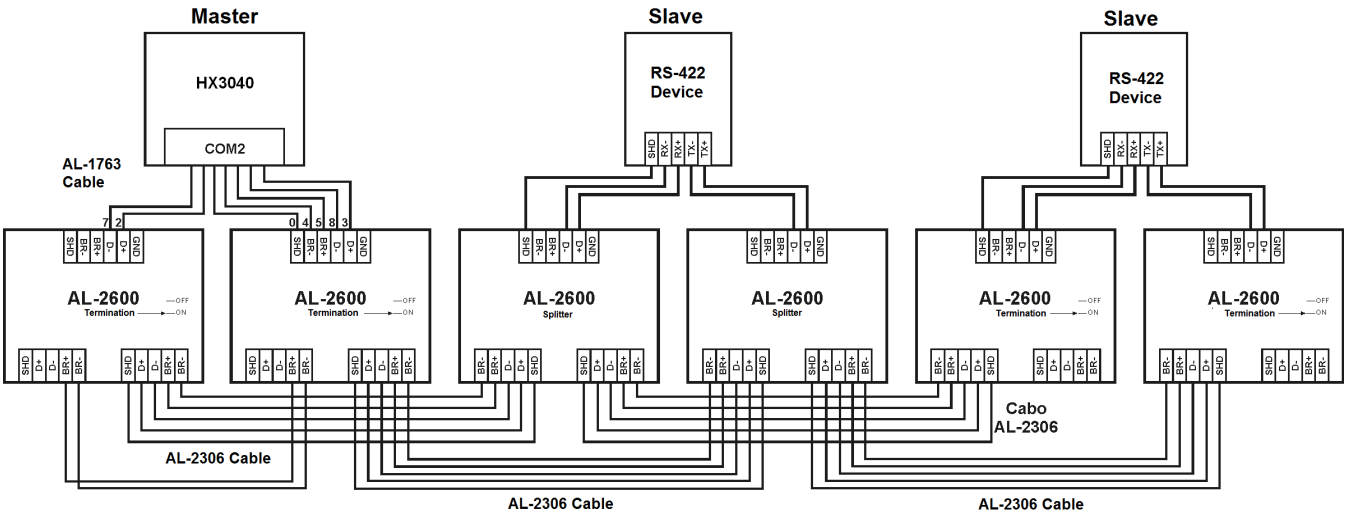
Figure 48: RS-422 Network Example

Diagram Note:

The AL-2600 modules at the ends of the network act as terminators. In this case AL-2600 switches must be configured in PROFIBUS Termination.

Note:

All termination methods are polarized for RS-485 and RS-422 communication - there are no passive terminations.

3.8. IRIG- B Connection

For applications where time synchronization with other equipment is required, the Hadron Xtorm Series CPU has one input port and one output port for the IRIG-B signal. Through the input port, the CPU will receive the time synchronization data and synchronize with its internal clock. And through the output port, the Xtorm CPU will be able to synchronize other equipment, retransmitting at the output port, the signal received at the input port. For cases in which the Hadron Xtorm Series bus expansion is used, the CPU internal clock synchronism will be in charge of synchronizing the time stamps for the generation of events of the I/O modules present in the bus expansion, as shown in the figure below.
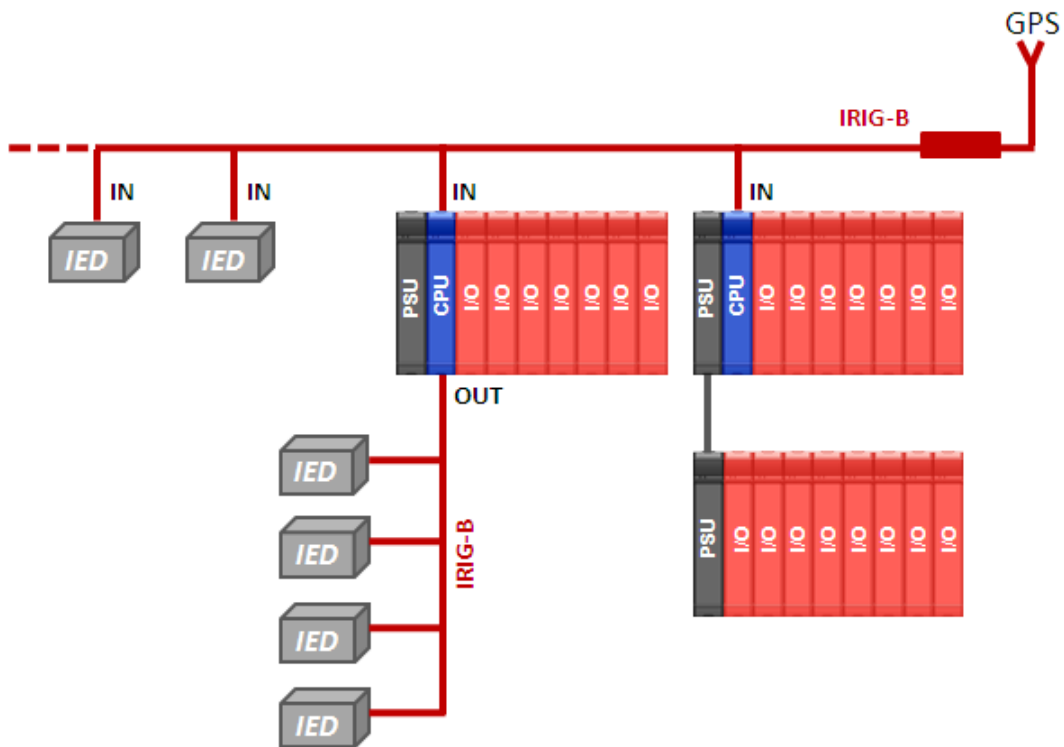
Figure 49: Example of Sync for Expansion Systems

For applications where CPU redundancy is used in the same rack, the time synchronization signal generated by GPS must be connected to the IRIG-B input of each of the CPUs, regardless of which CPU is in active mode and which is in standby mode, ensuring system time synchronism. In case it is necessary for the CPU to perform the function of repeater of the IRIG-B signal to other IEDs, only the output of one CPU must be used. In the figure below can be seen an example of this type of architecture.
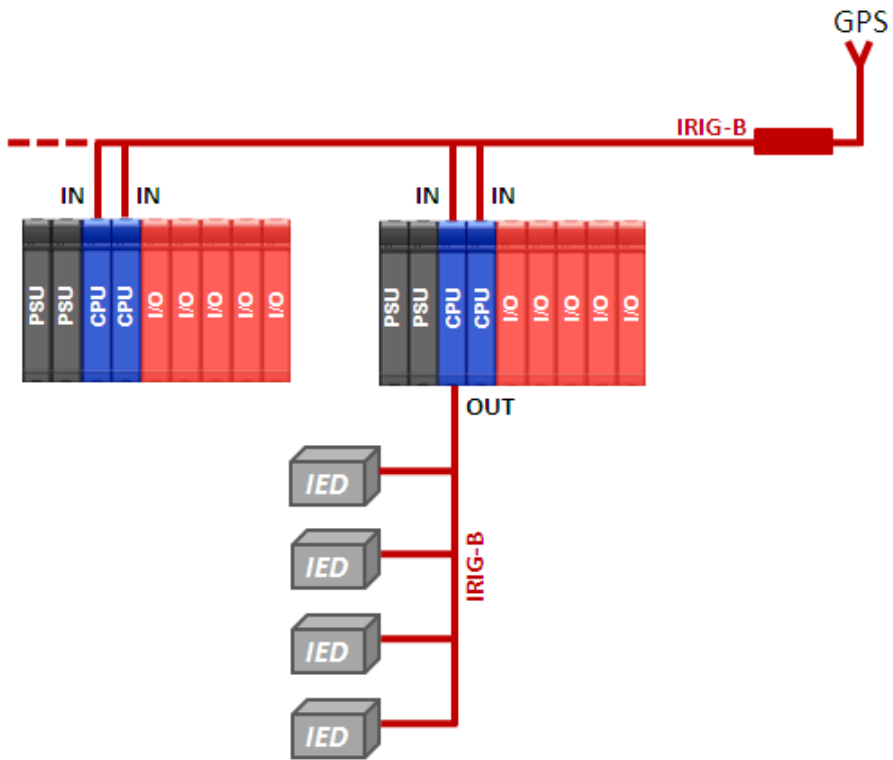
Figure 50: Example of Sync for Redundant System

3.8.1. IRIG-B Connection Pins

For connection to an IRIG-B time synchronization signal the terminals of the cable containing the IRIG-B signal must be connected to the respective terminals of the HX3040 CPU, according to the table below.

Name	CPU's Pin
IRIG-B +IN	1
IRIG-B -IN	2
IRIG-B +OUT	3
IRIG-B -OUT	4

Table 33: Connection between IRIG-B signals in CPU HX3040

> **ATTENTION**
>
> Prolonged short circuits to the IRIG-B connectors can damage the IRIG-B's output.

The figure below is the representation of the HX3040 CPU's IRIG-B connector.

altus
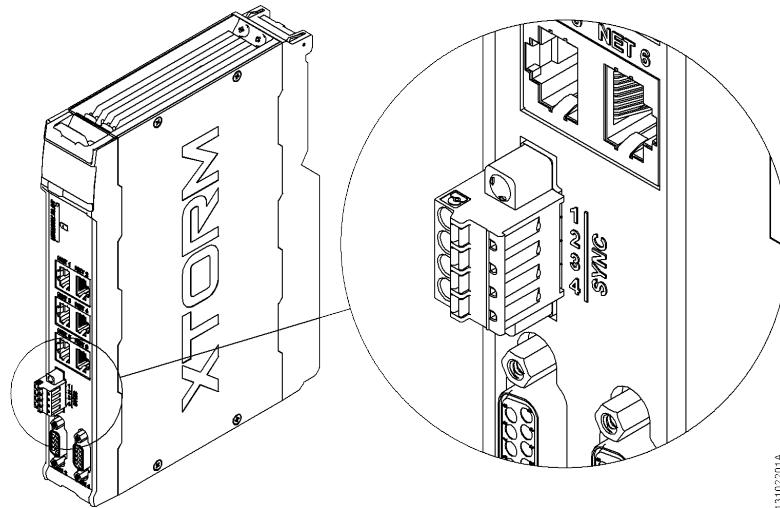
Figure 51: Interconnection Connector for IRIG-B Signal

3.9.   Memory Card Installation

This section describes how to insert the memory card into the Hadron Xtorm Series CPU, and for more information about using it, see section Memory Card.

Initially, attention must be paid to the correct position that the memory card must be inserted. One corner is different from the others and this corner should be used as a reference for the correct insertion of the card. Therefore, the memory card must be placed according to the drawing located on the front of the CPU or also as shown in the figure below.
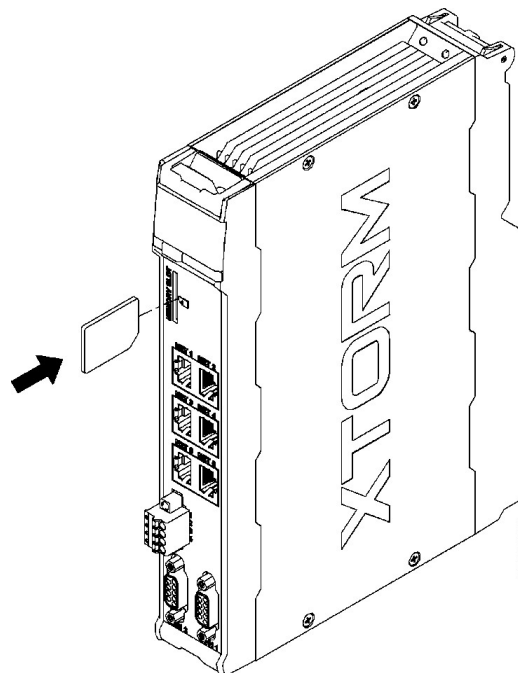
Figure 52: Memory Card Insertion into the CPU

When the card is properly installed, a symbol will appear on the CPU's graphical display. To remove the memory card, just access the *Memory Card* Menu on the CPU, using the diagnostics button, and select the memory card removal option, and wait until the card icon disappears from the graphic display status screen.

For this, the card must be pressed against the CPU until it clicks. Then, simply release it and remove it from the compartment, as shown in the figure below.
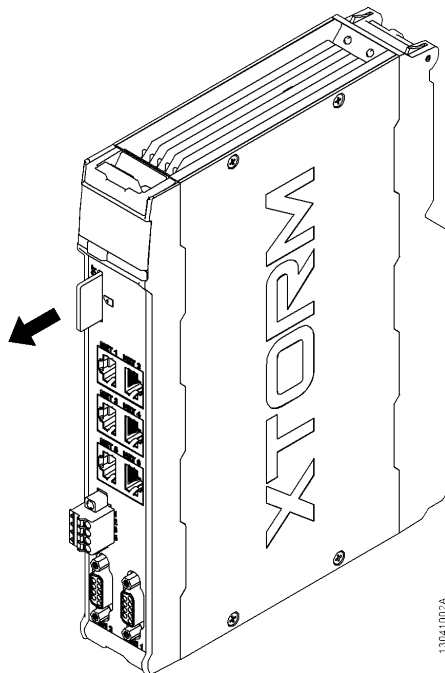
Figure 53: Memory Card Removal

3.10. Programmer Installation

To execute the MasterTool IEC XE development software installation, it is necessary to have the distribution CD-ROM or download the installation file from the site https://www.altus.com.br/en/. For further information about the step by step to installation, consult MasterTool IEC XE User Manual MT8500 – MU299609.

4. Configuration

The Hadron Xtorm Series CPU is configured and programmed using the MasterTool Xtorm software. The configuration performed defines the behavior and usage modes of the peripherals and CPU special features. The programming represents the application developed by the user, also known as application.

4.1. CPU Configuration

The parameters below are part of the CPU configuration inserted in the application. Each item must be properly reviewed for the correct operation of the project.

In addition to these parameters, it is possible to change the name and description of each module inserted in the application, for this, right-click on the module, in the item *Properties*, in the tab *Common*, change the name or description being both limited to 255 characters.

The rules and guidelines about tags, descriptions and names of I/O modules can be consulted in the section Accessing the module and I/O points description.

ATTENTION

It is recommended that the name of each module has only alphanumeric characters (upper or lower case and the text does not start with numbers). For the description we recommend alphanumeric characters (upper or lower case), white space and the period character ".". The use of any character other than the above is not recommended.

ATTENTION

When using the *ETD - Electronic Tag on Display* function to view the module description in the graphic display, the module description will be locked in the first 48 characters of the description.

ATTENTION

When using the *ETD - Electronic Tag on Display* or *Web Server* function to view the module name, the module name will be locked to the first 24 characters of the description.

4.1.1. General Parameters

The *General Parameters*, found on the initial CPU configuration screen, as shown in the figure below, are defined as:
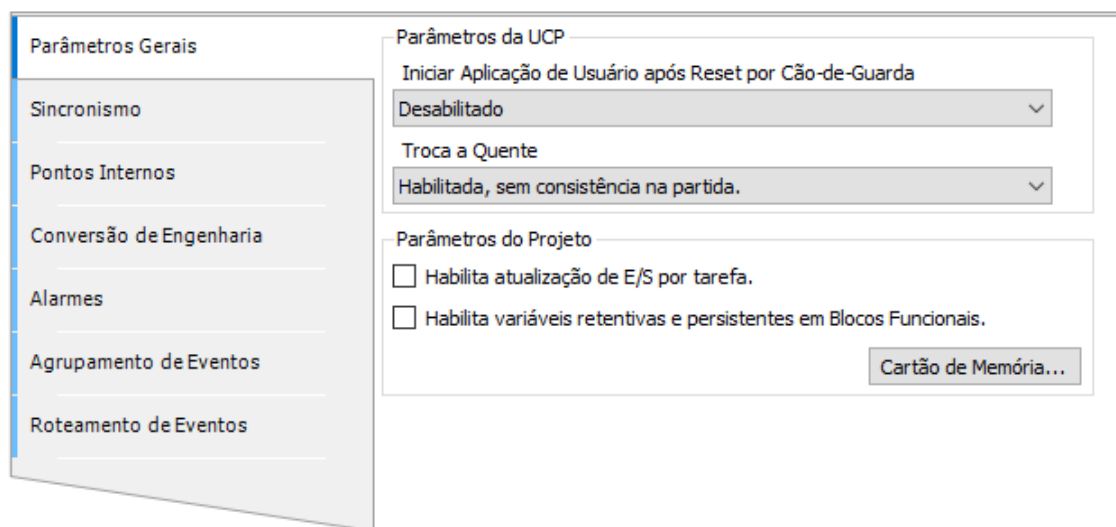
Figure 54: CPU General Parameters Configuration Screen

Configuration	Description	Default	Options
Start User Application After a Watchdog Reset	When enabled starts the user application after the hardware watchdog reset or through the Runtime restart, but keeps the indication via diagnostic LED button and variables.	Disabled	Enabled Disabled
Hot Swap Mode	Modules hot swap mode.	Enabled,without startup consistency	- Disabled for declared modules only: Configuration disabled for the modules which were declared in the configuration. - Enabled, with startup consistency for declared modules only: Configuration enabled with consistency in the startup only for the modules declared in the configuration. - Enabled,without startup consistency: Configuration enabled without consistency in the startup.
Enable I/O update per task	When enabled, performs the update of inputs and outputs on the context of the task where they are being used. For more information, see chapter I/O Scanning.	Disabled	Enabled Disabled

Table 34: CPU General Configurations

4.1.1.1. Hot Swap

The Hadron Xtorm Series CPU features the possibility of changing the bus I/O modules without shutting down the system and without information loss. This feature is known as hot swapping.

> **CAUTION**
>
> The Hadron Xtorm Series CPUs do not guarantee persistent and retentive variable retentivity in case the power supply, or the CPU is removed from the powered rack.

When hot swapping, the behavior of the related system changes according to the user-defined configuration, which presents the following options, as described in table 34.

- Disabled for Declared Modules Only
- Enabled, with Startup Consistency for Declared Modules Only
- Enabled, Without Statup Consistency

Thus, the user can choose how the system should behave in abnormal bus situations and when the CPU is in Run Mode. The table below shows the possible abnormal bus situations.

altus

Situation	Possible causes
Incompatible configuration	- Some module connected to the bus is different from the model that is declared in configuration.
Absent module	- The module was removed from the bus. - Some mal functioning module is not responding to CPU. - Some bus position is malfunctioning.

Table 35: Bus Abnormal Situations

For more information about the diagnoses corresponding to the situations described above, see section Diagnostics via Variables.

4.1.1.1.1.   Hot Swap Disabled, For Declared Modules Only

In this configuration, when an abnormal bus situation occurs (according to table 35) the CPU gets in *Stop Mode* in a time of up to 2 seconds, when the red LED starts blinking 4x (according to table 36). In order to get the CPU back to the normal *Run* state, in addition to undoing what caused the abnormal situation, it is necessary to perform any kind of *Reset* (this can be done by MasterTool Xtorm in the *Communication* menu).

> **ATTENTION**
>
> This option is not available for projects with CPU redundancy. For projects without CPU redundancy, the use of hot swapping disabled is also not indicated when using redundant power supplies.

4.1.1.1.2.   Hot Swap Enabled, with Startup Consistency for Declared Modules Only

The interval between CPU power-up (or reset command or application download) and the first time it enters Run Mode is considered *startup*. This setting checks if any abnormal bus situation occurred during startup, if so, the CPU enters Stop Mode and the DL LED starts blinking 4x (according to table 36). Afterwards, for the CPU to be put in Run Mode, and also to correct what caused the abnormal situation, a Reset command of any type must be executed, which can be done by the MasterTool Xtorm (Communication Menu).

After startup, if any module presents any of the situations mentioned in the previous table, the system will continue to work normally and will indicate the problem via diagnostics.

> **ATTENTION**
>
> - In this configuration, when there is a power failure (even if temporary), Warm Reset command, Cold Reset command, or when a new application has been downloaded, and some module is in an abnormal bus situation; the CPU will go into Stop Mode and the DL LED will start blinking 4x (according to the table below), because these are considered start situations.
> - This is the most recommended option, as it ensures the integrity of the system at startup and allows you to swap modules while the system is running.

4.1.1.1.3.   Hot Swap Enabled, without Startup Consistency

Allows the system to start up even when some module is in an abnormal bus situation (according to table 35). Abnormal situations are reported via diagnostics, both during and after startup.

> **ATTENTION**
>
> This option is recommended for the system implementation phase, as it allows the loading of new applications and the power off to be done without all the configured modules being present.

altus

4.1.1.1.4. How to perform the Hot Swap

> **CAUTION**
>
> Before hot-swapping, it is important to discharge any static potentials accumulated in the body. To do this, touch (with bare hands) a grounded metal surface before handling the modules. This ensures that the static electricity levels supported by the module are not exceeded.

> **ATTENTION**
>
> It is recommended that hot swap diagnostics be monitored in the user-developed control application to ensure that the value returned by the module is validated before being used.

The procedure for hot swapping modules is described in the sections Module Removal and Module Insertion.

In the case of output modules, it is desirable to have the points disconnected at the time of replacement to reduce arcing at the module connector. This can be done by switching off the field source or forcing the points via software tools. If the load is small, there is no need to turn off the points.

In the case of hot swapping power supplies, it is desirable that the external power supply of the module is turned off before removing the module from the bus in order to reduce arcing at the module connector. For inserting the power supply module into the bus, it is recommended to first mechanically fit the module to the bus and then connect the external power supply connector to the module.

It is important to note that in cases where the CPU goes into Stop Mode and the DL LED starts blinking 4x (according to table 36), due to any abnormal bus situation (according to table 35); the output modules put their points in fail-safe mode, in other words, turn them off (0 Vdc).

In the case of digital output modules, if the output module is removed from the powered bus, the logic state of the points is not changed. However, the physical point is de-energized. If the connector(s) are removed, the logic state of the points is not changed. However, the physical point is de-energized.

For analog RTD input modules, if the connector is removed from the powered bus, the logic state of the points will remain at the last value. If the connector(s) are removed, the logic state of the points will go to the full-scale value configured for each point, since the input impedance of each channel without the connector will tend towards infinity.

For analog voltage/current input modules, if it is removed from the powered bus, the logic state of the points will remain at the last value. If the connector(s) are removed, the logic state of the points will depend on the configuration type of each point, which will depend on the input type selected and the values configured for engineering scaling. For the 4-20 mA current scale the logic state of the point will depend on the value configured in the *Open Circuit Value* field.

> **ATTENTION**
>
> Always replace one module at a time, so that the CPU updates the modules' states.

The table 36 relates the bus conditions and the operating and DL LED status of the Hadron Xtorm CPU. For more information about the diagnostic LED states, see section Diagnostics via LED.

Condition	Enabled, with startup consistency for declared modules only	Enabled, without startup consistency	Disabled, for declared modules only
Non declared module	LED DL: Blinks 2x Blue Application: Run	LED DL: Blinks 2x Blue Application: Run	LED DL: Blinks 2x Blue Application: Run
Absent module	LED DL: Blinks 2x Blue Application: Run	LED DL: Blinks 2x Blue Application: Run	LED DL: Blinks 4x Red Application: Stop
Absent module (startup condition)	LED DL: Blinks 4x Red Application: Stop	LED DL: Blinks 2x Blue Application: Run	LED DL: Blinks 4x Red Application: Stop
Incompatible configuration	LED DL: Blinks 2x Blue Application: Run	LED DL: Blinks 2x Blue Application: Run	LED DL: Blinks 4x Red Application: Stop
Incompatible configuration (startup condition)	LED DL: Blinks 4x Red Application: Stop	LED DL: Blinks 2x Blue Application: Run	LED DL: Blinks 4x Red Application: Stop

altus

Condition	Enabled, with startup consistency for declared modules only	Enabled, without startup consistency	Disabled, for declared modules only
Duplicated slot address	LED DL: Blinks 4x Red Application: Stop	LED DL: Blinks 4x Red Application: Stop	LED DL: Blinks 4x Red Application: Stop
Non-operational module	LED DL: Blinks 4x Red Application: Stop	LED DL: Blinks 4x Red Application: Stop	LED DL: Blinks 4x Red Application: Stop

Table 36: Relation between Conditions and Hot Swaps

4.1.1.2. Memory Card

The following table shows the parameters available in the memory card configuration screen (figure 122).

Configurations	Description	Default	Possibilities
Copy Project from CPU to Memory Card	Copy the project from the CPU internal memory to the memory card	Disabled	- Enabled: Configuration enabled. - Disabled: Configuration disabled.
Password to Copy Project from CPU to Memory Card	Password for coping the project from the CPU internal memory to memory card	-	6 digits password (0 to 999999)
Copy Project from Memory Card to CPU	Copy the project from the memory card to the CPU internal memory	Disabled	- Enabled: Configuration enabled. - Disabled: Configuration disabled.
Password to Copy Project from Memory Card to CPU	Password for coping the project from the memory card to the CPU internal memory	-	6 digits password (0 to 999999)

Table 37: Memory Card Configuration Parameters

After setting up the project's copy possibilities and creating the startup application, the file *"Application.crc"* must be located for the settings regarding the memory card to take effect. This can be done in Select the "Application.crc" through the *" Find File..."* button, as can be seen in the picture 122.

4.1.2. Time Synchronization

For time synchronization, the HX3040 CPU uses the IRIG-B protocol, SNTP (Simple Network Time Protocol), control centers (DNP3 or IEC 60870-5-104 servers) or PTP (Precision Time Protocol), and for each of them, the user will have the option to enable/disable the protocol, as well as to choose the priority of receiving the synchronism signal among each of the protocols.

To use time synchronization protocols, you must configure the following parameters in the *Synchronism* tab, accessed through the HX3040 CPU, in the device tree:

altus
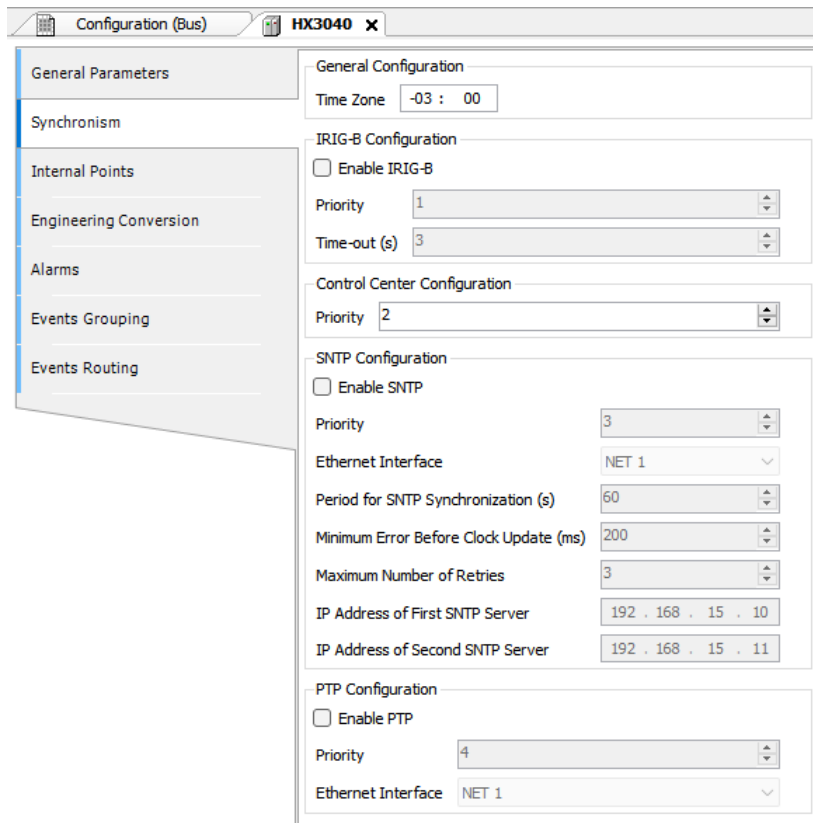
Figure 55: Time Synchronization Configuration

Configurations	Description	Default	Possibilities
Enable IRIG-B	Enables synchronism over the IRIG-B protocol.	Disabled	Disabled Enabled
Time Zone	Time zone. Difference between UTC (Universal Time Coordinated) time and local time.	-03:00	-12:59 to +13:59
Priority (IRIG-B)	Sets the priority of the IRIG-B synchronization source over the other sources. When enabled, the IRIG-B protocol synchronism must have priority over the other sources. Because of this, the field is read-only enabled.	1	1 2 3 4
Time-Out (s)	In case of IRIG-B signal failure, sets the maximum expected time (in seconds) for the restoration of the signal before looking for another synchronization source.	3	3 to 100
Priority (Control Center)	Defines the priority of the synchronism source per Control Center in relation to the others. The lower the field value, the higher its priority the synchronism source.	2	1 2 3 4
Enable SNTP	Enables synchronization over SNTP.	Disabled	Disabled Enabled

Configurations	Description	Default	Possibilities
Priority (SNTP)	Defines the priority of the synchronism source by SNTP in relation to the others. The lower the field value, the higher the priority of the synchronism source.	3	1 2 3 4
Ethernet Interface (SNTP)	Defines the Ethernet port through which communication is made with the SNTP server used for clock synchronization.	NET 1	NET 1 NET 2 NET 3 NET 4 NET 5 NET 6
Period for SNTP Synchronization (s)	The time period (in seconds) that must be waited between two synchronization requests.	60	1 to 255
Minimum Error Before Clock Update (x1 ms)	Before updating the time, the CPU compares the time received from the SNTP server with its current time. The CPU time will only be updated if the difference between the received time and the current time is higher than the time (in milliseconds) configured in this field.	200	0 to 65535
Maximum Number of Retries	This is the number of times the CPU will try to establish a connection to one of the SNTP servers. In case of an error in all attempts, in the next period it will try to connect to the other server.	3	0 to 100
IP Address of First SNTP Server	IP address of the first SNTP server to be used.	192.168.15.10	1.0.0.1 to 223.255.255.254
IP Address of Second SNTP Server	IP address of the second SNTP server to be used.	192.168.15.11	1.0.0.1 to 223.255.255.254
Enable PTP	Enables synchronization over PTP.	Disabled	Disabled Enabled
Priority (PTP)	Defines the priority of the synchronism source by PTP in relation to the others. The lower the field value, the higher the priority of the synchronism source.	4	1 2 3 4
Ethernet Interface (PTP)	Defines the Ethernet port through which communication is made with the PTP master used for clock synchronization.	NET 1	NET 1 NET 2 NET 3 NET 4 NET 5 NET 6

Table 38: Time Synchronization Settings

Notes:

Time Zone: The time zone setting is used to convert local time into UTC time and vice versa. While some synchronization sources use local time (IRIG-B, IEC 60870-5-104 protocol, SetDateAndTime function), others use UTC time (SNTP, DNP3 protocol). UTC time is usually used to stamp events (DNP3 and IEC 60870-5-104 protocols), while local time is used by other CPU functionality (GetDateAndTime function, OTD date and time information).

altus

Minimum Error Before Clock Update: If this parameter is set to zero, the clock will be updated on all SNTP synchronization requests.

4.1.2.1. IRIG-B

The time synchronization through the IRIG-B interface allows the RTU time setting with a maximum error of a few microseconds, and is recommended for applications that require high precision in event time stamping (1ms or less). The time is sent by the generating device (typically a GPS receiver) periodically every 1 second. The RTU's time is updated immediately after reception and is already used for the time stamp of the next events that occur. However, the internal RTC (which keeps the time when the RTU is switched off) is updated only every 10 seconds.

> **ATTENTION**
>
> Changes to the time zone at runtime when the time synchronization method is IRIG-B are not recommended. These changes can cause discontinuities in time stamps, which in turn would cause undesired behavior in the generation of certain events.

4.1.2.2. Control Center

In the case of synchronism through the control center, time synchronism must be enabled in the configuration screen of the communication protocol to be used (DNP3 Server, IEC 60870-5-104 Server) in order to receive clock synchronization. To configure the device with this option, see the parameters *Enable Synchronism* and *Synchronism Time-Out* available in section Configuration of the of DNP3 Server Application Layer, or the parameter *Enable Time Synchronization* available in section Application Layer.

> **ATTENTION**
>
> If the RTU receives a time synchronization command from the control center, and this option is disabled, an error response will be returned for that command. If the option is enabled, the RTU will return success to the control center, even if the sync command is discarded because there is an active sync method with higher priority.

This synchronization method should only be used as an auxiliary synchronization method, since the accuracy of the RTU clock synchronization process is highly dependent on network delay and traffic, as well as the processing load of the CPU, since this mechanism is handled by a low priority task.

> **ATTENTION**
>
> In architectures with UCP redundancy, the DNP3 Server and IEC 60870-5-104 Server drivers are disabled on the non-active UCP. Thus, it is not recommended to use these synchronization methods in redundant systems, because the non-active UCP may take several seconds after a switchover until its clock is synchronized. In redundant systems, the use of SNTP and/or IRIG-B is recommended.

4.1.2.3. SNTP

For synchronization via SNTP the CPU will behave as an SNTP client, that is, it will send time synchronization requests to an SNTP/NTP server, which can be on the local network or on the Internet. The accuracy of time synchronization over SNTP depends on the characteristics of the Ethernet network where it is being applied. If the SNTP Server is on the same local network as the RTU, in an ideal situation an accuracy of 1ms can be achieved. However, when the SNTP Server is in a remote network (Internet, etc.) this number can reach tens of milliseconds. That is, when a synchronization is performed, the updated time on the client may be a few milliseconds earlier or later than on the server.

The HX3040 CPU sends synchronization requests cyclically, according to the time set in the SNTP Synchronization Period field. On the first synchronization attempt, right after service initialization, the request is sent to the first server, configured in IP Address of the 1st Server. If this does not respond, the requests are directed to the second server, configured in IP Address of the 2nd server, providing a redundancy of SNTP servers. If the second server does not respond either, the same process of attempting synchronization is performed again, but only after the Synchronization Period has passed. That is, at each synchronization period the CPU tries to connect once to each server, it tries the second one if the first one does not respond. The timeout for a response from the SNTP server is set by default to 5 seconds and cannot be changed.

If, after a synchronization request, the difference between the current time of the HX3040 CPU and that received by the server is greater than the value configured in parameter *Minimum Error Before Clock Update*, the time of the HX3040 CPU is updated.

The SNTP client execution process can be exemplified with the following steps:

- Attempts to synchronize through the first server. If the synchronization is successful, the CPU waits the time for the new synchronization (Synchronization Period) and tries to synchronize again with this server, then using this as the primary server. In case of failure (the server does not respond in less than 5 s) step 2 is executed.
- Attempts to synchronize through the second server. If the synchronization is successful, the CPU waits the time for the new synchronization (Synchronization Period) and tries to synchronize again with this server, using it as the primary server. In case of failure (the server does not respond in less than 5 s) the time for the *Synchronization Period* is waited and step 1 is executed again.

Since the time to wait for a response from the SNTP server is 5 s, care must be taken when setting values smaller than 10 s for the *Synchronization Period*. If the primary server does not respond, the time for synchronization will be at least 5 s (waiting for the response from the primary server and trying to synchronize with the secondary server). If neither the primary server nor the secondary server responds, the time for synchronization will be at least 10 s (waiting for the response from both servers and a new connection attempt with the first server).

> **ATTENTION**
>
> - Depending on the subnet of the SNTP server, the client will use the Ethernet interface that is on the corresponding subnet to make synchronization requests. If the server is outside the two subnets of the NET 1 and NET 2 interfaces, the request can be made by whichever of the two interfaces can find a route to the server.
> - The SNTP Service depends on the user application only for its configuration. Therefore, this service will run even when the CPU is in STOP or BREAKPOINT modes, provided there is an application on the CPU with the SNTP client enabled and correctly configured.

> **CAUTION**
>
> If IRIG-B synchronization is not used, it is vital to set up at least one SNTP server. It is recommended to set up two SNTP servers (primary and secondary). SNTP synchronism is necessary to generate events with a consistent time stamp between CPUs A and B and with world time. Another use is to avoid discontinuities during a switchover in applications that reference date and time, since there is no date and time synchronism between the CPUs through the redundancy's synchronism channels.

4.1.2.4. PTP

The PTP (Precision Time Protocol), which has been implemented according to the IEEE 1588 standard, as time source. Additionally, it was built to only operate in slave mode and using the End-To-End (E2E) communication mode.

To do this, it is necessary for the user to enable the service in the synchronization tab of the CPU and configure its priority along with the communication port with the PTP master.

The PTP available in the Xtorm Series has been designed at the software level, so it has an accuracy of at least 1 ms and, as it only operates via Ethernet transport protocol level 2 (IEEE 802.3), it is not necessary for the CPU to be on the same subnet as the PTP master. Therefore, to establish seamless communication, the points mentioned above must be observed in the configuration of the PTP master that will be used in conjunction with the controller.

Furthermore, for this protocol to be used as the primary source of time synchronization, it needs to have communication between the configured interface and a PTP master. This source should have the lowest priority among those available at the time. The controller periodically checks the status of communication between the CPU and the PTP master in order to quickly identify any faults and inform the device so that another time source can take over as active.

> **ATTENTION**
>
> The PTP service depends on the user application only for its configuration. Therefore, this service will run even when the CPU is in STOP or BREAKPOINT modes, once there is an application on the CPU with the PTP enabled and correctly configured.

4.1.2.5. Daylight Saving Time (DST)

The daylight saving time setting must be done indirectly, using the function *SetTimeZone*, which changes the time zone applied to the RTC. At the beginning of daylight saving time, the function should be used to increase the time zone by one hour. At the end of daylight saving time, it is used again to decrease it by one hour.

For more information, see the section Function Blocks for RTC Reading and Writing.

4.1.3. Internal Points

A communication point is stored in the RTU's memory in the form of two distinct variables. One of them represents the value of the point (type BOOL, BYTE, WORD, etc...), while the other represents its quality (type QUALITY). Internal points are those whose value and quality are calculated internally by the user application, that is, they do not have an external origin, as occurs for points associated with IEDs (Master/Client communication drivers) or I/O modules.

The function of this Internal Points configuration tab is to relate the variable representing the value of a point to the variable representing its quality. It should be used to relate value and quality variables created internally in the RTU program (as in a GVL for example), which will typically be mapped later to a Server type communication driver for communication with the control center.

> **ATTENTION**
>
> If a value variable does not have a related quality variable, a constant standard good quality (no significant indication) will be reported when the value variable is reported to a customer or control center.

Thus, the purpose of this tab is not to create or declare internal points. To do this, simply declare the value and/or quality variables in a GVL and map it to the communication driver.

The configuration of the internal points, visualized in the following figure, follows the parameters described below. It is possible to configure up to 5120 entries in the Internal Points table.
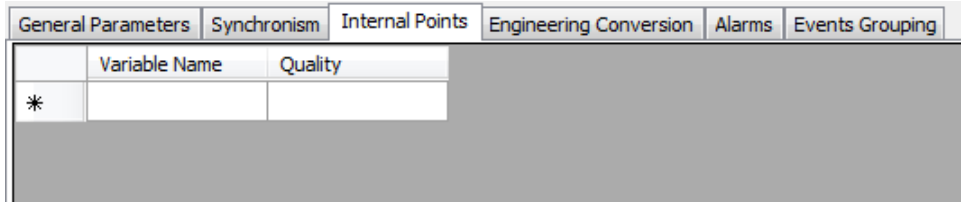
Figure 56: Internal Points Settings

Configurations	Description	Default	Possibilities
Variable Name	Symbolic variable that stores the value of the internal point.	-	It accepts variables of type BOOL, WORD, DWORD, LWORD, INT, DINT, LINT, UINT, UDINT, ULINT, REAL, LREAL or DBP. The variable can be simple, array or array element and can be in structures.
Quality	Symbolic variable that stores the quality of the internal point represented in "Variable Name".	-	Variables of type QUALITY (LibRtuStandard), which can be single, array or array element and can be in structures.

Table 39: Internal Points Settings

The figure below shows an example of an internal point configuration:
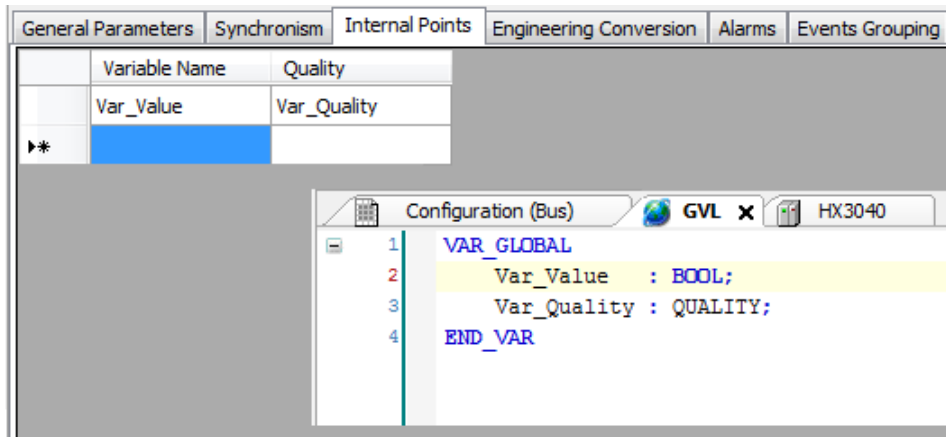
Figure 57: Internal Point Configuration Example

4.1.3.1. Quality Conversions

The quality of an internal point is information that indicates the level of confidence that can be had in the value that is stored in that point. The quality may tell the user, for example, that the stored value is out of scale, or that it is valid but unreliable.

The IEC 61850, DNP3 and IEC 60870-5-104 standards have their own formats for representing the quality information of a point. The Hadron Xtorm Series, on the other hand, has its own quality format (but very similar to IEC 61850) called *Internal Quality*. This format is defined by the QUALITY type (LibRtuStandard library) and is used internally for quality storage, allowing conversions between protocols without loss of information.

When mapping the same communication point between two drivers, quality conversion is performed automatically in two stages. For example: if a communication point is mapped from a DNP3 Client driver to an IEC 61850 Server driver, first the quality will be converted from DNP3 format to internal format (and stored internally in the CPU), and then it will be converted from internal format to IEC 61850 format.

The following tables define the conversions from the proprietary formats of the protocols to the internal format. If the conversion between protocols needs to be queried, it is necessary to perform the two-stage analysis by querying each of the tables for the internal format and then performing the correlation between them.

> **ATTENTION**
>
> In the case of internal points mapped to communication drivers, it is not recommended to change the value of quality flags that have no correspondence in the protocol in question (in other words, that are not in the following tables). This will result in the generation of events equal to the previous one (with the most recent time stamp) and, thus, depending on the configuration selected for the transmission mode of the analog input events, may overwrite the previous event if it has not yet been delivered to the control center.

4.1.3.1.1. Internal Quality

This is the QUALITY structure, the table below shows each of its Variables in detail.

Bit	Name	Type	Description
0	FLAG_RESTART	BOOL	The RESTART flag indicates that the data has not been updated by the field since the device was Reset.
1	FLAG_COMM_FAIL	BOOL	Indicates that there is a communication failure in the path between the device where the data originates and the reporting device.

altus

Bit	Name	Type	Description
2	FLAG_REMOTE_SUBSTITUTED	BOOL	If TRUE, the data value is replaced in a remote communication device.
3	FLAG_LOCAL_SUBSTITUTED	BOOL	If TRUE, the data value is replaced by the device that generated this flag. This behavior may have occurred due to a diagnostic or temporary operation due to human intervention.
4	FLAG_FILTER	BOOL	Flag used to signal and prevent overloading of event communication channels, such as oscillations (rapid changes) in the binary inputs.
5	FLAG_OVERFLOW	BOOL	This identifier must indicate a quality problem, where the value of the attribute that the quality is associated with is beyond the capability of representation.
6	FLAG_REFERENCE_ERROR	BOOL	This identifier must indicate that the value may not be a correct value due to the reference being out of calibration.
7	FLAG_INCONSISTENT	BOOL	This identifier must indicate that an evaluation function has detected an inconsistency.
8	FLAG_OUT_OF_RANGE	BOOL	This identifier must indicate a quality error that the attribute with which the quality was associated is beyond the capability of predefined values.
9	FLAG_INACCURATE	BOOL	This identifier must indicate that the value does not meet the stated precision of the source.
10	FLAG_OLD_DATA	BOOL	A value seems to be out of date, if an update does not occur in a specified period of time.
11	FLAG_FAILURE	BOOL	This identifier should indicate that a supervisory function has detected an internal or external failure.
12	FLAG_OPERATOR_BLOCKED	BOOL	Update has been blocked by the operator.
13	FLAG_TEST	BOOL	*Test* should be an additional identifier that can be used to classify a value as a test value and that will not be used for operational purposes.
14-15	RESERVED	-	Reserved.

altus

Bit	Name	Type	Description
16-17	VALIDITY	QUALITY_VALIDITY	0 – Good (Reliable value, no abnormal condition) 1 – Invalid (Value does not correspond to the value in IED) 2 – Reserved (Reserved) 3 – Questionable (Present value may not be the same as IED)

Table 40: QUALITY Structure

4.1.3.1.2. DNP3 Conversion

The 41 table illustrates the DNP3 quality conversion for internal points, and the 42 table lists the Hadron Xtorm Series internal points to DNP3 quality conversion available for Mastertool.

DNP3 -> Internal Points		
	Internal Quality	
DNP3 Quality	Flags	VALIDITY
ONLINE	-	if just ONLINE, VALIDITY_GOOD
NOT ONLINE	FLAG_OLD_DATA	VALIDITY_QUESTIONABLE
RESTART	FLAG_RESTART	VALIDITY_QUESTIONABLE
COMM_LOST	FLAG_COMM_FAIL and FLAG_OLD_DATA	VALIDITY_QUESTIONABLE
REMOTE_FORCED	FLAG_REMOTE_SUBSTITUTED and FLAG_OPERATOR_BLOCKED	if just REMOTE_FORCED and ONLINE, VALIDITY_GOOD
LOCAL_FORCED	FLAG_LOCAL_SUBSTITUTED and FLAG_OPERATOR_BLOCKED	if just LOCAL_FORCED and ONLINE, VALIDITY_GOOD
CHATTER_FILTER	FLAG_FILTER	VALIDITY_QUESTIONABLE
OVER_RANGE	FLAG_OVERFLOW and FLAG_OUT_OF_RANGE	VALIDITY_INVALID
DISCONTINUITY	FLAG_REFERENCE_ERROR	VALIDITY_QUESTIONABLE
REFERENCE_ERR	FLAG_INACCURATE and FLAG_REFERENCE_ERROR	VALIDITY_QUESTIONABLE

Table 41: DNP3 to Internal Points Conversion

Internal Points -> DNP3		
Internal Quality		
Flags	VALIDITY	DNP3 Quality
FLAG_RESTART	ANY	RESTART
FLAG_COMM_FAIL	ANY	COMM_LOST
FLAG_REMOTE_SUBSTITUTED	ANY	REMOTE_FORCED
FLAG_LOCAL_SUBSTITUTED	ANY	LOCAL_FORCED
FLAG_FILTER	ANY	CHATTER_FILTER
FLAG_OVERFLOW	ANY	OVER_RANGE
FLAG_REFERENCE_ERROR	ANY	DISCONTINUITY (for counters) REFERENCE_ERR (for analog)
FLAG_INCONSISTENT	ANY	not ONLINE

altus

Internal Points -> DNP3		
Internal Quality		
Flags	VALIDITY	DNP3 Quality
FLAG_OUT_OF_RANGE	ANY	OVER_RANGE
FLAG_INACCURATE	ANY	REFERENCE_ERR
FLAG_OLD_DATA	ANY	not ONLINE
FLAG_FAILURE	ANY	not ONLINE
FLAG_OPERATOR_BLOCKED and NOT LOCAL_SUBSTITUTED or/and NOT REMOTE_SUBSTITUTED	ANY	LOCAL_FORCED
FLAG_TEST	ANY	not ONLINE
NOT FLAG_OLD_DATA and NOT FLAG_FAILURE and NOT FLAG_INCONSISTENT and NOT FLAG_TEST	VALIDITY_INVALID	INVALID

Table 42: Internal Points to DNP3 Conversion

4.1.3.1.3. *IEC 60870-5-104 Conversion*

The tables 43 to 45 present respectively the conversion of digital, analog, Step Position and counters IEC 60870-5-104 to internal point quality. The tables 46 to 48 present the conversion of digital, analog, Step Position, Bitstring and counter internal points quality to IEC 60870-5-104 quality of Hadron Xtorm Series available for Mastertool.

IEC 60870-5-104 Digital -> Internal Points		
	Internal Quality	
IEC 60870-5-104 Quality	Flags	VALIDITY
BLOCKED	FLAG_OPERATOR_BLOCKED	if just ONLINE, VALIDITY_QUESTIONABLE
SUBSTITUTED	FLAG_REMOTE_SUBSTITUTED	if just SUBSTITUTED, VALIDITY_GOOD
NOT TOPICAL	FLAG_OLD_DATA	VALIDITY_QUESTIONABLE
INVALID	FLAG_FAILURE	VALIDITY_INVALID
IEC 60870-5-104 driver started, but has not yet communicated successfully	FLAG_RESTART	VALIDITY_INVALID
Communication failure, and it was already communicating	FLAG_COMM_FAIL	VALIDITY_QUESTIONABLE

Table 43: IEC 60870-5-104 Digital to Internal Points Conversion

IEC 60870-5-104 Analog and Step Position -> Internal Points		
	Internal Quality	
IEC 60870-5-104 Quality	**Flags**	**VALIDITY**
OVERFLOW	FLAG_OVERFLOW	VALIDITY_INVALID
BLOCKED	FLAG_OPERATOR_BLOCKED	if just BLOCKED, VALIDITY_QUESTIONABLE
SUBSTITUTED	FLAG_REMOTE_SUBSTITUTED	if just SUBSTITUTED, VALIDITY_GOOD
NOT TOPICAL	FLAG_OLD_DATA	VALIDITY_QUESTIONABLE
INVALID	FLAG_FAILURE	VALIDITY_INVALID
IEC 60870-5-104 driver started, but has not yet communicated successfully	FLAG_RESTART	VALIDITY_INVALID
Communication failure, and it was already communicating	FLAG_COMM_FAIL	VALIDITY_QUESTIONABLE

Table 44: IEC 60870-5-104 Analog and Step Position to Internal Points Conversion

IEC 60870-5-104 Counters -> Internal Points		
	Internal Quality	
IEC 60870-5-104 Quality	**Flags**	**VALIDITY**
CARRY	FLAG_OVERFLOW	VALIDITY_GOOD
INVALID	FLAG_FAILURE	VALIDITY_INVALID
IEC 60870-5-104 driver started, but has not yet communicated successfully	FLAG_RESTART	VALIDITY_INVALID
Communication failure, and it was already communicating	FLAG_COMM_FAIL	VALIDITY_QUESTIONABLE

Table 45: IEC 60870-5-104 Counters to Internal Points Conversion

Internal Points -> IEC 60870-5-104 Digital		
Internal Quality		
Identifiers	**VALIDITY**	**IEC 60870-5-104 Quality**
FLAG_RESTART	ANY	NOT TOPICAL
FLAG_COMM_FAIL	ANY	NOT TOPICAL
FLAG_REMOTE_SUBSTITUTED	ANY	SUBSTITUTED
FLAG_LOCAL_SUBSTITUTED	ANY	SUBSTITUTED
FLAG_FILTER	ANY	-
FLAG_OVERFLOW	ANY	-

Internal Points -> IEC 60870-5-104 Digital		
Internal Quality		
Identifiers	**VALIDITY**	**IEC 60870-5-104 Quality**
FLAG_REFERENCE_ERROR	ANY	-
FLAG_INCONSISTENT	ANY	-
FLAG_OUT_OF_RANGE	ANY	-
FLAG_INACCURATE	ANY	-
FLAG_OLD_DATA	ANY	NOT TOPICAL
FLAG_FAILURE	ANY	INVALID
FLAG_OPERATOR_BLOCKED	ANY	BLOCKED
FLAG_TEST	ANY	-
ANY	VALIDITY_INVALID	INVALID

Table 46: Conversion of Internal Points to IEC 60870-5-104 Digital

Internal Points -> IEC 60870-5-104 Analog, Step Position and Bitstring		
Internal Quality		
Identifiers	**VALIDITY**	**IEC 60870-5-104 Quality**
FLAG_RESTART	ANY	NOT TOPICAL
FLAG_COMM_FAIL	ANY	NOT TOPICAL
FLAG_REMOTE_SUBSTITUTED	ANY	SUBSTITUTED
FLAG_LOCAL_SUBSTITUTED	ANY	SUBSTITUTED
FLAG_FILTER	ANY	-
FLAG_OVERFLOW	ANY	OVERFLOW
FLAG_REFERENCE_ERROR	ANY	INVALID
FLAG_INCONSISTENT	ANY	INVALID
FLAG_OUT_OF_RANGE	ANY	OVERFLOW
FLAG_INACCURATE	ANY	INVALID
FLAG_OLD_DATA	ANY	NOT TOPICAL
FLAG_FAILURE	ANY	INVALID
FLAG_OPERATOR_BLOCKED	ANY	BLOCKED
FLAG_TEST	ANY	-
ANY	VALIDITY_INVALID	INVALID

Table 47: Conversion of Internal Points to IEC 60870-5-104 Analog, Step Position and Bitstring

Internal Points -> IEC 60870-5-104 Counters		
Internal Quality		
Identifiers	**VALIDITY**	**IEC 60870-5-104 Quality**
FLAG_RESTART	ANY	-
FLAG_COMM_FAIL	ANY	-
FLAG_REMOTE_SUBSTITUTED	ANY	-
FLAG_LOCAL_SUBSTITUTED	ANY	-
FLAG_FILTER	ANY	-
FLAG_OVERFLOW	ANY	OVERFLOW
FLAG_REFERENCE_ERROR	ANY	-
FLAG_INCONSISTENT	ANY	-

Internal Points -> IEC 60870-5-104 Counters		
Internal Quality		
Identifiers	**VALIDITY**	**IEC 60870-5-104 Quality**
FLAG_OUT_OF_RANGE	ANY	-
FLAG_INACCURATE	ANY	-
FLAG_OLD_DATA	ANY	-
FLAG_FAILURE	ANY	INVALID
FLAG_OPERATOR_BLOCKED	ANY	-
FLAG_TEST	ANY	-
ANY	VALIDITY_INVALID	INVALID

Table 48: Conversion of Internal Points to IEC 60870-5-104 Counters

4.1.3.1.4. IEC 61850 Conversion

The table below lists the type of internal point quality with the IEC 61850 quality in the Hadron Xtorm Series available for Mastertool.

Internal Points -> IEC 61850		
Internal Quality Flags	**VALIDITY IEC 61850**	**IEC 61850 Flags**
FLAG_RESTART	VALIDITY_QUESTIONABLE	OLDDATE
FLAG_COMM_FAIL	VALIDITY_QUESTIONABLE	OLDDATE
FLAG_REMOTE_SUBSTITUTED	GOOD	Source = SUBSTITUTED
FLAG_LOCAL_SUBSTITUTED	GOOD	Source = SUBSTITUTED
FLAG_FILTER	VALIDITY_QUESTIONABLE or VALIDITY_INVALID	OSCILLATORY
FLAG_OVERFLOW	VALIDITY_INVALID	OVERFLOW
FLAG_REFERENCE_ERROR	VALIDITY_QUESTIONABLE or VALIDITY_INVALID	BADREFERENCE
FLAG_INCONSISTENT	VALIDITY_QUESTIONABLE	INCONSISTENT
FLAG_OUT_OF_RANGE	VALIDITY_QUESTIONABLE or VALIDITY_INVALID	OUTOFRANGE
FLAG_INACCURATE	VALIDITY_QUESTIONABLE	INNACURATE
FLAG_OLD_DATA	VALIDITY_QUESTIONABLE	OLDDATE
FLAG_FAILURE	VALIDITY_INVALID	FAILURE
FLAG_OPERATOR_BLOCKED	GOOD	OPERATORBLOCKED
FLAG_TEST	GOOD	TEST

Table 49: Internal Points to IEC 61850 conversion

The process of converting the internal quality format to IEC 61850 follows the recommendations defined by the IEC 61850 standard. According to the standard, the validity field is defined as a function of the quality flags. Some flags allow the validity to assume either the value VALIDITY_QUESTIONABLE or VALIDITY_INVALID, in these cases the assumed validity is the one defined in the point to be converted.

4.1.3.1.5. MODBUS Internal Quality

As the MODBUS standard does not specify quality types for each point, but to help the use of communication diagnostics for each point, the Hadron Xtorm Series allows through its own internal structure the mapping of quality variables for each MODBUS point. The table below describes the quality types that each MODBUS point can assume.

Resulting Quality	Resulting VALIDITY	Description
FLAG_RESTART	VALIDITY_INVALID	Initialization value. The point has never been updated.
No error flag (0)	VALIDITY_GOOD	Communication OK. The point is updated.
FLAG_COMM_FAIL and FLAG_RESTART	VALIDITY_INVALID	Communication error. Point was never updated.
FLAG_COMM_FAIL and FLAG_OLD_DATA	VALIDITY_QUESTIONABLE	An error occurred but the point was updated and has an old value.
FLAG_FAILURE and FLAG_RESTART	VALIDITY_INVALID	Received an exception response, and the point is still at the initial value.
FLAG_FAILURE and FLAG_OLD_DATA	VALIDITY_QUESTIONABLE	Received an exception response, but the point has a valid old value.
FLAG_RESTART and FLAG_OLD_DATA	VALIDITY_QUESTIONABLE	Device is stopped. The point contains an old value.

Table 50: MODBUS Quality

4.1.3.1.6. I/O Module Quality

To assist in the use of diagnostics for each I/O point, Hadron Xtorm Series automatically creates a quality structure for each module used in the RTU design, through its own internal structure accessible via structure QUALITY, available in the GVL IOQualities automatically created by the Mastertool template.

The table below describes the quality types of each I/O point.

For more information see the section GVL IOQualities.

Diagnostic	Resultant Flags	Resultant VALIDITY	Description
Don't care	FLAG_RESTART	VALIDITY_INVALID	Quality has this value before it is read or written for the first time.
None	-	VALIDITY_GOOD	Communication OK. The point is updated.
None	FLAG_OLD_DATA and FLAG_FAILURE	VALIDITY_QUESTIONABLE	Non-operational module, however data has been read/written at least once.
bOverRange or bUnderRange	FLAG_OUT_OF_RANGE	VALIDITY_INVALID	The input value is over or under the allowed range.
bInputNotEnabled or bOutputNotEnabled	FLAG_OPERATOR_BLOCKED	VALIDITY_INVALID	Input/output is not enabled.
bOpenLoop	FLAG_FAILURE	VALIDITY_INVALID	Input/Output with Open loop.
bDisconnected	FLAG_FAILURE	VALIDITY_INVALID	Input/output disconnected.
bHwFailure	FLAG_FAILURE	VALIDITY_INVALID	Input/Output with hardware failure.
bFatalError	FLAG_FAILURE	VALIDITY_INVALID	Fatal hardware failure.

altus

Diagnostic	Resultant Flags	Resultant VALIDITY	Description
bNoExternalSupply	FLAG_FAILURE	VALIDITY_INVALID	External power supply is below the minimum operating limit.
bShortCircuit or bOutputShortCircuit	FLAG_FAILURE	VALIDITY_INVALID	Short-circuit in the output.
bCalibrationError	FLAG_INACCURATE	VALIDITY_INVALID	Calibration error.
bColdJunctionSensorError	FLAG_INACCURATE	VALIDITY_INVALID	Cold junction sensor error.
bWrongSequence	FLAG_FAILURE	VALIDITY_INVALID	Configured phase sequence differs from the observed one. This diagnostic affects the quality of a subset of the module inputs, the calculation of which depends on a correct phase sequence.

Table 51: I/O Module Quality

4.1.4. Engineering Conversion

The Engineering Conversion tab is intended to facilitate linear conversion of scales. It is common that a given input or output signal needs to be converted from a scale that refers to the limits of a sensor, for example, to a different scale that has representativeness in the context where this signal is interpreted. For this purpose the user of a control system is usually faced with the demand to convert engineering units. This conversion is usually associated with the relationship between physical points and internal points.

The configuration of Engineering Conversions, shown in the figure below, follows the parameters described in the following table. It is possible to configure up to 5120 inputs in the Engineering Conversion table.

Figure 58: Engineering Conversion Parameters Configuration Screen

altus

Configuration	Description	Default	Possibilities
Input	Symbolic input variable of the conversion.	-	It accepts symbolic variables of type INT, DINT, UINT, UDINT, WORD and DWORD. The variable can be simple, array or array element and can be in structures.
Engineering	Symbolic variable that stores the result of the conversion.	-	Accepts only variables of type REAL. The variable can be simple, array or array element and can be in structures.
Minimum Input Value	Lowest input value.	-	Numeric constant that must obey the limits of the variable type used in "Input".
Maximum Input Value	Highest input value.	-	Numeric constant that must obey the limits of the variable type used in "Input".
Minimum Engineering Value	Lowest engineering value.	-	Numeric constant that must obey the limits of type REAL.
Maximum Engineering Value	Highest engineering value.	-	Numeric constant that must obey the limits of type REAL.

Table 52: Configuration of Engineering Conversion Parameters

4.1.5. Alarms

Alarms are variables that store a binary value, which can be true or false, and this value is the result of comparing an input variable with a given setpoint (reference value). The value of an alarm is usually used to trigger information events of application or process states.

The configuration of the Alarms, visualized in the figure below, follows the parameters described below. It is possible to configure up to 5120 entries in the Alarms table.
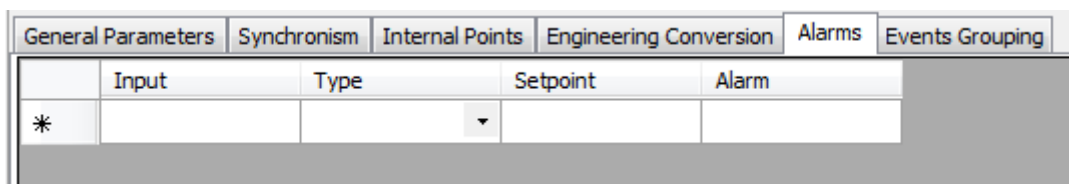
Figure 59: Alarm Settings Screen

Configurations	Description	Default	Possibilities
Input	Symbolic variable that contains the information used for alarm generation.	-	It accepts variables of type INT, DINT, LINT, UINT, UDINT, ULINT, WORD, DWORD, LWORD, REAL or LREAL. The variable can be simple, array or array element and can be in structures.
Type	Type of comparison that will be performed between the input and the setpoint.	-	It may take two pre-set values: " Higher than" or "Lower than".

altus

Configurations	Description	Default	Possibilities
Setpoint	Reference value used to identify if the input value has generated an alarm or not.	-	It accepts symbolic variables of type INT, DINT, LINT, UINT, UDINT, ULINT, WORD, DWORD, LWORD, REAL or LREAL. The variable can be simple, array or array element and can be in structures. Numeric constants respecting the limits of types LINT, LWORD and LREAL are also accepted.
Alarm	Symbolic variable that will receive the value TRUE or FALSE, depending on the result of the comparison between "Input" and "Setpoint".	-	Accepts only variables of type BOOL. The variable can be simple, array, or array element, and can be in structures.

Table 53: Alarm Configuration

4.1.6. Event Grouping

It is possible to create several event groupings in an Xtorm RTU. Each event grouping allows you to summarize up to 128 digital events into a single new digital event. This allows a server configured in the Xtorm RTU to report a lower number of events for certain clients.

Digital grouped events can be derived from:

- Local digital inputs, installed on the Xtorm RTU bus;
- Remote digital inputs (events received by the Xtorm RTU from external servers, via client communication drivers configured in the Xtorm RTU);
- Digital internal points.

The digital inputs or digital internal points that make up an event grouping can be single (value attribute of type BOOL) or double (value attribute of type DBP). They also have a quality attribute (variable of type QUALITY). The summary event of each grouping has a value attribute of type BOOL and a quality attribute of type QUALITY.

The Xtorm RTU automatically calculates the value, quality, and time stamp of the summary events of all the configured groupings. To do this, it considers the value, quality, and time stamp of the digital events that make up these groupings.

The following sections describe how the event groupings are configured and how the value, quality, and time stamp calculations of the summary events of the groupings are made.

4.1.6.1. Event Grouping Configuration

The following figure shows an example of a screen containing the configuration of two event groupings.

altus
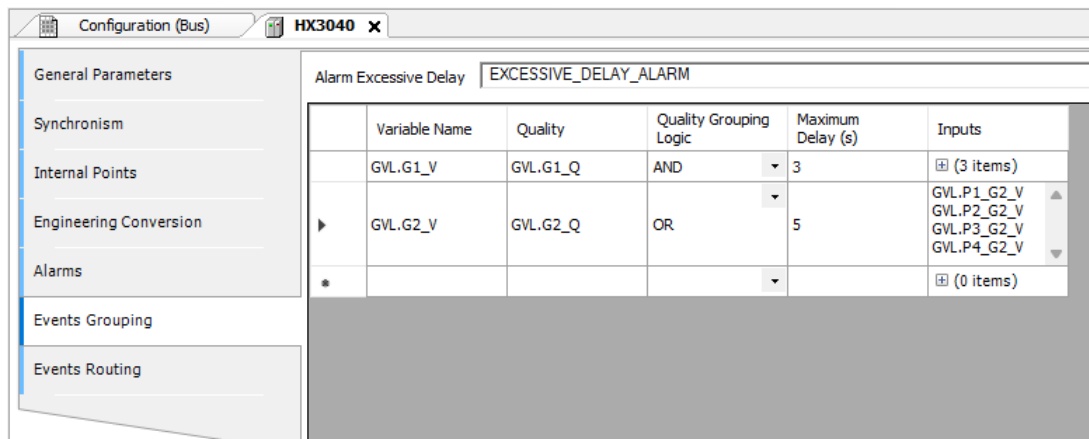
Figure 60: Event Grouping Configuration Screen

At the top of the screen, you must configure a *"Alarm Excessive Delay"*, common to all event groupings. This is a BOOL-type variable that the user must create. The example above shows the variable **EXCESSIVE_DELAY_ALARM**. For more details on the function of this variable, see section Using the Grouping's Maximum Delay Configuration.

Then, a table must be filled in where each row contains the configuration of an event grouping. Each row contains 5 columns:

1. The *"Variable Name"* column contains the name of a BOOL variable that corresponds to the value attribute of the summary event. It must be created, for example, in a GVL. The calculation of this value attribute is described in section Calculation of the Event Value Attribute Summary of a Grouping.

2. The *"Quality"* column contains the name of a variable of type QUALITY, which corresponds to the quality attribute of the summary event. It must be created, for example, in a GVL. The calculation of this quality attribute is described in section Calculation of the Event Quality Attribute Summary of a Grouping.

3. The *"Quality Grouping Logic"* column allows you to select between two methods for calculating the quality attribute of the summary event (AND or OR). More details in the section Calculation of the Event Quality Attribute Summary of a Grouping.

4. The *"Maximum Delay (s)"* column allows you to set a delay that needs to be used when calculating the value and quality of the summary event. The delay value can vary between 1 and 60 seconds. More details in the section Using the Grouping's Maximum Delay Configuration.

5. The *"Inputs"* column defines a list of up to 128 digital variables that make up the grouping.

These are the value attributes of these variables, which can be of type BOOL (single variables) or DBP (double variables). To insert several variables in this column, use the *SHIFT+ENTER* keys after each variable.

ATTENTION

The summary value and quality of the grouping should not be confused with internal points. They should not be inserted in the *"Internal Points"* tab below the HX3040 CPU. They should only be created (in a GVL or POU) and configured in the *"Events Grouping"* tab below the HX3040 CPU.

4.1.6.2. Using the Grouping's Maximum Delay Configuration

The various events that compose a grouping can arrive from different servers and have different propagation delays. So they can arrive at the Xtorm RTU out of chronological sequence. To correctly calculate summary events, the events that make up each cluster must be evaluated in chronological order. This imposes some requirements:

- It is essential that the Xtorm RTU and all the servers that report events to it are synchronized with good precision.
- When an event that makes up a cluster reaches the Xtorm RTU, its age cannot exceed the maximum delay configured for the cluster:
  - The age of an event corresponds to the difference between the current time of the Xtorm RTU and the time stamp of the event.
  - Therefore, the user must configure a maximum delay greater than the sum of the delays involved from the time an event is detected on a server until it reaches the Xtorm RTU (internal server delays, communication delays, etc.).

altus

- If the age of an event is greater than the maximum delay when the event reaches the Xtorm RTU, this event is disregarded in the calculations. In addition, the *"Alarm Excessive Delay"* is activated for a short time. The activation of this alarm indicates that the calculation of the summary event may be incorrect.
  - It's recommended that an event be configured for the *"Alarm Excessive Delay"* variable and that this event be reported to the same client that receives the grouped events. This way, this client will know that there may be incorrect calculations of grouped events at times close to the event stamp of this alarm.

- It is possible to configure the summary event of an X cluster as an event that makes up another Y cluster, creating a type of chaining of clustered events. In this case, the maximum delay configured for cluster Y must be greater than the maximum delay configured for cluster X.
- The calculation procedure for a summary event requires the events in the cluster to be sorted chronologically. In addition, before processing an incoming event that makes up the cluster, it must wait until the age of this event is equal to the maximum delay setting of the cluster.
  - To carry out this procedure, the Xtorm RTU uses an auxiliary queue with a capacity of 8192 events shared by all the clusters. In this queue, events are sorted and stored until they reach the age required to be processed (the maximum delay setting).
  - This queue size is sufficient to guarantee correct operation under normal operating conditions. If a large number of events are generated during the maximum delay time to fill this queue, the most recent events will be discarded, keeping the oldest.

4.1.6.3. Calculation of the Event Value Attribute Summary of a Grouping

The value of the summary event corresponds to the *"or logical"* of the values of the events that compose the grouping.

The following figure shows an example of a cluster whose summary event value is G and whose cluster component event values are P1, P2, and P3.

Figure 61: Calculation of the summary event value attribute

Note that in this example, two events are generated for G:
- In the first event, the value of G rises to 1 (TRUE) and receives the time stamp of the rise of point P2 (the first point that rises).
- In the second event, the value of G drops to 0 (FALSE) and receives the time stamp of the descent of point P1 (the last point to descend).

In other words, the value G is a *"or logical"* of the values P1, P2, and P3.

For a user monitoring the values of P1, P2, P3 and G using Mastertool Xtorm, or on the HMI of a customer connected to Xtorm, the time diagram in the previous figure does not seem to show reality. The user should consider the following observations:
- When monitoring with Mastertool Xtorm, the G value rises after the P2 point. The delay can reach the value of the maximum delay configured for the cluster, due to the reasons explained in section Using the Grouping's Maximum Delay Configuration. This is because the G value is only changed to 1 once the age of the event at point P2 is equal to the value of the maximum delay. However, the time stamp assigned to the ascent event in G will be the same as that of the ascent event at point P2.
- When monitoring with Mastertool Xtorm, the G value drops after the P1 point. The delay can reach the value of the maximum delay configured for the cluster due to the reasons explained in the section Using the Grouping's Maximum Delay Configuration. This is because the G value is only changed to 0 once the age of the event at point P1 is equal to the value of the maximum delay. However, the time stamp assigned to the descent event in G will be the same as that of the descent event at point P1.

Some additional observations should be considered when an event that makes up the grouping is double (DBP type). The DBP type is a structure with two fields: OFF and ON, both of BOOL type.
- If OFF = TRUE and ON = FALSE, the event value will be considered FALSE in the grouping calculations.
- If OFF = FALSE and ON = TRUE, the event value will be considered TRUE in the grouping calculations.
- The other two cases (OFF = ON = FALSE or OFF = ON = TRUE) are not used in the grouping calculations.

altus

4.1.6.4. Calculation of the Event Quality Attribute Summary of a Grouping

The quality of the summary event of a cluster and the qualities of the events that make up this cluster are variables of type *QUALITY*.

A variable of type *QUALITY* is a structure that has two fields: **VALIDITY** and **FLAGS**.

The **FLAGS** field is a structure that contains 16 fields of type BIT. For the quality of the summary event, each of these 16 bits is calculated as the logical or the same bits of the events that make up the grouping.

The **VALIDITY** field is an enumeration that can take one of the following four values:

- *VALIDITY_GOOD*
- *VALIDITY_QUESTIONABLE*
- *VALIDITY_INVALID*
- *VALIDITY_RESERVED* (this value is reserved and in principle, should never be observed)

When configuring the grouping, the user can select between two methods in the *"Quality Grouping Logic"* column, and this influences the calculation of the **VALIDITY** field:

- *AND*
- *OR*

Operation of the *AND* logic for the **VALIDITY** field:

- All it takes is for an event in the cluster to have textitVALIDITY_INVALID, for the summary quality to be *VALIDITY_INVALID*.
- If no event in the cluster has *VALIDITY_INVALID*, it is enough for an event in the cluster to have *VALIDITY_QUESTIONABLE* for the summary quality to be *VALIDITY_QUESTIONABLE*.
- If no event in the grouping has *VALIDITY_INVALID* or *VALIDITY_QUESTIONABLE*, the summary quality will be *VALIDITY_GOOD*. Note that *VALIDITY_RESERVED* is assumed to be *VALIDITY_GOOD*.

Operation of the *OR* logic for the **VALIDITY** field:

- All it takes is for an event in the grouping to have *VALIDITY_GOOD* for the summary quality to be *VALIDITY_GOOD*.
- If no event in the cluster has *VALIDITY_GOOD*, it is enough for an event in the cluster to have *VALIDITY_QUESTIONABLE* so that the summary quality is *VALIDITY_QUESTIONABLE*.
- If no event in the grouping has *VALIDITY_GOOD* or *VALIDITY_QUESTIONABLE*, the summary quality will be *VALIDITY_INVALID*. Note that *VALIDITY_RESERVED* is assumed to be *VALIDITY_INVALID*.

When calculating the **VALIDITY** and **FLAGS** fields of the summary event, according to the rules explained above, it is expected that the age of the component event of the group will reach the value configured as *"Maximum Delay (s)"* of the grouping, as explained in section Using the Grouping's Maximum Delay Configuration.

A summary event quality change event occurs when there is a change in the **VALIDITY** field or the **FLAGS** field of the summary event quality. The time stamp comes from the component event of the grouping that caused this change.

4.1.7. Events Routing

Event routing is a feature that allows the user to copy events that occur in one variable to another. Using the Event Routing table, the user can associate an input variable with an output variable that will receive a copy of the event with the same value, quality, and time stamp as the input variable. Both variables need to be simple and of the same type.
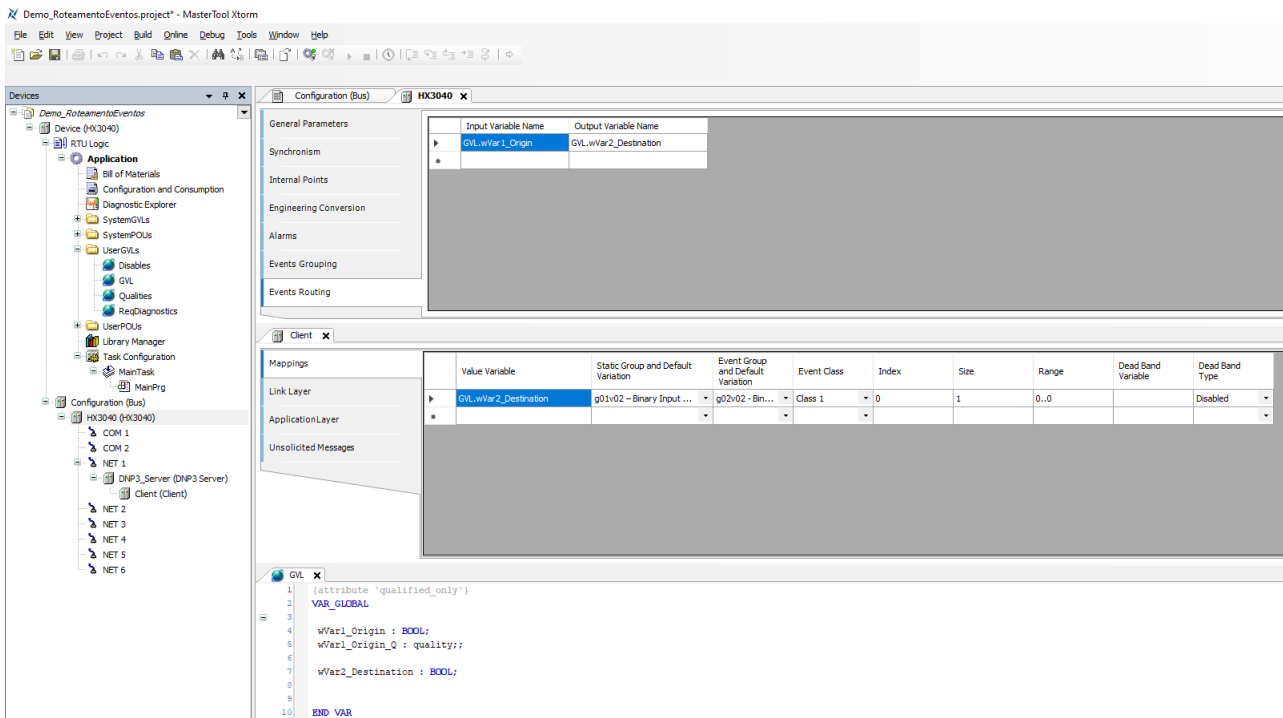
Figure 62: Events Routing

The event routing to the output variable only occurs when the input variable receives a new event. If the value or quality of the output variable is manipulated by the user or application via MasterTool Xtorm, no event will be generated for it. The same applies to the input variable unless it is an internal point.

If the output variable is mapped to a server and the server is not connected to a client, the event will be registered in the CPU's event queue.

> **ATTENTION**
>
> - An output variable cannot be an input variable of another output.
> - The input and output variables must be of the same type.
> - No event is generated if the value or quality of the output variable is changed by the application or user via MasterTool.
> - No event is generated if the value or quality of the input variable is changed by the application or user via MasterTool (except if it is an internal point).

Event Routing converts the quality to Internal format. Then, when using input/output points from the same protocol (like DNP3 or IEC104), the quality of Output variable may slightly different compared to the Input variable (according to the quality conversion tables described on this manual).

4.2. Serial Interfaces Configuration

4.2.1. COM 1

The communication interface COM 1 is composed of a DB9 female connector for RS-232C and RS-485 standard (not isolated). It allows point-to-point communication (or networking using a converter) in open protocols, Slave MODBUS RTU or Master MODBUS RTU.

The table below shows the parameters that must be configured for the application to work properly.

When using the MODBUS Master/Slave protocol, some of these parameters (such as Serial Mode, Data Bits, RX Threshold and Serial Events) are automatically adjusted by the MasterTool for the correct operation of this protocol.

altus

Configurations	Description	Default	Possibilities
Serial Type	Serial channel type configuration.	RS-232C	RS-232C or RS-485
Baud Rate	Serial communication port speed.	115200	600, 1200, 1800, 2400, 4800, 9600, 19200, 38400, 57600, 115200 bps
Parity	Set the serial port parity.	No Parity	Odd Even Parity Always One Parity Always Zero No Parity
Data Bits	Set the number of data bits in each character of the serial communication.	8	5, 6, 7 and 8
Stop Bits	Set the serial port stop bits.	1	1, 1.5 and 2
Serial Mode	Set the operation mode of the serial port.	Normal Mode	- Extended Mode: Extended operating mode of the serial communication, in which information about the received data frame is provided. - Normal Mode: Normal operating mode of the serial communication.

Table 54: COM 1 Configurations

Notes:

Extended Mode: In this operation mode of serial communication, information about the received data frame is provided. The information provided are the followings:

One byte for received data (RX_CHAR : BYTE): Stores the five, six, seven, or eight bits of received data, depending on the serial configuration.

One byte for signal errors (RX_ERROR : BYTE): Has the following format:

- Bit 0: 0 - the character in bits 0 to 7 is valid. 1 - the character in bits 0 to 7 is not valid (or may not be valid), due to the problems indicated in bits 10 to 15.
- Bit 1: Not used.
- Bit 2: Not used.
- Bit 3: UART interrupt error. The serial input has remained in logic 0 (parity always zero) for a time longer than one character (start bit + data bit + parity bit + stop bit).
- Bit 4: UART frame error. Logic 0 (parity always zero) was read when the first stop bit was expected, and should have been logic 1 (parity always one).
- Bit 5: UART parity error. The parity bit read does not agree with the parity bit calculated.
- Bit 6: UART overrun error. Data was lost during the FIFO UART reading, because new characters were received before the old ones were removed. This error will only be indicated on the first character read after the overrun error is indicated. This means that some old data has been lost.
- Bit 7: RX queue overrun error: This character was written when the RX queue was complete, overwriting unread characters.

Two bytes for the time stamp signal (RX_TIMESTAMP : WORD): Indicates the silence time, in the range 0 to 65535, using as time base 10 us. Saturates in 655.35 ms, if the silence time is higher than 65535 units. The RX_TIMESTAMP of a character measures the time from a reference which can be one of the three options below:

- In most cases, the end of the previous character.
- Serial port configuration.
- The end of the serial transmission using SERIAL_TX FB, that is, when the last character has been sent on the line.

altus

In addition to measuring the silence time before each character, RX_TIMESTAMP is also important, because it measures the silence time of the last character in the RX queue.

The silence measurement is important to correctly implement some protocols, such as MODBUS RTU. This protocol specifies an inter-frame greater than 3.5 characters and an inter-byte less than 1.5 characters.

Data Bits: The Data Bits setting of the serial interfaces limits the Stop Bits and Parity fields of the communication, that is, the number of Stop Bits and the Parity method will vary according to the number of Data Bits. The table below shows the allowed settings for the COM 1 interface.

Data Bits	Stop Bits	Parity
5	1,1.5	NO PARITY, ODD, EVEN, PARITY ALWAYS ONE, PARITY ALWAYS ZERO.
6	1,2	NO PARITY, ODD, EVEN, PARITY ALWAYS ONE, PARITY ALWAYS ZERO.
7	1,2	NO PARITY, ODD, EVEN, PARITY ALWAYS ONE, PARITY ALWAYS ZERO.
8	1,2	NO PARITY, ODD, EVEN, PARITY ALWAYS ONE, PARITY ALWAYS ZERO.

Table 55: Specific Settings

4.2.1.1. Advanced Configurations

The advanced settings are related to the control signals of the serial communication, that is, when it is necessary to use a more refined control of the transmission and reception of data.

Configurations	Description	Default	Possibilities
Advanced Port Parameters			
Modem Signals	It controls the request to transmit a command, through the RS-232C interface.	RTS Always Off	- RTS: Enabled at the beginning of the transmission and restarted, as soon as possible, after the end of the transmission. Ex: Control of an external RS-232C/RS-485 converter. - RTS Always Off: Always disabled. - RTS Always On: Always enabled. - RTS/CTS: If CTS is disabled, RTS is enabled. Then, CTS is waited for the transmission to start, and RTS is restarted, as soon as possible, at the end of the transmission. For example, controlling radio modems with the same modem signal. - Manual RTS: the user is responsible for controlling all control signals.

altus

Configurations	Description	Default	Possibilities
Advanced Port Parameters			
UART RX Threshold	The number of bytes that must be received to generate a new UART interruption. Low values make the TIMESTAMP more precise when the EXTENDED MODE is used and minimize overrun errors. However, low values can cause many interrupts that can delay the CPU.	8	1, 4, 8 e 14
Serial Events			
RX on TX	If enabled, all bytes received during transmission will be discarded instead of going to the RX queue. Used to disable full-duplex operation of the RS-232C interface.	Disabled	- Enabled: Configuration enabled. - Disabled: Configuration disabled.
RX DCD Event	If enabled it generates an external event due to the change of the DCD signal.	Enabled	- Enabled: Configuration enabled. - Disabled: Configuration disabled.
RX CTS Event	If enabled it generates an external event due to the change of the CTS signal.	Enabled	- Enabled: Configuration enabled. - Disabled: Configuration disabled.

Table 56: COM 1 Advanced Settings

Notes:

RX on TX: This advanced parameter is valid for RS-232C and RS-422 configurations.

RX DCD Event: External events, such as the COM 1 DCD signal, can only be assigned to tasks of the Custom Project Profile, described in the section Custom Profile.

RX CTS Event: External events, such as the COM 1 CTS signal, can only be assigned to tasks of the Custom Project Profile, described in the section Custom Profile.

4.2.2. COM 2

The communication interface COM 2 is composed by a DB9 female connector for RS-422C and RS-485 standards. It allows peer-to-peer or network communication in open protocols, Slave MODBUS RTU or Master MODBUS RTU.

The table below shows the parameters that must be configured for the proper functioning of the application.

When using the MODBUS Master/Slave protocol, some of these parameters (such as Serial Mode, Data Bits, RX Threshold and Serial Events) are automatically adjusted by the MasterTool for the correct operation of this protocol.

altus

Configurations	Description	Default	Possibilities
Serial Type	Serial channel type configuration.	RS-485	RS-232C or RS-485
Baud Rate	Serial communication port speed.	115200	600, 1200, 1800, 2400, 4800, 9600, 19200, 38400, 57600, 115200 bps
Parity	Set the serial port parity.	No Parity	Odd Even Parity Always One Parity Always Zero No Parity
Data Bits	Set the number of data bits in each character of the serial communication.	8	5, 6, 7 and 8
Stop Bits	Set the serial port stop bits.	1	1, 1.5 and 2
Serial Mode	Set the operation mode of the serial port.	Normal Mode	- Extended Mode: Extended operating mode of the serial communication, in which information about the received data frame is provided. (see section note COM 1). - Normal Mode: Normal operating mode of the serial communication.

Table 57: COM 2 Configurations

The Data Bits setting of the serial interfaces limits the Stop Bits and Parity communication fields, that is, the number of Stop Bits and the Parity method will vary according to the number of Data Bits. The table below shows the allowed settings for the COM 2 interface.

Data Bits	Stop Bits	Parity
5	1,1.5	NO PARITY, ODD, EVEN, PARITY ALWAYS ONE, PARITY ALWAYS ZERO.
6	1,2	NO PARITY, ODD, EVEN, PARITY ALWAYS ONE, PARITY ALWAYS ZERO.
7	1,2	NO PARITY, ODD, EVEN, PARITY ALWAYS ONE, PARITY ALWAYS ZERO.
8	1,2	NO PARITY, ODD, EVEN, PARITY ALWAYS ONE, PARITY ALWAYS ZERO.

Table 58: Specific Settings

altus

4.2.2.1. Advanced Configurations

The advanced settings are related to the control signals of the serial communication, that is, when it is necessary to use a more refined control of the transmission and reception of data.

Configurations	Description	Default	Possibilities
UART RX Threshold	The number of bytes that must be received to generate a new UART interruption. Low values make the TIMESTAMP more precise when the EXTENDED MODE is used and minimize overrun errors. However, low values can cause many interrupts that can delay the CPU.	8	1, 4, 8 and 14

Table 59: COM 2 Advanced Settings

4.3. Ethernet Interfaces Configuration

The Hadron Xtorm CPU has six local Ethernet interfaces (NET 1 ... NET 6), which can operate independently or as pairs. Each interface (or each pair) must be configured in a separate subnet.

4.3.1. Local Ethernet Interfaces

4.3.1.1. NET 1 .. NET 6

The interfaces NET 1 to NET 6 are composed of one RJ45-type connector for standard 10/100Base-TX communication. They allow peer-to-peer or network communication on supported protocols. NET 1 is always enabled, and it is through it (or the pair NET 1+2) that communication with the MasterTool Xtorm tool takes place. The interfaces NET 2 to NET6 can be enabled/disabled through configuration.

The table below describes the configuration parameters for the Ethernet interfaces:

Configurations	Description	Default	Possibilities
Enable	Enables the Ethernet interface (only for NET 2 ... NET 6).	Disabled	Disabled Enabled
IP Address	IP address of the Controller in the Ethernet bus.	192.168.15.1 (NET 1) 192.168.16.1 (NET 2) 192.168.17.1 (NET 3) 192.168.18.1 (NET 4) 192.168.19.1 (NET 5) 192.168.20.1 (NET 6)	1.0.0.1 to 223.255.255.254
Subnet Mask	Controller Subnet Mask in the Ethernet bus.	255.255.255.0	0.0.0.1 to 255.255.255.254
Gateway Address	Gateway Address of the Controller on the Ethernet bus.	192.168.15.253 (NET 1) 192.168.16.253 (NET 2) 192.168.17.253 (NET 3) 192.168.18.253 (NET 4) 192.168.19.253 (NET 5) 192.168.20.253 (NET 6)	1.0.0.1 to 223.255.255.254

Table 60: NET 1 ... NET 6 Interfaces Configurations

> **ATTENTION**
>
> It is not possible to configure two or more Ethernet interfaces of a CPU on the same Subnet, this type of configuration is blocked by the MasterTool Xtorm. Therefore, each Ethernet interface must be configured on a separate Subnet.

4.3.2. Reserved TCP/UDP Ports

The following TCP/UDP ports of the Ethernet interfaces, both local and remote, are used by CPU services (depending on availability according to table Protocols) and, therefore, are reserved and must not be used by the user.

Service	TCP	UDP
System Web Page	80	-
SNTP	-	123
SNMP	-	161
MODBUS TCP	502*	-
Mastertool	1217*	1740:1743
SQL Server	1433	-
MQTT	1883* / 8883*	-
EtherNet/IP	44818	2222
IEC 60870-5-104	2404*	-
DNP3	20000* / 20005*	-
OPC UA	4840	-
WEBVISU	8080	-
CODESYS ARTI	11740	-
PROFINET	-	34964
Portainer Docker	9000	-

Table 61: Reserved TCP/UDP ports

* Default port, but user changeable.

4.3.3. Ethernet Interfaces Advanced Configurations

The HX3040 CPU's Ethernet channels can be configured in three different modes of operation: individually, arranged in NIC Teaming pairs or in Switch mode. Possible pairs are NET 1+2, NET 3+4 and NET 5+6, with the pair configuration always stored in the odd NET. Thus, when configured as part of a pair, the configuration fields of NETs 2, 4 and 6 are disabled.

The advanced Ethernet interface settings, shown in the figure below, follow the parameters described in the following table.

altus
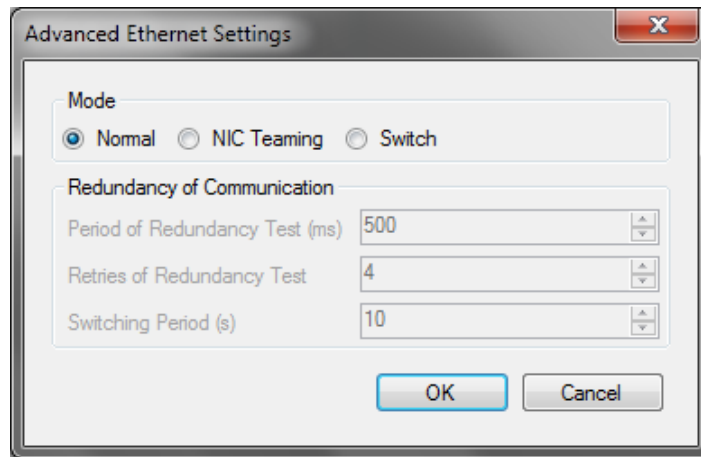
Figure 63: Ethernet Interfaces Advanced Settings Screen

Configurations	Description	Default	Possibilities
Mode	Enables the Ethernet Interface mode of operation.	Normal	Normal Nic Teaming Switch

Table 62: Advanced Ethernet Interface Settings

Normal Mode: In this mode, the interface operates as an independent Ethernet port, with no relation to the subsequent interface.

NIC Teaming Mode: In this mode, the interface forms a redundant pair with the subsequent interface, operating in an active/reserve scheme. A NIC Teaming pair has a single IP address, associated with the interface that is currently active. In this way, a client such as a SCADA or Mastertool Xtorm, connected to a server on the CPU, does not have to worry about changing the IP address if any of the ports on the pair fails. In addition, each of the interfaces that form a NIC Teaming pair has separate diagnostics, in order to facilitate debugging any faults that may arise.

When NIC Teaming Mode is selected, on the same screen, other parameters are automatically enabled and must be configured:

- Period of Redundancy Test (ms): Period of sending the communication test frame between the two NETs. Can be configured with values between 100 and 9900, default 500.
- Retries of Redundancy Test: Maximum number of times the NET that sent the frame will wait for a response. Can be set to values between 1 and 100, default 4.
- Switching Period (s): The maximum period of time that Active NET will wait for any given package. Can be configured with values between 1 and 25, default 10.

If the response time for the Redundancy Test reaches Test Period x Number of Retries and the active interface remains longer than the Switching Period without receiving any package, a switchover will occur, making the previously inactive interface active. It is important to note that there is a delay between failure detection and activation of the inactive interface due to the time required for its configuration. This delay can reach a few tens of milliseconds.

When one of the NETs is active, it will assume the configured IP address, and the inactive NET will remain with its IP Address, Subnet Mask and Gateway Address parameters blank in the CPU diagnostics.

Switch Mode: In this mode, the interface forms a pair with the subsequent interface operating as an Ethernet switch, thus allowing communication over both ports. Thus, this mode allows the "cascade" connection of several Xtorm RTUs, enabling the implementation of a ring network topology. However, to close this ring, it is mandatory that the ends of this "cascade" are connected to an external switch that supports the STP or RSTP protocols. The advantage of this architecture is the reduction in the number of ports on the external switch when compared to the star network topology. However, the main disadvantage is the network recovery time in case of a failure (convergence time), which can be up to 50 seconds using STP or up to 6 seconds using RSTP. Thus, it is important to evaluate if the system being developed can live without major problems with these network interruption times. If it is a critical system that cannot cope with such long interruptions (for example, control or protection systems with very fast response times), the star network topology should be used.

4.4. Double Points

The dual digital input and output points representation is realized through a special data type called DBP (defined in the LibDataTypes library). This type basically consists of a structure of two BOOL elements called OFF and ON (equivalent to TRIP and CLOSE respectively). Variables of this type can be associated with Input and Output modules that have event and pulse command support (HX1100, HX2200, HX2300 and HX2320 for example), where each of the OFF and ON elements must be mapped individually to each physical input/output of the module. In the case of input modules, the %I memory is updated normally even for points that are configured as DBP. In the case of output modules, the points configured as DBP no longer consider the value present in the %Q memory.

More information about the configuration of double points in the input and output modules can be found in their respective Technical Characteristics documents.

4.5. Protocols Configuration

Hadron Xtorm provides several communication protocols, including MODBUS (symbolic only), DNP3, IEC 60870-5-104, and others. The following table describes the configuration limits:

	HX3040
Mapped Points	20000
Mappings (Per Instance / Total)	5000 / 5000
Requests	512
NETs - Client or Server Instances (Per NET / Total)	4 / 12
COM (n) - Master or Slave Instances	1
DNP3 connections	100
IEC 60870-5-104 connections	100
Control Centers	5

Table 63: Protocol limits per CPU

Notes:

Mapped points: This is the maximum number of mapped points that the CPU supports. Remember that each mapping supports 1 or more mapped points, depending on the data size, when used with variables of type ARRAY.

Mappings: A "mapping" is a relationship between an internal application variable and an application protocol object. This field informs the maximum amount of mappings supported by the CPU. It corresponds to the sum of all mappings performed on communication protocol instances and their respective devices.

Requests: The sum of communication protocols' requests, declared on the devices, cannot exceed the maximum amount of requests supported by the CPU.

NETs - Client or Server Instances: This field defines the maximum number of protocol instances for each Ethernet interface, as well as the maximum total number distributed among all Ethernet interfaces in the system.

COM (n) - Master or Slave instances: Due to its characteristics, each serial interface supports only one instance of communication protocol. Examples of supported instances for Serial interfaces are: MODBUS RTU Master and MODBUS RTU Slave.

DNP3 connections: This is the sum of the maximum number of DNP3 Client and Server connections of one HX3040 CPU, not allowed to exceed the total number of connections informed in the table for a correct operation of the application.

IEC 60870-5-104 Connections: This is the sum of the maximum number of IEC 60870-5-104 Client and Server connections of one HX3040 CPU, not allowed to exceed the total number of connections informed in the table for a correct operation of the application.

Control Centers: "Control Center" is every Client device connected to the CPU via DNP3, IEC 60870-5-104 or MMS (IEC 61850) protocols. This field informs the maximum number of Client Devices, of control center type, supported by the CPU. It corresponds to the sum of all client devices of the DNP3, IEC 60870-5-104 and MMS Server communication protocol instances (it does not include the Masters and Clients of the MODBUS RTU Slave and MODBUS Server protocols).

The following table shows the maximum configuration and operation limits for the communication protocols of the Serial and Ethernet interfaces.

altus

	MODBUS RTU Master	MODBUS RTU Slave	MODBUS Client	MODBUS Server	DNP3 Client	DNP3 Server	IEC 60870-5-104 Client	IEC 60870-5-104 Server
Devices per instance	64	1[1]	64	64[2]	32	5[3]	32	5[4]
Requests per instance	128	NA	128	NA	128	NA	NA	NA
Requests per device	32	NA	32	NA	32	NA	NA	NA
Simultaneous requests per instance	1	1	128	64	32	5	160	5
Simultaneous requests per device	1	1	8	64	1	1	5	1

Table 64: Limits of Communication Protocols

Notes:

Devices per instance:

- **Master or Client Protocols:** Number of Slave or Server devices supported by each Master or Client protocol instance.
- **MODBUS RTU Slave protocol:** The limit [1] informed refers to the serial interfaces, which do not allow a Slave to establish communication, simultaneously, through the same serial interface, with more than one Master device. It is not necessary, and it is not possible, to declare or configure the Master device under the Slave MODBUS RTU protocol instance. The Master device will have access to all mappings made directly in the Slave MODBUS RTU protocol instance.
- **MODBUS Server protocol:** The limit [2] informed refers to the Ethernet interfaces, which limit the number of connections that can be established with other devices through the same Ethernet interface. It is not necessary, and is not possible, to declare or configure Client devices under the MODBUS Server protocol instance. All Client devices will have access to all mappings made directly in the MODBUS Server protocol instance.
- **DNP3 Server protocol:** The number of client devices, control center type, supported by each DNP3 Server protocol instance. The reported [3] limit can be lower depending on the total CPU limits (see Table 63).
- **IEC 60870-5-104 Server protocol:** Number of Client devices, control center type, supported by each IEC 60870-5-104 Server protocol instance. The reported [4] limit can be lower depending on the total CPU limits (see Table 63).

Requests per instance: The sum of the requests configured on each of the devices in a Master or Client instance cannot exceed this limit.

Requests per device: Number of requests, such as reading or writing holding registers, that can be configured for each of the devices (Slaves or Servers) of the Master or Client protocol instances. For the DNP3 Client protocol, this limit refers to custom requests, configured via MasterTool. This parameter is not applicable (NA) to the Slave or Server protocol instances, as well as to the IEC 61870-5-104 Client protocol, whose custom requests must be declared and managed by the user application via Function Blocks.

Simultaneous requests per instance: Number of requests that can be transmitted simultaneously by each Client protocol instance, or that can be received simultaneously by each Server protocol instance. MODBUS RTU protocol instances, Master or Slave, do not support simultaneous requests.

Simultaneous requests per device: Number of requests that can be transmitted simultaneously to each MODBUS Server device, or that can be received simultaneously from each MODBUS Client device. MODBUS RTU devices, Master or Slave, as well as DNP3 devices, Client or Server, and IEC 60870-5-104 Server, do not support simultaneous requests. IEC 60870-5-104 Client devices support the sending of simultaneous requests, provided they are for different sectors.

> **ATTENTION**
>
> Simultaneous requests for the same variable associated with communication points that support the SBO (Select Before Operate) operating mode are not supported, even if received by different devices. Once the selection and/or operation of a point has been initiated by a given device, it must be completed before this point can be commanded by another device.

altus

4.5.1. Protocol Behavior x CPU State

The table below shows in detail each behavior assumed for each communication protocol existing in the HX3040 CPU in all its operating states, a legend specifying further information is presented in the table.

Protocol	Type	CPU Operational State					
		STOP			RUN		
		After download before application starts	After the application goes into STOP	After an exception	Non-Redundant or Active	Redundant in Standby	After a break-point in MainPrg
MODBUS	Slave/Server	✗ / ✔	✗ / ✔	✗ / ✔	✔	✗	✔
	Master/Client	✗	✗	✗	✔	✗	✗
DNP3	Outstation/Server	✗	✗	✗	✔	✗	✔
	Master/Client	✗	✗	✗	✔	✗	✔
IEC 60870-5-104	Server	✗	✗	✗	✔	✗	✔
	Client	✗	✗	✗	✔	✗	✔
MMS	Server	✗	✗	✗	✔	✗	✔
GOOSE	Publisher	✗	✗	✗	✔	✗	✔
	Subscriber	✗	✗	✗	✔	✗	✔
SNTP	Client	✔	✔	✔	✔	✔	✔
PTP	Slave	✔	✔	✔	✔	✔	✔
HTTP	Server	✔	✔	✔	✔	✔	✔
IRIG-B	Transmitter	✔	✔	✔	✔	✔	✔
	Receiver	✔	✔	✔	✔	✔	✔
PROFINET	Controller	✗	✗	✗	✔	NA	✔

Table 65: Behavior of Protocols x CPU States

Notes:

Symbol ✔ : The protocol remains active and operating normally.

Symbol ✗: The protocol is disabled.

Symbol ✗ / ✔ : To keep the protocol communicating when the CPU isn't in RUN, it's need to check the option "Keep the communication running on CPU stop".

4.5.2. CPU Event Queue

The CPU has a FIFO (First In, First Out) event queue used to temporarily store the events related to the communication points until they are transferred to their final destination.

All communication point events generated in the RTU are directed and stored in the CPU queue. This queue has the following characteristics:

- Size: 4500 events.
- Overflow policy: keeps most recent events.

This event queue is stored in a retentive (non-volatile) memory area exclusively for this purpose, and does not occupy the area of retentive/persistent symbolic variables. In this way, the events present in the CPU queue and that have not yet been transmitted to the control center will not be lost in an eventual shutdown of the RTU.

The queue is also redundant, that is, it is synchronized cycle-by-cycle between the two CPUs when using CPU redundancy. More information can be found in the specific section about CPU redundancy.

The input and output of events in this queue follows a producer/consumer concept. Producers are those system elements capable of generating events, adding events to the CPU queue, while consumers are the system elements that receive and use

altus

these events, removing events from the CPU queue. The figure below describes this operation, including an example of some consumers and producers of events:
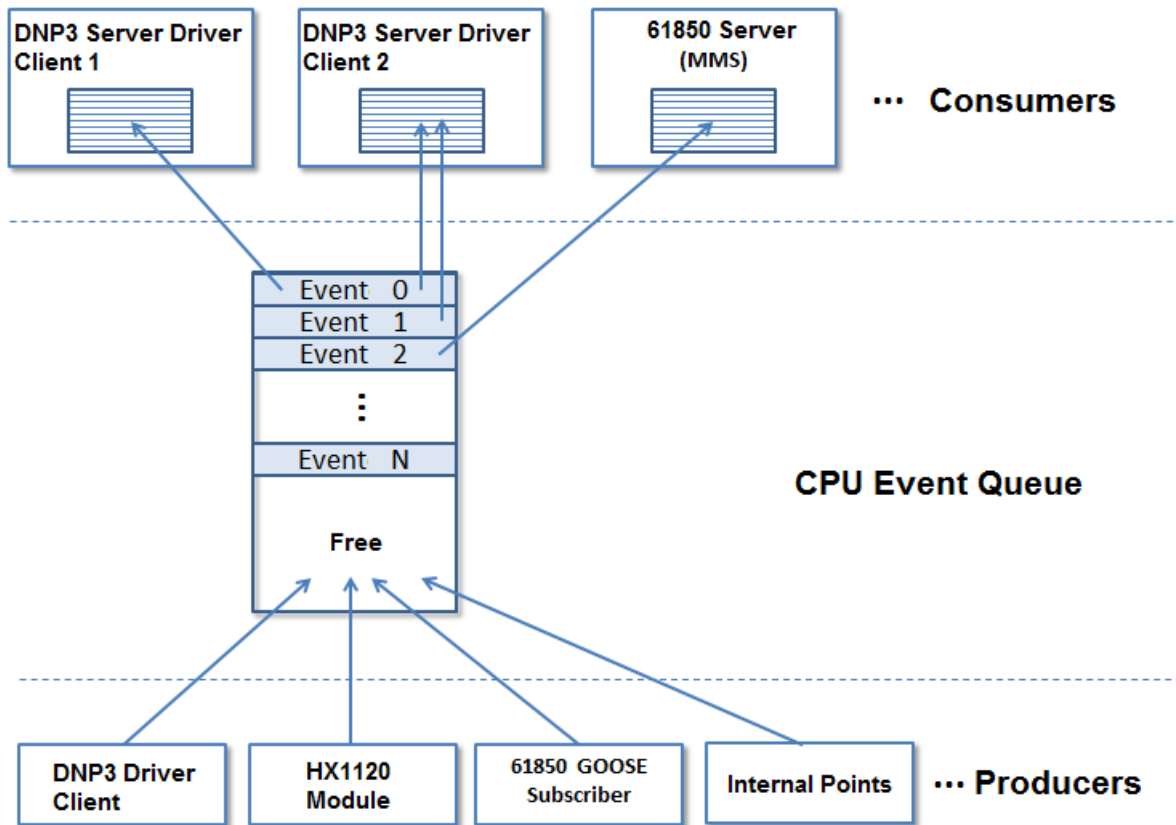
Figure 64: CPU Events Queue

4.5.2.1. Consumers

Consumers are typically communication drivers that will communicate with a SCADA or HMI. After being stored in the CPU queue, consumers receive events relative to communication points mapped in their configuration. These events are then stored in a consumer's own event queue, whose size and operation is described in the driver-specific section.

4.5.2.2. Event Queue Working Principles

Once stored in the CPU queue, each event is transmitted to the consumer that has this communication point in its database. In the figure above, Event 0 refers to a communication point mapped to 2 DNP3 control centers (Client 1 and 2). Event 1, on the other hand, refers to a communication point mapped to only one DNP3 control center (Client 2). In turn, Event 2 refers to a communication point mapped to an IEC 61850 control center.

Events remain stored in the CPU queue until all its consumers confirm their reception. The criterion used to confirm receipt is specific to each consumer. In the specific case of DNP3 Server, confirmation only occurs when the control center (SCADA) confirms receipt of the message (which provides good security against loss of events in case of network failure). For other types of consumers (as is the case with Buffered Reports in IEC 61850 Server), confirmation occurs when the event has been transferred to the driver's internal queue. The occupancy of the event queue of the CPU can be monitored through the diagnostics available in the diagnostics area of the CPU through the variable DG_HX3040.tDetailed.Events.*, which also provides information about queue overflow, among others.

4.5.2.2.1. Overflow Signalling

Overflow signaling to the CPU event queue occurs in two situations:

- If the event queue of the CPU is already partially occupied and the available space is not enough to store the new events occurring in the execution cycle.
- If the CPU aborted event generation (because more events occurred in a single execution cycle than the total size of its event queue).

Overflow signaling to the consumer event queue, on the other hand, occurs in two situations:

- When the consumer's event queue has no more space to store new events.
- If the CPU aborted event generation (because more events occurred in a single execution cycle than the total size of its event queue).

4.5.2.3. Producers

Producers are typically communication drivers or internal RTU elements that are capable of generating events. The figure above shows some examples:

- HX1100 and HX1120 module: Events are generated and stored internally in the module, and are then read by the RTU's local bus and inserted in the CPU queue.
- DNP3 Client Driver: Events are generated and stored in Outstations, and then are read by the driver and entered in the CPU queue.
- IEC 60870-5-104 Client Driver: Events are generated and stored in the controlled stations, and then they are read by the driver and inserted in the CPU queue.
- IEC 61850 GOOSE Subscriber Driver: Events are generated at the moment a GOOSE message is received (provided the received data causes a value change in some variable), and then are inserted in the CPU queue.
- Internal Points: This is an internal element in the CPU firmware, which performs event detection at each execution cycle (MainTask) for those communication points that do not have a defined origin and then inserts the events in the CPU queue. The maximum number of events that can be detected at each MainTask cycle is equal to the size of the CPU event queue. If more events than this are generated in a single MainTask cycle, the excess events will be lost.
- MODBUS Driver (Client/Server/Master/Slave): Value changes in variables caused by MODBUS reads/writes are detected at each MainTask cycle and then events are inserted in the CPU queue. In the case of Client/Master drivers, quality events are also generated when there is a communication failure with the Slave device.

Note:

HX1100 and HX1120: The HX1100 and HX1120 modules internally store the events in a structure that can contain from 1 to 32 events. When only 1 input changes state, this structure will store only 1 event. Now, when all 32 inputs change state simultaneously, this structure will store 32 events. The module has an internal queue that can store up to 420 structures (that is, from 420 to 13440 events). The CPU performs, through the local bus, the reading of 38 structures every 3 cycles of RTU execution (MainTask).

> **ATTENTION**
>
> Using the AT directive to assign a symbolic variable to the %I channels of the HX1100 and HX1120 modules will result in the loss of the precision functionality of the event timestamp (it will be stamped in the CPU cycle, as a standard internal point). In this case, it is necessary to use one of the two methods offered by the "Bus I/O Mapping" tab of the module:
> - Map to an existing variable;
> - Create a new variable;

4.5.3. Interception of Commands from Control Center

The Xtorm RTU has a function block that allows selection and operation commands for output points received by server drivers (DNP3, IEC 60870-5-104, IEC 61850, etc...) to be handled by the user logic. This feature allows the implementation of interlocks, as well as the manipulation of the command data received in the user logic or even the redirection of the command to different IEDs.

Command interception is implemented by the function block *CommandReceiver*. The input and output parameters are described in the following tables:

Parameter	Type	Description
bExec	BOOL	When TRUE, executes the command interception.
bDone	BOOL	Indicates that the command output data has already been processed, releasing the function block to receive a new command.
dwVariableAddr	DWORD	Address of the variable that will receive the command.
eCommandResult	ENUM	User defined input action from the list below: SUCCESS(0); NOT_SUPPORTED(1); BLOCKED_BY_SWITCHING_HIERARCHY(2); SELECT_FAILED(3); INVALID_POSITION(4); POSITION_REACHED(5) PARAMETER_CHANGE_IN_EXECUTION(6); STEP_LIMIT(7); BLOCKED_BY_MODE(8); BLOCKED_BY_PROCESS(9); BLOCKED_BY_INTERLOCKING(10); BLOCKED_BY_SYNNCHROCHECK(11); COMMAND_ALREADY_IN_EXECUTION(12); BLOCKED_BY_HEALTH (13); ONE_OF_N_CONTROL(14); ABORTION_BY_CANCEL(15); TIME_LIMIT_OVER(16); ABORTION_BY_TRIP(17); OBJECT_NOT_SELECTED(18); OBJECT_ALREADY_SELECTED(19); NO_ACCESS_AUTHORITY(20); ENDED_WITH_OVERSHOOT(21); ABORTION_DUE_TO_DEVIATION(22); ABORTION_BY_COMMUNICATION_LOSS(23); BLOCKED_BY_COMAND(24); NONE(25); INCONSISTENT_PARAMETERS(26); LOCKED_BY_OTHER_CLIENT(27); HARDWARE_ERROR(28); UNKNOWN(29);
dwTimeout	DWORD	User defined time-out is an input parameter for the function.

Table 66: CommandReceiver Function Block Input Parameters

Notes:

bExec: When FALSE, the command just stops being intercepted for the user application, but the command continues to be processed normally by the server.

bDone: After the command is intercepted, the user will be responsible for handling it. At the end of the handling, this input must be turned on so that a new command can be received. If this input is not turned on before the time set in the **dwTimeout** input has passed, the server will send a negative acknowledgement for the command in the protocol. As long as the *bDone* input is not turned on, no further commands can be received for the variable specified by the *dwVariableAddr* input. If other commands are received for this variable, the server will immediately send a negative acknowledgement for the command in the protocol.

eCommandResult: Result of the command handling, intercepted by the user. The result returned to the Client that sent the command, which must be assigned together with the *bDone* input, is converted to the format of the protocol from which

altus

the command was received. When intercepting the IEC 60870-5-104 protocol, any return other than *SUCCESS* results in a negative acknowledgment in the protocol.

> **ATTENTION**
>
> It is not recommended that commands for the same variable be intercepted simultaneously by two or more CommandReceiver function blocks. Only one of the function blocks will correctly intercept the command, but may suffer undesired interference from the other function blocks if addressed to the same variable.

Parameter	Type	Description
bCommandAvailable	BOOL	Indicates that a command has been intercepted, and that the data are available to be processed.
sCommand	STRUCT	This structure stores the data of the received command, which is composed by the following fields: eCommand sSelectParameters sOperateParameters *The description of each field is described in this section
eStatus	ENUM	Output action of the function based on the result obtained, according to the list below: OK_SUCCESS(0) ERROR_FAILED(1)

Table 67: CommandReceiver Function Block Output Parameters

Notes:

eStatus: Returns of the register process of the interception of a command for a communication point. When the interception is successfully registered OK_SUCCESS is returned, otherwise ERROR_FAILED. In case of failure to register the interceptor, commands for the given point are not intercepted by the respective function block.

The supported commands are described in the table below:

Parameter	Type	Options
sCommand	ENUM	NO_COMMAND(0) SELECT(1) OPERATE(2)

Table 68: CommandReceiver Function Block Supported Commands

The parameters that compose the *sSelectParameters*, *sOperateParameters* and *sCancelParameters* structures are described in the following tables:

Parameter	Type	Description
sSelectConfig	STRUCT	Configuration of the received selection command. The parameters of this structure are described in the table 70
sValue	STRUCT	Value received by a selection command with value. The parameters of this structure are described in the table 73

Table 69: sSelectParameters

altus

Parameter	Type	Description
bSelectWithValue	BOOL	When true it indicates the reception of a selection command with value.

Table 70: sSelectConfig Parameters

Parameter	Type	Description
sSelectConfig	STRUCT	Configuration of the received selection command. The parameters of this structure are described in the table 72
sValue	STRUCT	Value referring to the received operation command. The parameters of this structure are described in the table 73

Table 71: sOperateParameters Parameters

Parameter	Type	Description
bDirectOperate	BOOL	When true indicates the reception of an operation command without select.
bNoAcknowledgement	BOOL	When true indicates the reception of a command that does not require an acknowledge of receipt.
bTimedOperate	BOOL	When true indicates that a time-activated operation command was received.
liOperateTime	LINT	Programming of the moment when the command should be executed. This field is valid only when bTimedOperate is true.
bTest	BOOL	When true indicates that the received command was sent for testing purposes only, so it should not be executed.

Table 72: sOperateConfig Parameters

Parameter	Type	Description
eParamType	ENUM	Informs the received command type: NO_COMMAND(0) SINGLE_POINT_COMMAND(1) DOUBLE_POINT_COMMAND(2) INTEGER_STATUS_COMMAND(3) ENUMERATED_STATUS_COMMAND(4) ANALOGUE_VALUE_COMMAND(5)
sSinglePoint	STRUCT	When a command is received, depending on the type of the received command, defined by eParamType, the corresponding data structure is filled in. The parameters of these structures are described in the table 74 to table 78
sDoublePoint	STRUCT	
sIntegerStatus	STRUCT	
sEnumeratedStatus	STRUCT	
sAnalogueValue	STRUCT	

Table 73: sValue Parameters

altus

Parameter	Type	Description
bValue	BOOL	Operation point value.
sPulseConfig	STRUCT	The configuration parameters of a pulsed command are stored in this structure. The parameters of this structure are described in the table 79

Table 74: sSinglePoint Parameters

Parameter	Type	Description
bValue	BOOL	Operation point value.
sPulseConfig	STRUCT	The configuration parameters of a pulsed command are stored in this structure. The parameters of this structure are described in the table 79

Table 75: sDoublePoint Parameters

Parameter	Type	Description
diValue	DINT	Operation point value.

Table 76: sIntegerStatus Parameters

Parameter	Type	Description
dwValue	DWORD	Operation point value.

Table 77: sEnumeratedStatus Parameters

Parameter	Type	Description
eType	ENUM	Informs the data type of the analog value received: INTEGER (0) FLOAT (1)
diValue	DINT	Operation point value, in integer format.
fValue	REAL	Operation point value, in float format.

Table 78: sAnalogueValue Parameters

Parameter	Type	Description
bPulseCommand	BOOL	When true indicates that the received command is pulsed.
dwOnDuration	DWORD	This is the duration, in milliseconds, that the digital output must remain on.
dwOffDuration	DWORD	This is the duration, in milliseconds, that the digital output must remain off.
dwPulseCount	DWORD	Number of times the command should be performed.

Table 79: sPulseConfig Parameters

In order to intercept commands for a given point, first load in the parameter *dwVariableAddr* the address of the variable corresponding to the point where you want to intercept commands and then turn on the input parameter *bExec*. Once the

command is received, the function block reports that a command has been intercepted via the parameter *bCommandAvailable*. The intercepted command information is then filled into the output parameters *sCommand* and *eStatus* according to the type of command that was received. This operation depends only on the type of command received, regardless of the data type of the variable for which the command is being intercepted. The interception is finished and then the function block can be released to intercept a new command when set to true in the *bDone* parameter. The result of the command processing must be indicated in *andCommandResult*.

Below is an example of a command interceptor application, using the ST language, which sends a command received by the Server protocol (DNP3 or IEC 60870-5-104), through the *dbpDoublePointServer* variable, to a double point of a Hadron Xtorm Series digital output, where the *dbpDoublePointIO* variable is mapped.

For this example code the command *C_DC_NA* can be used in *Execute Only* mode and with the qualifier *Short pulse* of the IEC 60870-5-104 Server protocol, or the function *Direct Operate* of the type *Control Relay Output Block* and with control code *Trip* or *Close* of the DNP3 Server protocol.

The example also uses the *PulsedCommand* function from the LibRtuStandard library, detailed in the Xtorm Series digital output modules' technical characteristics document (see the HX2320 module CT).

```
VAR
  CRReceive: CommandReceiver; // Interceptor Instance
  CCommand: COMMAND_T;       // Structure with the data of the command
  CRResult: COMMAND_RESULT; // Result of the intercepted command
  byResult: BYTE;           // Result of the Trip/Close function
  byPulseTimeON: BYTE;      // Time pulse on
  byPulseTimeOFF: BYTE;     // Time pulse off
  dbpDoublePointIO: DBP;     // Variable mapped in the output module
  dbpDoublePointServer: DBP;  // Variable mapped in server driver
END_VAR

CRReceive.dwVariableAddr:= ADR(dbpDoublePointServer);
CRReceive.bExec:= TRUE;
CRReceive.dwTimeout:= 1000;


// If a command is captured:
IF CRReceive.bCommandAvailable THEN
  // Saves the command data
  CCommand:= CRReceive.sCommand;
  // Sending the command to the card by the PulsedCommand function
  IF CCommand.sOperateParameters.sValue.sDoublePoint.bValue THEN
    // Captures the configured value for the pulse time
    byPulseTimeON:= DWORD_TO_BYTE(CCommand.sOperateParameters.sValue.
   sDoublePoint. sPulseConfig.dwOnDuration / 10);
    // Performs the PulsedCommand function for the output module
    byResult:= PulsedCommand(bCommandType:= 101, bRackNumber:= 0, bSlotNumber:=
   4, bPairIndex:= 0, bPulseTime:= byPulseTimeON);
  ELSE
    // Capture the set value for the pulse time
    byPulseTimeOFF:= DWORD_TO_BYTE(CCommand.sOperateParameters.sValue.
   sDoublePoint. sPulseConfig.dwOffDuration / 10);
    // Performs the PulsedCommand function for the output module
    byResult:= PulsedCommand(bCommandType:= 102, bRackNumber:= 0, bSlotNumber:=
   4, bPairIndex:= 0, bPulseTime:= byPulseTimeOFF);
  END_IF
  // Handles the result of the pulsed command function and generates the
    response for the intercepted command
  CASE byResult OF
    1:  // Invalid command type
       CRResult:= COMMAND_RESULT.NOT_SUPPORTED;
       CRReceive.eCommandResult:= CRResult;
       CRReceive.bDone:= TRUE;
```

```
      2, 3: // Invalid input parameters
        CRResult:= COMMAND_RESULT.INCONSISTENT_PARAMETERS;
        CRReceive.eCommandResult:= CRResult;
        CRReceive.bDone:= TRUE;
      4:  // Module did not respond to the command
        CRResult:= COMMAND_RESULT.TIME_LIMIT_OVER;
        CRReceive.eCommandResult:= CRResult;
        CRReceive.bDone:= TRUE;
      6:  // Another command was sent to this point and is running
        CRResult:= COMMAND_RESULT.BLOCKED_BY_COMAND;
        CRReceive.eCommandResult:= CRResult;
        CRReceive.bDone:= TRUE;
      7, 5: // Command completed successfully
        CRResult:= COMMAND_RESULT.SUCCESS;
        CRReceive.eCommandResult:= CRResult;
        CRReceive.bDone:= TRUE;
    END_CASE
  END_IF

  CRReceive();

  IF CRReceive.bDone THEN
    CRReceive.bDone:= FALSE;
  END_IF
```

4.5.4. MODBUS – Data Types

The table below shows the variable type supported by the HX3040 for each of the protocol data types, applicable for all MODBUS devices.

Relation Type	Accepted Variables Type
Coil	BOOL
Input Status	BOOL
Holding Registers	INT, UINT, WORD, REAL, DINT, UDINT, DWORD, LREAL, LINT, ULINT ou LWORD
Input Registers	INT, UINT, WORD, REAL, DINT, UDINT, DWORD, LREAL, LINT, ULINT ou LWORD

Table 80: Declaration of Variables for MODBUS

Note:

For Holding Register mask writing only the types INT, UINT or WORD can be used. Variables of type REAL, DINT, UDINT and WORD have size 2, while variables of type LREAL, LINT, ULINT and LWORD have size 4. The other variables have size 1. In the case of arrays mappings, the number of elements of the arrays will be multiplied by the size of the data type previously entered.

4.5.5. MODBUS RTU Master

This protocol is available for the Hadron Xtorm Series CPU on its serial channels. By selecting this option in MasterTool Xtorm, the CPU becomes the MODBUS communication Master, allowing access to other devices with the same protocol, when it is in Run mode.

To configure this protocol, the following steps must be performed:

1. Add the MODBUS RTU Master protocol instance to serial channels COM 1 or COM 2 (or both, in case of two communication networks).
2. To perform this procedure, see the section Serial Interfaces Configuration.
3. Configure the serial interface, choosing the communication speed, the behavior of the RTS/CTS signals, the parity, channel stop bits, among others by double-clicking on the COM 1 or COM 2 serial channel.
4. Configure the general parameters of the MODBUS Master protocol, such as: send delay time and minimum interframe.
5. Add and configure devices by setting the Slave address, communication time-out, and number of communication retries.
6. Add and configure MODBUS mappings by specifying the variable name, data type, data start address, data size, and the variable that will receive the quality data.
7. Add and configure MODBUS requests, specifying the desired function, the scan time of the request (polling), the starting address (Read/Write), the data size (Read/Write), the variable that will receive the quality data, and the variable responsible for disabling the request.

The descriptions of each configuration are described below in this section.

4.5.5.1. MODBUS Master Protocol Configuration by Symbolic Mapping

4.5.5.1.1. MODBUS Master Protocol General Parameters – Symbolic Mapping Configuration

The general parameters, found on the MODBUS protocol initial screen (figure below), are defined as:
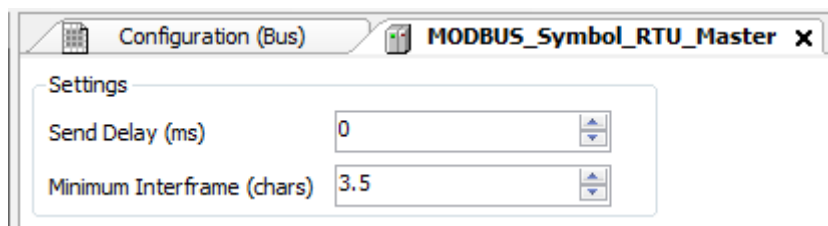
Figure 65: MODBUS RTU Master Configuration Screen

Configuration	Description	Default	Options
Send Delay (ms)	Delay for the answer transmission.	0	0 to 65535
Minimum Interframe (chars)	Minimum silence time between different frames.	3.5	3.5 to 100.0

Table 81: MODBUS RTU Master General Configurations

Notes:

Send Delay: The answer to a MODBUS protocol may cause problems in certain moments, as in the RS-485 interface or other half-duplex. Sometimes there is a delay between the slave answer time and the physical line silence (slave delay to put RTS in zero and put the RS-485 in high impedance state). To solve this problem, the master can wait the determined time in this field before sending the new request. Otherwise, the first bytes transmitted by the master could be lost.

Minimum Interframe: The MODBUS standard defines this time as 3.5 characters, but this parameter is configurable in order to attend the devices which do not follow the standard.

The MODBUS protocol diagnostics and commands configured, either by symbolic mapping or direct representation, are stored in *T_DIAG_MODBUS_RTU_MASTER_1* variables. For the direct representation mapping, they are also in 4 bytes and 8 words which are described in table below (where "n" is the configured value in the *%Q Start Address of Diagnostics Area* field):

Direct Repre-sentation Variable	Diagnostic Variable T_DIAG_MODBUS _RTU_MASTER_1.*	Size	Description	altus
Diagnostics Bits:				
%QX(n).0	tDiag. bRunning	BIT	The master is running.	
%QX(n).1	tDiag. bNotRunning	BIT	The master is not running (see bit: bInterruptedByCommand).	
%QX(n).2	tDiag. bInterruptedByCommand	BIT	The bit bNotRunning was enabled as the master was interrupted by the user through command bits.	
%QX(n).3	tDiag. bConfigFailure	BIT	Discontinued diagnosis.	
%QX(n).4	tDiag. bRXFailure	BIT	Discontinued diagnosis.	
%QX(n).5	tDiag. bTXFailure	BIT	Discontinued diagnosis.	
%QX(n).6	tDiag. bModuleFailure	BIT	Indicates if there is failure in the module or the module is not present.	
%QX(n).7	tDiag. bDiag_7_reserved	BIT	Reserved	
Error Codes:				
%QB(n+1)	eErrorCode	SERIAL_STATUS (BYTE)	0: there are no errors 1: invalid serial port 2: invalid serial port mode 3: invalid baud rate 4: invalid data bits 5: invalid parity 6: invalid stop bits 7: invalid modem signal parameter 8: invalid UART RX Threshold parameter 9: invalid time-out parameter 10: busy serial port 11: UART hardware error 12: remote hardware error 20: invalid transmission buffer size 21: invalid signal modem method 22: CTS time-out = true 23: CTS time-out = false 24: transmission time-out error 30: invalid reception buffer size 31: reception time-out error 32: flow control configured differently from manual 33: invalid flow control for the configured serial port 34: data reception not allowed in normal mode 35: data reception not allowed in extended mode 36: DCD interruption not allowed	

Direct Representation Variable	Diagnostic Variable T_DIAG_MODBUS _RTU_MASTER_1.*	Size	Description
			37: CTS interruption not allowed
			38: DSR interruption not allowed
			39: serial port not configured
			50: internal error in the serial port
Command bits, automatically initialized:			
%QX(n+2).0	tCommand. bStop	BIT	Stop master.
%QX(n+2).1	tCommand. bRestart	BIT	Restart master.
%QX(n+2).2	tCommand. bResetCounter	BIT	Restart diagnostics statistics (counters).
%QX(n+2).3	tCommand. bDiag_19_reserved	BIT	Reserved
%QX(n+2).4	tCommand. bDiag_20_reserved	BIT	Reserved
%QX(n+2).5	tCommand. bDiag_21_reserved	BIT	Reserved
%QX(n+2).6	tCommand. bDiag_22_reserved	BIT	Reserved
%QX(n+2).7	tCommand. bDiag_23_reserved	BIT	Reserved
%QB(n+3)	byDiag_3_reserved	BYTE	Reserved
Communication Statistics:			
%QW(n+4)	tStat. wTXRequests	WORD	Counter of request transmitted by the master (0 to 65535).
%QW(n+6)	tStat. wRXNormalResponses	WORD	Counter of normal responses received by the master (0 to 65535).
%QW(n+8)	tStat. wRXExceptionResponses	WORD	Counter of responses with exception codes received by the master (0 to 65535).
%QW(n+10)	tStat. wRXIllegalResponses	WORD	Counter of illegal responses received by master – invalid syntax, not enough received bytes, invalid CRC – (0 to 65535).
%QW(n+12)	tStat. wRXOverrunErrors	WORD	Counter of overrun errors during reception - UART FIFO or RX line – (0 to 65535).
%QW(n+14)	tStat. wRXIncompleteFrames	WORD	Counter of answers with construction errors, parity or failure during reception (0 to 65535).
%QW(n+16)	tStat. wCTSTimeOutErrors	WORD	Counter of CTS time-out error, using RTS/CTS handshake, during transmission (0 to 65535).
%QW(n+18)	tStat. wDiag_18_Reserved	WORD	Reserved

Table 82: MODBUS RTU Master Diagnostics

Note:

Counters: All MODBUS RTU Master diagnostics counters return to zero when the limit value 65535 is exceeded.

4.5.5.1.2. *Devices Configuration – Symbolic Mapping configuration*

The devices configuration, shown on figure below, follows the following parameters:
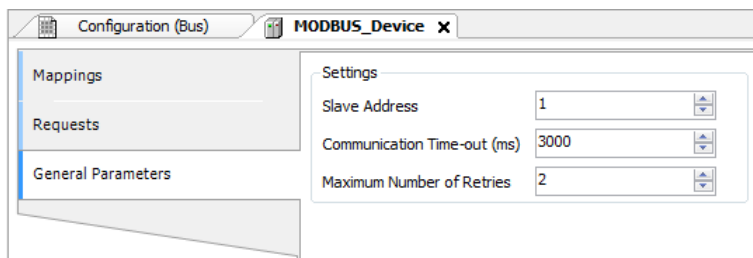
Figure 66: Device General Parameters Settings

Configuration	Description	Default	Options
Slave Address	MODBUS slave address	1	0 to 255
Communication Time-out (ms)	Defines the application level time-out	3000	10 to 65535
Maximum Number of Retries	Defines the numbers of retries before reporting a communication error	2	0 to 9

Table 83: Device Configurations

Notes:

Slave Address: According to the MODBUS standard, the valid slave addresses are from 0 to 247, where the addresses from 248 to 255 are reserved. When the master sends a writing command with the address configured as zero, it is making broadcast requests in the network.

Communication Time-out: The communication time-out is the time that the master waits for a response from the slave to the request. For a MODBUS RTU master device it must be taken into account at least the following system variables: the time it takes the slave to transmit the frame (according to the baud rate), the time the slave takes to process the request and the response sending delay if configured in the slave. It is recommended that the time-out is equal to or greater than the time to transmit the frame plus the delay of sending the response and twice the processing time of the request. For more information, see Communication Performance section.

Maximum number of retries: Sets the number of retries before reporting a communication error. For example, if the slave does not respond to a request and the master is set to send three retries, the error counter number is incremented by one unit when the execution of these three retries. After the increase of the communication error trying to process restarts and if the number of retries is reached again, new error will increment the counter.

4.5.5.1.3. *Mappings Configuration – Symbolic Mapping Settings*

The MODBUS relations configuration, showed on figure below, follows the parameters described on table below:
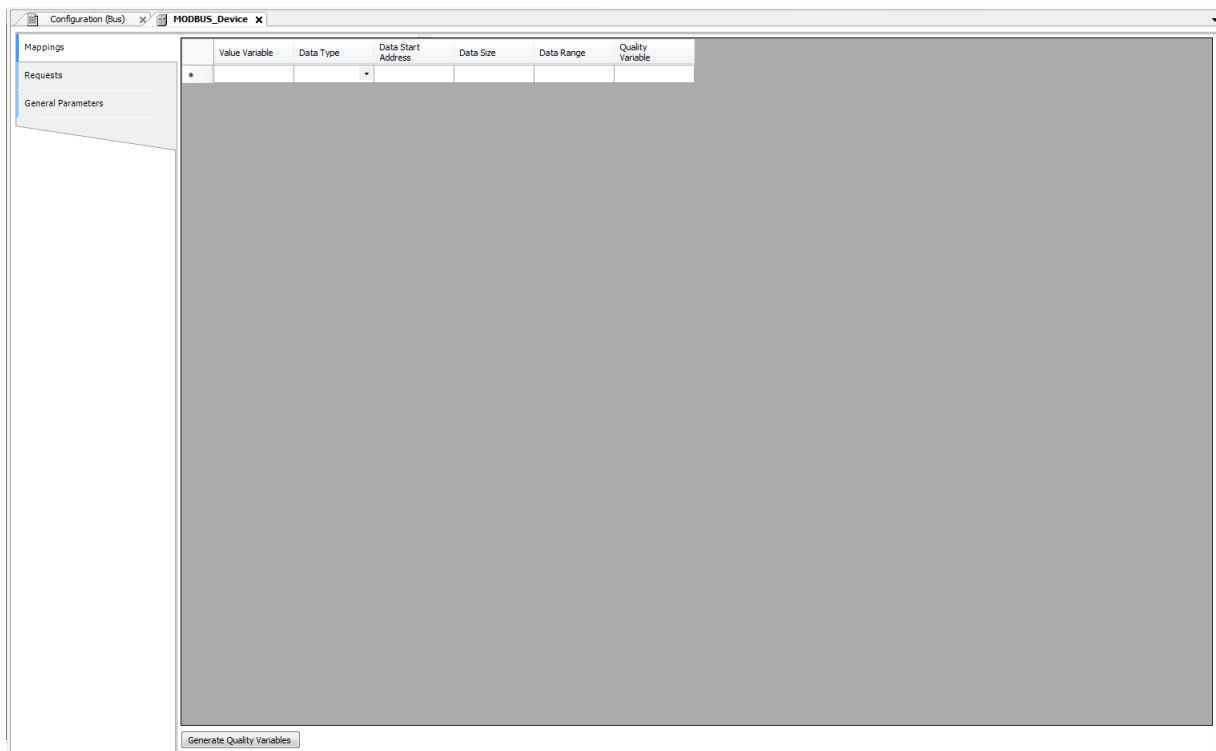
Figure 67: MODBUS Data Mappings Screen

Configuration	Description	Default	Options
Value Variable	Symbolic variable name	-	Name of a variable declared in a program or GVL
Data Type	MODBUS data type	-	Coil - Write (1 bit) Coil - Read (1 bit) Holding Register - Write (16 bits) Holding Register - Read (16 bits) Holding Register – Mask AND (16 bits) Holding Register – Mask OR (16 bits) Input Register (16 bits) Input Status (1 bit)
Data Start Address	Initial address of the MODBUS data	-	1 to 65536
Data Size	Size of the MODBUS data	-	1 to 65536
Data Range	The address range of configured data	-	-

Table 84: MODBUS Mappings Settings

Notes:

Value Variable: this field is used to specify a symbolic variable in MODBUS relation.

Data type: this field is used to specify the data type used in the MODBUS relation.

altus

Data Type	Size [bits]	Description
Coil - Write	1	Writing digital output.
Coil - Read	1	Reading digital output.
Holding Register - Write	16	Writing analog output.
Holding Register - Read	16	Reading analog output.
Holding Register - Mask AND	16	Analog output which can be read or written with AND mask.
Holding Register - Mask OR	16	Analog output which can be read or written with OR mask.
Input Register	16	Analog input which can be only read.
Input Status	1	Digital input which can be only read.

Table 85: Data Types Supported in MODBUS

Data Start Address: Data initial address of a MODBUS mapping.

Data Size: The size value specifies the maximum amount of data that a MODBUS interface can access, from the initial address. Thus, to read a continuous address range, it is necessary that all addresses are declared in a single interface. This field varies with the MODBUS data type configured.

Data Range: This field shows to the user the memory address range used by the MODBUS interface.

4.5.5.1.4. Requests Configuration – Symbolic Mapping Settings

The configuration of the MODBUS requests, viewed in figure below, follow the parameters described in table below:
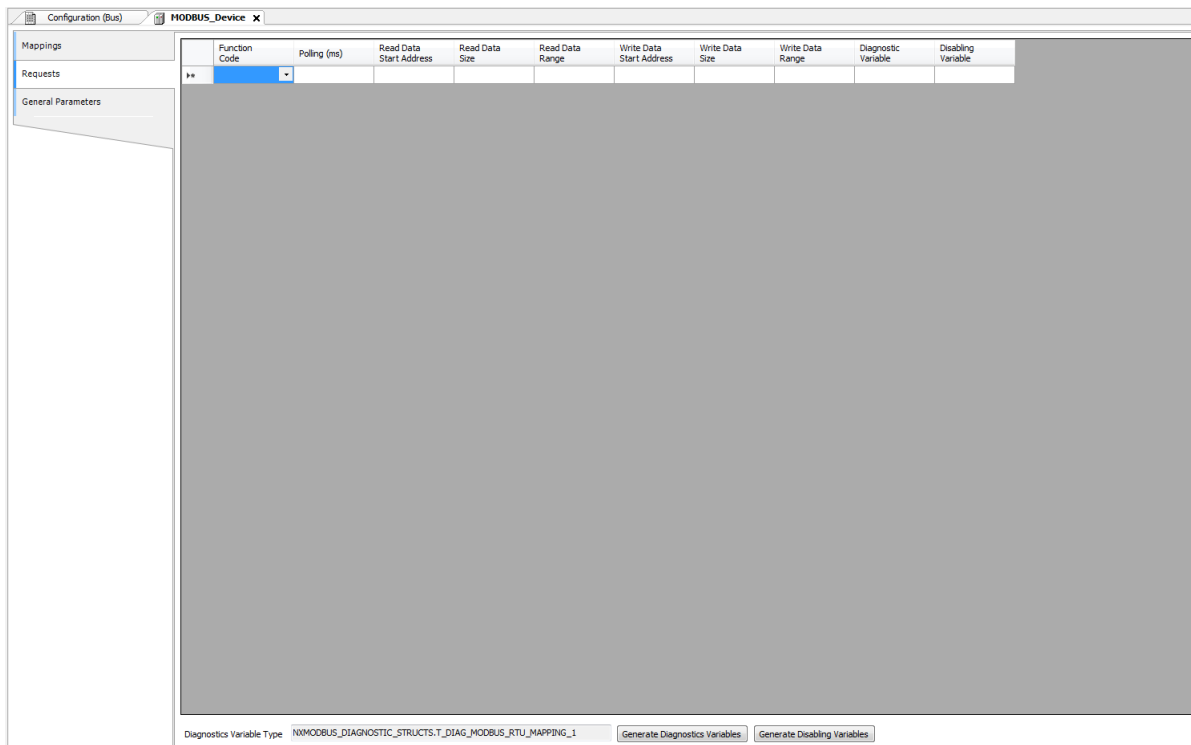
Figure 68: Data Requests Screen MODBUS Master

Configuration	Description	Default Value	Options
Function Code	MODBUS function type	-	01 – Read Coils 02 – Read Input Status 03 – Read Holding Registers 04 – Read Input Registers 05 – Write Single Coil 06 – Write Single Register 15 – Write Multiple Coils 16 – Write Multiple Registers 22 – Mask Write Register 23 – Read/Write Multiple Registers
Polling (ms)	Communication period (ms)	100	0 to 3600000
Read Data Start Address	Initial address of the MODBUS read data	-	1 to 65536
Read Data Size	Size of MODBUS Read data	-	Depends on the function used
Read Data Range	MODBUS Read data address range	-	0 to 2147483646
Write Data Start Address	Initial address of the MODBUS write data	-	1 to 65536
Write Data Size	Size of MODBUS Write data	-	Depends on the function used
Write Data Range	MODBUS Write data address range	-	0 to 2147483647
Diagnostic Variable	Diagnostic variable name	-	Name of a variable declared in a program or GVL
Disabling Variable	Variable used to disable MODBUS relation	-	Field for symbolic variable used to disable, individually, MODBUS requests configured. This variable must be of type BOOL. The variable can be simple or array element and can be in structures.

Table 86: MODBUS Relations Configuration

Notes:

Setting: the number of factory default settings and the values for the column Options may vary according to the data type and MODBUS function (FC).

Function Code: MODBUS (FC) functions available are the following:

altus

Code		Description
DEC	**HEX**	
1	0x01	Read Coils (FC 01)
2	0x02	Read Input Status (FC 02)
3	0x03	Read Holding Registers (FC 03)
4	0x04	Read Input Registers (FC 04)
5	0x05	Write Single Coil (FC 05)
6	0x06	Write Single Holding Register (FC 06)
15	0x0F	Write Multiple Coils (FC 15)
16	0x10	Write Multiple Holding Registers (FC 16)
22	0x16	Mask Write Holding Register (FC 22)
23	0x17	Read/Write Multiple Holding Registers (FC 23)

Table 87: MODBUS Functions Supported by Nexto CPUs

Polling: this parameter indicates how often the communication set for this request must be performed. By the end of a communication will be awaited a time equal to the value configured in the field polling and after that, a new communication will be executed.

Read Data Start Address: field for the initial address of the MODBUS read data.

Read Data Size: the minimum value for the read data size is 1 and the maximum value depends on the MODBUS function (FC) used as below:

- Read Coils (FC 01): 2000
- Read Input Status (FC 02): 2000
- Read Holding Registers (FC 03): 125
- Read Input Registers (FC 04): 125
- Read/Write Multiple Registers (FC 23): 121

Read Data Range: this field shows the MODBUS read data range configured for each request. The initial address, along with the read data size will result in the range of read data for each request.

Write Data Start Address: field for the initial address of the MODBUS write data.

Write Data Size: the minimum value for the write data size is 1 and the maximum value depends on the MODBUS function (FC) used as below:

- Write Single Coil (FC 05): 1
- Write Single Register (FC 06): 1
- Write Multiple Coils (FC 15): 1968
- Write Multiple Registers (FC 16): 123
- Mask Write Register (FC 22): 1
- Read/Write Multiple Registers (FC 23): 121

Write Data Range: this field shows the MODBUS write data range configured for each request. The initial address, along with the read data size will result in the range of write data for each request.

Diagnostic Variable: The MODBUS request diagnostics configured by symbolic mapping or by direct representation, are stored in variables of type *T_DIAG_MODBUS_RTU_MAPPING_1* for Master devices and *T_DIAG_MODBUS_ETH_CLIENT_1* for Client devices and the mapping by direct representation are in 4-byte and 2-word, which are described in Table 88 ("n" is the value configured in the *%Q Start Address of Diagnostics Area* field).

altus

Direct Repre-sentation Variable	Diagnostic variable of type T_DIAG_MODBUS _RTU_MAPPING_1.*	Size	Description
Communication status bits:			
%QX(n).0	byStatus. bCommIdle	BIT	Communication idle (waiting to be exe-cuted).
%QX(n).1	byStatus. bCommExecuting	BIT	Active communication.
%QX(n).2	byStatus. bCommPostponed	BIT	Communication delayed, because the max-imum number of concurrent requests was reached. Deferred communications will be carried out in the same sequence in which they were ordered to avoid indeterminacy. The time spent in this State is not counted for the purposes of time-out. The bCom-mIdle and bCommExecuting bits are false when the bCommPostponed bit is true.
%QX(n).3	byStatus. bCommDisabled	BIT	Communication disabled. The bCommIdle bit is restarted in this condition.
%QX(n).4	byStatus. bCommOk	BIT	Communication terminated previously was held successfully.
%QX(n).5	byStatus. bCommError	BIT	Communication terminated previously had an error. Check error code.
%QX(n).6	byStatus. bCommAborted	BIT	Not used in MODBUS RTU Master.
%QX(n).7	byStatus. bDiag_7_reserved	BIT	Reserved
Last error code (enabled when bCommError = true):			
%QB(n+1)	eLastErrorCode	MASTER_ERROR _CODE (BYTE)	Informs the possible cause of the last error in the MODBUS mapping. Consult Table 104 for further details.
Last exception code received by master:			
%QB(n+2)	eLastExceptionCode	MODBUS _EXCEP-TION (BYTE)	NO_EXCEPTION (0) FUNCTION_NOT_SUPPORTED (1) MAPPING_NOT_FOUND (2) ILLEGAL_VALUE (3) ACCESS_DENIED (128)* MAPPING_DISABLED (129)* IGNORE_FRAME (255)*
Communication statistics:			
%QB(n+3)	byDiag_3_reserved	BYTE	Reserved.
%QW(n+4)	wCommCounter	WORD	Finished communications counter (with or without errors). The user can test when communication has finished testing the variation of this counter. When the value 65535 is reached, the counter returns to zero.
%QW(n+6)	wCommErrorCounter	WORD	Finished communications counter (with er-rors). When the value 65535 is reached, the counter returns to zero.

Table 88: MODBUS Relations Diagnostics

altus

Notes:

Exception Codes: The exception codes presented in this field are values returned by the slave. The definitions of the exception codes 128, 129 and 255 presented in the table are valid only when using Altus slaves. Slaves from other manufacturers might use other definitions for each code.

Disabling Variable: variable of Boolean type used to disable, individually, MODBUS requests configured on request tab via button at the bottom of the window. The request is disabled when the variable, corresponding to the request, is equal to 1, otherwise the request is enabled.

Last Error Code: The codes for the possible situations that cause an error in the MODBUS communication can be consulted below:

Code	Enumerable	Description
1	ERR_EXCEPTION	Reply is in an exception code (see eLastExceptionCode = Exception Code).
2	ERR_CRC	Reply with invalid CRC.
3	ERR_ADDRESS	MODBUS address not found. The address that replied the request was different than expected.
4	ERR_FUNCTION	Invalid function code. The reply's function code was different than expected.
5	ERR_FRAME_DATA_COUNT	The amount of data in the reply was different than expected.
7	ERR_NOT_ECHO	The reply is not an echo of the request (FC 05 and 06).
8	ERR_REFERENCE_NUMBER	Invalid reference number (FC 15 and 16).
9	ERR_INVALID_FRAME_SIZE	Reply shorter than expected.
20	ERR_CONNECTION	Error while establishing connection.
21	ERR_SEND	Error during transmission stage.
22	ERR_RECEIVE	Error during reception stage.
40	ERR_CONNECTION_TIMEOUT	Application level time-out during connection.
41	ERR_SEND_TIMEOUT	Application level time-out during transmission.
42	ERR_RECEIVE_TIMEOUT	Application level time-out while waiting for reply.
43	ERR_CTS_OFF_TIMEOUT	Time-out while waiting CTS = false in transmission.
44	ERR_CTS_ON_TIMEOUT	Time-out while waiting CTS = true in transmission.
128	NO_ERROR	No error since startup.

Table 89: MODBUS Relations Error Codes

> **ATTENTION**
>
> Differently from other application tasks, when a depuration mark in the MainTask is reached, the task of a Master MODBUS RTU instance and any other MODBUS task will stop running at the moment that it tries to perform a writing in a memory area. It occurs in order to keep the consistency of the memory areas data while a MainTask is not running.

4.5.6. MODBUS RTU Slave

This protocol is available for the Hadron Xtorm Series CPU on its serial channels. By selecting this option in MasterTool Xtorm, the CPU becomes a MODBUS communication Slave, allowing connection to MODBUS RTU Master devices. This protocol is only available when the CPU is in execution mode (Run mode).

To configure this protocol, perform the steps below:

1. Add the MODBUS RTU Slave protocol instance to serial channel COM 1 or COM 2 (or both, in case of two communication networks).
2. To perform this procedure, see the section Inserting a Protocol Instance.
3. Configure the serial interface, choosing the communication speed, the behavior of the RTS/CTS signals, the parity, the channel stop bits, among others by double-clicking on the COM 1 or COM 2 serial channel.

altus

4. Configure the general parameters of the MODBUS Slave protocol, such as: Slave Address and communication times (available in the advanced Slave settings button).
5. Add and configure MODBUS relations by specifying the variable name, MODBUS data type, data start address, and data size.

 Descriptions of each configuration are listed below in this section.

4.5.6.1. MODBUS Slave Protocol Configuration via Symbolic Mapping

4.5.6.1.1. MODBUS Slave Protocol General Parameters – Configuration via Symbolic Mapping

The general parameters, found on the MODBUS protocol initial screen (figure below), are defined as.
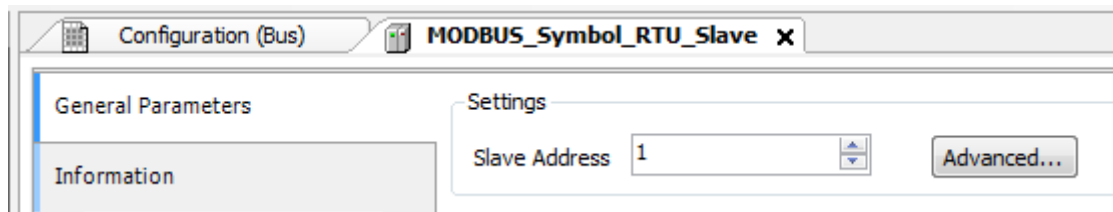
Figure 69: MODBUS RTU Slave Configuration Screen

Configuration	Description	Default	Options
Slave Address	MODBUS slave address	1	1 to 255

Table 90: Slave Configurations

The MODBUS slave protocol communication times, found in the *Advanced...* button on the configuration screen, are divided in: *Task Cycle*, *Send Delay* and *Minimum Interframe* as shown in figure below and in table below.
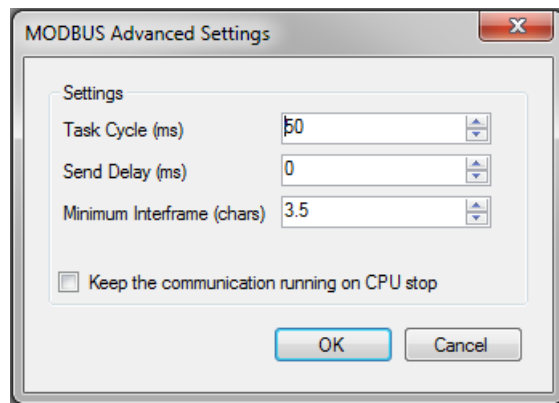
Figure 70: Modbus Slave Advanced Configurations

Configuration	Description	Default	Options
Task Cycle (ms)	Time for the instance execution within the cycle, without considering its own execution time	50	20 to 100
Send Delay (ms)	Delay for the transmission response	0	0 to 65535
Minimum Interframe (chars)	Minimum silence time between different frames	3.5	3.5 to 100.0
Keep the communication running on CPU stop	Enable the MODBUS Symbol Slave to run while the CPU is in STOP or standing in a breakpoint	Unchecked	Checked or unchecked

Table 91: Modbus Slave Advanced Configurations

Notes:

Task Cycle: the user will have to be careful when changing this parameter as it interferes directly in the answer time, data volume for scan and mainly in the CPU resources balance between communications and other tasks.

Send Delay: the answer to a MODBUS protocol may cause problems in certain moments, as in the RS-485 interface or other half-duplex. Sometimes there is a delay between the time of the request from the master and the silence on the physical line (slave delay to put RTS in zero and put the RS-485 in high impedance state). To solve this problem, the master can wait the determined time in this field before sending the new request. On the opposite case, the first bytes transmitted by the master could be lost.

Minimum Interframe: the MODBUS standard defines this time as 3.5 characters, but this parameter is configurable in order to attend the devices which don't follow the standard.

The MODBUS Slave protocol diagnostics and commands configured, either by symbolic mapping or direct representation, are stored in *T_DIAG_MODBUS_RTU_SLAVE_1* variables. For the direct representation mapping, they are also in 4 bytes and 8 words which are described in table below (where "n" is the configured value in the *%Q Start Address of Diagnostics Area*):

Direct Representation Variable	Diagnostic Variable T_DIAG_MODBUS _RTU_SLAVE_1.*	Size	Description
Diagnostic Bits:			
%QX(n).0	tDiag. bRunning	BIT	The slave is in execution mode.
%QX(n).1	tDiag. bNotRunning	BIT	The slave is not in execution (see bit: bInterruptedByCommand).
%QX(n).2	tDiag. bInterruptedByCommand	BIT	The bit bNotRunning was enabled as the slave was interrupted by the user through command bits.
%QX(n).3	tDiag. bConfigFailure	BIT	Discontinued diagnosis.
%QX(n).4	tDiag. bRXFailure	BIT	Discontinued diagnosis.
%QX(n).5	tDiag. bTXFailure	BIT	Discontinued diagnosis.
%QX(n).6	tDiag. bModuleFailure	BIT	Discontinued diagnosis.
%QX(n).7	tDiag. bDiag_7_reserved	BIT	Reserved.

Direct Representation Variable	Diagnostic Variable T_DIAG_MODBUS _RTU_SLAVE_1.*	Size	Description
Error codes:			
%QB(n+1)	eErrorCode	SERIAL_STATUS (BYTE)	0: there are no errors 1: invalid serial port 2: invalid serial port mode 3: invalid baud rate 4: invalid data bits 5: invalid parity 6: invalid stop bits 7: invalid modem signal parameter 8: invalid UART RX Threshold parameter 9: invalid time-out parameter 10: busy serial port 11: UART hardware error 12: remote hardware error 20: invalid transmission buffer size 21: invalid signal modem method 22: CTS time-out = true 23: CTS time-out = false 24: transmission time-out error 30: invalid reception buffer size 31: reception time-out error 32: flow control configured differently from manual 33: invalid flow control for the configured serial port 34: data reception not allowed in normal mode 35: data reception not allowed in extended mode 36: DCD interruption not allowed 37: CTS interruption not allowed 38: DSR interruption not allowed 39: serial port not configured 50: internal error in the serial port
Command bits, automatically initialized:			
%QX(n+2).0	tCommand. bStop	BIT	Stop slave.
%QX(n+2).1	tCommand. bRestart	BIT	Restart slave.
%QX(n+2).2	tCommand. bResetCounter	BIT	Restart diagnostics statistics (counters).
%QX(n+2).3	tCommand. bDiag_19_reserved	BIT	Reserved.
%QX(n+2).4	tCommand. bDiag_20_reserved	BIT	Reserved.
%QX(n+2).5	tCommand. bDiag_21_reserved	BIT	Reserved.

altus

Direct Representation Variable	Diagnostic Variable T_DIAG_MODBUS _RTU_SLAVE_1.*	Size	Description
%QX(n+2).6	tCommand. bDiag_22_reserved	BIT	Reserved.
%QX(n+2).7	tCommand. bDiag_23_reserved	BIT	Reserved.
%QB(n+3)	byDiag_3_reserved	BYTE	Reserved.
Communication Statistics:			
%QW(n+4)	tStat. wRXRequests	WORD	Counter of normal requests received by the slave and answered normally. In case of a broadcast command, this counter is incremented, but it is not transmitted (0 to 65535).
%QW(n+6)	tStat. wTXExceptionResponses	WORD	Counter of normal requests received by the slave and answered with exception code. In case of a broadcast command, this counter is incremented, but it isn't transmitted (0 to 65535). Exception codes: 1: the function code (FC) is legal, but not supported. 2: relation not found in these MODBUS data. 3: illegal value for this field. 128: the master/client hasn't right for writing or reading. 129: the MODBUS relation is disabled.
%QW(n+8)	tStat. wRXFrames	WORD	Counter of frames received by the slave. It's considered a frame something which is processed and it is followed by a Minimum Interframe Silence, in other words, an illegal message is also computed (0 to 65535).
%QW(n+10)	tStat. wRXIllegalRequests	WORD	Illegal request counter. These are frames which start with address 0 (broadcast) or with the MODBUS slave address, but aren't legal requests – invalid syntax, smaller frames, invalid CRC – (0 to 65535).
%QW(n+12)	tStat. wRXOverrunErrors	WORD	Counter of frames with overrun errors during reception – UART FIFO or RX line – (0 to 65535).
%QW(n+14)	tStat. wRXIncompleteFrames	WORD	Counter of frames with construction errors, parity or failure during reception (0 to 65535).
%QW(n+16)	tStat. wCTSTimeOutErrors	WORD	Counter of CTS time-out error, using the RTS/CTS handshake, during the transmission (0 to 65535).
%QW(n+18)	tStat. wDiag_18_Reserved	WORD	Reserved.

Table 92: MODBUS RTU Slave Diagnostic

altus

Note:
Counters: all MODBUS RTU Slave diagnostics counters return to zero when the limit value 65535 is exceeded.

4.5.6.1.2. Configuration of the Relations – Symbolic Mapping Setting

The MODBUS relations configuration, showed on figure below, follows the parameters described on table below:

Figure 71: MODBUS Data Mappings Screen

Configuration	Description	Default	Options
Value Variable	Symbolic variable name	-	Name of a variable declared in a program or GVL
Data Type	MODBUS data type	-	Coil Input Status Holding Register Input Register
Data Start Address	MODBUS data initial address	-	1 to 65536
Absolute Data Start Address	Absolute initial address of MODBUS data according to its type	-	-
Data Size	MODBUS data size	-	1 to 65536
Data Range	Data address range configured	-	-

Table 93: MODBUS Mappings Configurations

Notes:
Value Variable: this field is used to specify a symbolic variable in MODBUS relation.
Data Type: this field is used to specify the data type used in the MODBUS relation.

Data Type	Size [bits]	Description
Coil	1	Digital output that can be read or written.
Input Status	1	Digital input (read only).
Holding Register	16	Analog output that can be read or written.
Input Register	16	Analog input (read only).

Table 94: MODBUS data types supported by Nexto CPUs

Data Start Address: data initial address of the MODBUS relation.
Data Size: the Data Size value sets the maximum amount of data that a MODBUS relation can access from the initial address. Thus, in order to read a continuous range of addresses, it is necessary that all addresses are declared in a single relation. This field varies according to the configured type of MODBUS data.

Data Range: this field shows the user the memory address range used by the MODBUS relation.

ATTENTION

Differently from other application tasks, when a depuration mark in the MainTask is reached, the task of a MODBUS RTU Slave instance and any other MODBUS task will stop running at the moment that it tries to perform a writing in a memory area. It occurs in order to keep the consistency of the memory areas data while a MainTask is not running.

4.5.7. MODBUS Ethernet

The multi-master communication network enables the Hadron Xtorm CPU to read or write MODBUS variables in other controllers or HMIs compatible with MODBUS TCP or MODBUS RTU protocols via TCP. The Hadron Xtorm CPU may simultaneously be Client and Server in the same communication network, or even have more instances associated to the Ethernet interface, regardless if they are MODBUS TCP or MODBUS RTU via TCP, as described in the table 62.

The figure below represents some of the communication possibilities using the MODBUS TCP protocol simultaneously with the MODBUS RTU protocol via TCP.
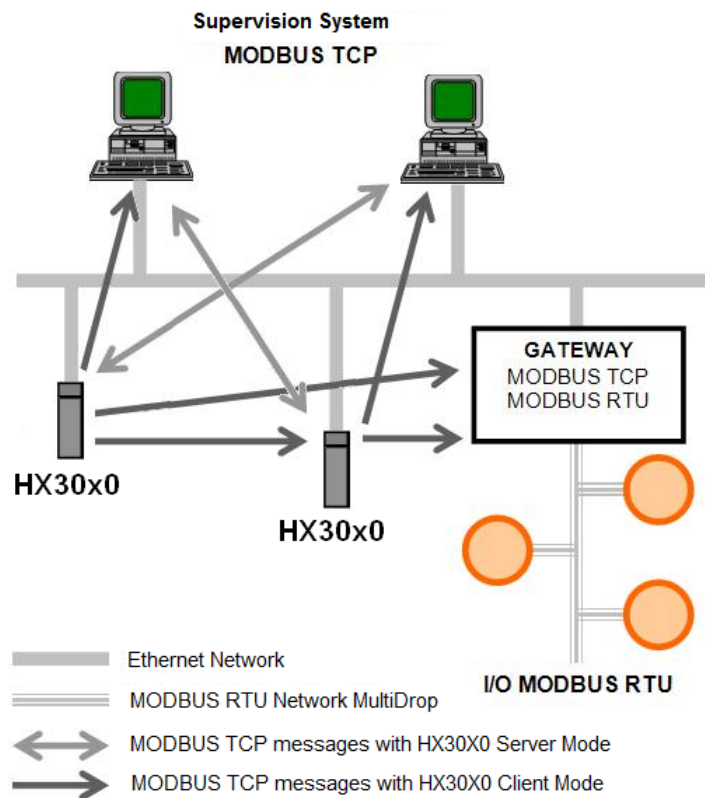
Figure 72: MODBUS TCP Communication Network

The association of MODBUS variables with symbolic variables of the CPU is performed by the user through the definition of relations via the MasterTool Xtorm configuration tool. It is possible to configure up to 32 relations for Server mode and up to 128 relations for Client mode. A relation, in Server mode, can define a large area of MODBUS data and make it available to several Clients. Client mode relations, on the other hand, must respect the maximum data size of a MODBUS function: 125 registers (input registers or holding registers) or 2000 bits (coils or input status). This information is detailed in the description of each protocol.

All relations, in Client or Server mode, can be disabled through symbolic variables identified in the Disable column in MasterTool Xtorm. Disabling occurs through these specific bits, affecting specific relations.

For the Server mode relations, IP addresses clusters (filters) can be defined with writing and reading allowance. This is made through the definition of an IP network address and of a subnet mask, resulting in a group of client IPs which can read and write in the relation variables. This information is detailed in the MODBUS Ethernet Server protocol.

---

altus

When the MODBUS TCP protocol is used in Client mode, the feature of multiple requests can be taken advantage of, using the same TCP connection to accelerate communication with Servers. When this feature is not desired or is not supported by the Server, it can be disabled (relation level action). It is important to note that the maximum number of TCP connections between Client and Server is 63, and if some parameters are changed, inactive communications can be closed, allowing new connections to be opened.

4.5.8. MODBUS Ethernet Client

This protocol is available for the Hadron Xtorm Series CPU on its Ethernet channels. By selecting this option in MasterTool Xtorm, the CPU becomes a MODBUS communication Client, allowing access to other devices with the same protocol, when it is in execution mode (Run Mode).

To configure this protocol, the following steps must be performed:

1. Add the MODBUS Ethernet Client protocol instance to one of the available Ethernet channels (NET 1 ... NET 6). To perform this procedure, see the section Inserting a Protocol Instance.
2. Configure the Ethernet interface. To do this, see the section Ethernet Interfaces Configuration.
3. Configure the general parameters of the MODBUS Client protocol, with the TCP protocol or RTU via TCP.
4. Add and configure devices by defining IP address, port, slave address and communication time-out (available in the Device Advanced Settings button).
5. Add and configure MODBUS mappings by specifying the variable name, data type, data start address, data size, and the variable that will receive the quality data.
6. Add and configure MODBUS requests, specifying the desired function, the request scan time (polling), the initial address (Read/Write), the data size (Read/Write), the variable that will receive the quality data, and the variable responsible for disabling the request.

The descriptions of each configuration are described below in this section.

4.5.8.1. MODBUS Ethernet Client Configuration via Symbolic Mapping

4.5.8.1.1. MODBUS Client Protocol General Parameters – Configuration via Symbolic Mapping

The general parameters, found on the MODBUS protocol configuration initial screen (figure below), are defined as:
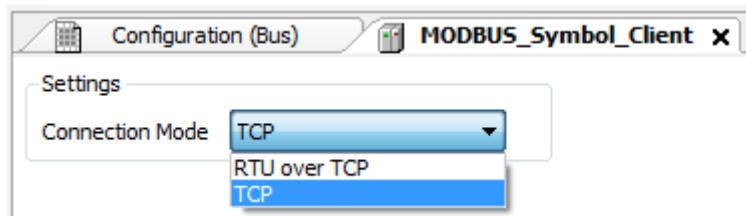
Figure 73: MODBUS Client General Parameters Configuration Screen

Configuration	Description	Default	Options
Connection Mode	Protocol selection	TCP	RTU via TCP TCP

Table 95: MODBUS Client General Configurations

The MODBUS Client protocol diagnostics and commands configured, either by symbolic mapping or direct representation, are stored in *T_DIAG_MODBUS_ETH_CLIENT_1* variables. For the direct representation mapping, they are also in 4 bytes and 8 words which are described in table below (where "n" is the configured value in the *%Q Start Address of Diagnostics Area*):

Direct Representation Variable	Diagnostic Variable T_DIAG_MODBUS _ETH_CLIENT_1.*	Size	Description
Diagnostic Bits:			
%QX(n).0	tDiag. bRunning	BIT	The client is in execution mode.
%QX(n).1	tDiag. bNotRunning	BIT	The client is not in execution mode (see bit bInterruptedByCommand).
%QX(n).2	tDiag. bInterruptedByCommand	BIT	The bit bNotRunning was enabled, as the client was interrupted by the user through command bits.
%QX(n).3	tDiag. bConfigFailure	BIT	Discontinued diagnostics.
%QX(n).4	tDiag. bRXFailure	BIT	Discontinued diagnostics.
%QX(n).5	tDiag. bTXFailure	BIT	Discontinued diagnostics.
%QX(n).6	tDiag. bModuleFailure	BIT	Indicates if there is failure in the module or the module is not present.
%QX(n).7	tDiag. bAllDevicesCommFailure	BIT	Indicates that all devices configured in the Client are in failure.
%QB(n+1)	byDiag_1_reserved	BYTE	Reserved.
Command bits, automatically initialized:			
%QX(n+2).0	tCommand. bStop	BIT	Stop client.
%QX(n+2).1	tCommand. bRestart	BIT	Restart client.
%QX(n+2).2	tCommand. bResetCounter	BIT	Restart the diagnostic statistics (counters).
%QX(n+2).3	tCommand. bDiag_19_reserved	BIT	Reserved.
%QX(n+2).4	tCommand. bDiag_20_reserved	BIT	Reserved.
%QX(n+2).5	tCommand. bDiag_21_reserved	BIT	Reserved.
%QX(n+2).6	tCommand. bDiag_22_reserved	BIT	Reserved.
%QX(n+2).7	tCommand. bDiag_23_reserved	BIT	Reserved.
%QB(n+3)	byDiag_3_reserved	BYTE	Reserved.
Communication Statistics:			
%QW(n+4)	tStat. wTXRequests	WORD	Counter of number of requests transmitted by the client (0 to 65535).
%QW(n+6)	tStat. wRXNormalResponses	WORD	Counter of normal answers received by the client (0 to 65535).
%QW(n+8)	tStat. wRXExceptionResponses	WORD	Counter of answers with exception code (0 to 65535).

altus

Direct Representation Variable	Diagnostic Variable T_DIAG_MODBUS _ETH_CLIENT_1.*	Size	Description
%QW(n+10)	tStat. wRXIllegalResponses	WORD	Counter of illegal answers received by the client – invalid syntax, invalid CRC or not enough bytes received (0 to 65535).
%QW(n+12)	tStat. wDiag_12_reserved	WORD	Reserved.
%QW(n+14)	tStat. wDiag_14_reserved	WORD	Reserved.
%QW(n+16)	tStat. wDiag_16_reserved	WORD	Reserved.
%QW(n+18)	tStat. wDiag_18_Reserved	WORD	Reserved.

Table 96: MODBUS Client Protocol Diagnostics

Note:

Counters: all MODBUS TCP Client diagnostics counters return to zero when the limit value 65535 is exceeded.

4.5.8.1.2. Device Configuration – Configuration via Symbolic Mapping

The devices configuration, shown on figure below, follows the following parameters:
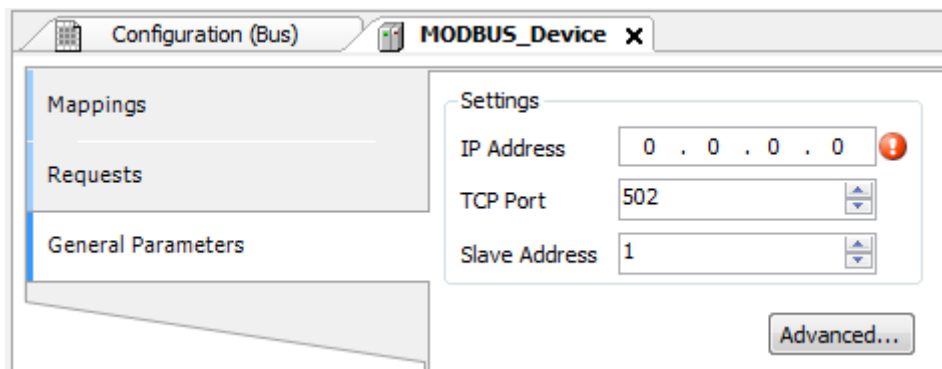
Figure 74: Device General Parameters Settings

Configuration	Description	Default	Options
IP Address	Server IP address	0.0.0.0	1.0.0.1 to 223.255.255.255
TCP Port	TCP port	502	2 to 65534
Slave Address	MODBUS Slave address	1	0 to 255

Table 97: MODBUS Client General Configurations

Notes:

IP Address: IP address of Modbus Server Device.

TCP Port: if there are multiple instances of the protocol added in a single Ethernet interface, different TCP ports must be selected for each instance. Some TCP ports, among the possibilities mentioned above, are reserved and therefore cannot be used. See table Reserved TCP/UDP ports.

Slave address: according to the MODBUS standard, the valid address range for slaves is 0 to 247, where addresses 248 to 255 are reserved. When the master sends a command of writing with the address set to zero, it is performing broadcast requests on the network.

The parameters in the advanced settings of the MODBUS Client device, found on the button *Advanced...* in the *General Parameters* tab are divided into: *Maximum Simultaneous Requests*, *Communication Time-out*, *Mode* of Connection Time-out and *Inactive Time*.

Configuration	Description	Default	Options
Maximum Simultaneous Request	Number of simultaneous request the client can ask from the server	1	1 to 8
Communication Time-out (ms)	Application level time-out in ms	3000	10 to 65535
Mode	Defines when the connection with the server finished by the client	Connection is closed after an inactive time of (s): 10 to 3600.	Connection is closed after a time-out. Connection is closed at the end of each communication. Connection is closed after an inactive time of (s): 10 to 3600.
Inactive Time (s)	Inactivity time	10	3600

Table 98: MODBUS Client Advanced Configurations

Notes:

Maximum Simultaneous Requests: it is used with a high scan cycle. This parameter is fixed in 1 (not editable) when the configured protocol is MODBUS RTU over TCP.

Communication Time-out: the Communication time-out is the time that the client will wait for a server response to the request. For a MODBUS Client device, two variables of the system must be considered: the time the server takes to process a request and the response sending delay in case it is set in the server. It is recommended that the time-out is equal or higher than twice the sum of these parameters. For further information, check Communication Performance section.

Mode: defines when the connection with the server is finished by the client. Below follows the available options:

- Connection is closed after a time-out or Connection is never closed in normal situations: Those options presents the same behavior of Client, close the connection due non response of a request by the Server before reaching the Communication Time-out.
- Connection is closed at the end of each communication: The connection is closed by the Client after finish each request.
- Connection is closed after an Inactive Time: The connection will be closed by the Client if it reach the Inactive Time without performing a request to the Server.

Inactive Time: inactivity connection time.

4.5.8.1.3. Mappings Configuration – Configuration via Symbolic Mapping

The MODBUS relations configuration, showed on figure below, follows the parameters described on table below:

altus
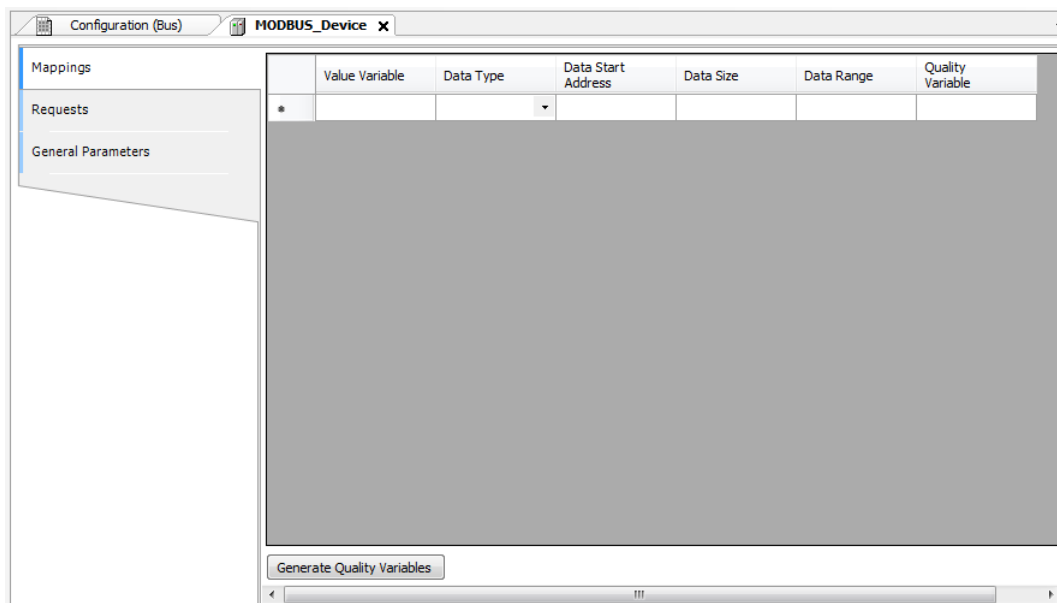
Figure 75: MODBUS Data Type

Configuration	Description	Default	Options
Value Variable	Symbolic variable name	-	Name of a variable declared in a program or GVL
Data Type	MODBUS data type	-	Coil - Write (1 bit) Coil - Read (1 bit) Holding Register - Write (16 bits) Holding Register - Read (16 bits) Holding Register – Mask AND (16 bits) Holding Register – Mask OR (16 bits) Input Register (16 bits) Input Status (1 bit)
Data Start Address	Initial address of the MODBUS data	-	1 to 65536
Data Size	Size of the MODBUS data	-	1 to 65536
Data Range	The address range of configured data	-	-

Table 99: MODBUS Mappings Settings

Notes:

Value Variable: this field is used to specify a symbolic variable in MODBUS relation.

Data type: this field is used to specify the data type used in the MODBUS relation.

Data Type	Size [bits]	Description
Coil - Write	1	Writing digital output.
Coil - Read	1	Reading digital output.
Holding Register - Write	16	Writing analog output.
Holding Register - Read	16	Reading analog output.
Holding Register - Mask AND	16	Analog output which can be read or written with AND mask.
Holding Register - Mask OR	16	Analog output which can be read or written with OR mask.
Input Register	16	Analog input which can be only read.
Input Status	1	Digital input which can be only read.

Table 100: Data Types Supported in MODBUS

Data Start Address: Data initial address of a MODBUS mapping.

Data Size: The size value specifies the maximum amount of data that a MODBUS interface can access, from the initial address. Thus, to read a continuous address range, it is necessary that all addresses are declared in a single interface. This field varies with the MODBUS data type configured.

Data Range: This field shows to the user the memory address range used by the MODBUS interface.

4.5.8.1.4. Requests Configuration – Configuration via Symbolic Mapping

The configuration of the MODBUS requests, viewed in figure below, follow the parameters described in table below:
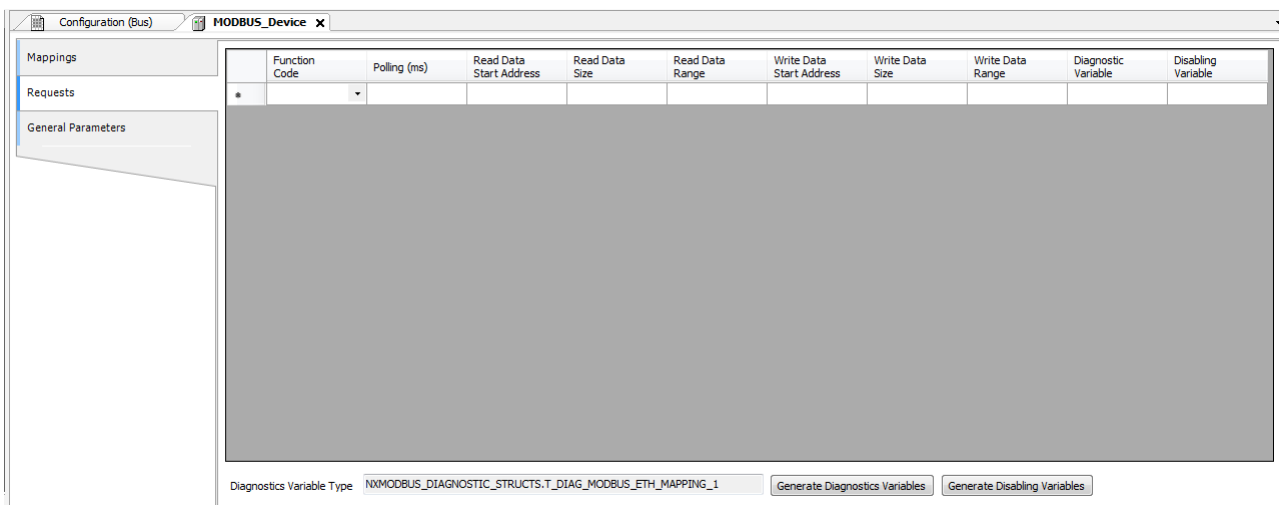
Figure 76: MODBUS Data Request Screen

Configuration	Description	Default Value	Options
Function Code	MODBUS function type	-	01 – Read Coils 02 – Read Input Status 03 – Read Holding Registers 04 – Read Input Registers 05 – Write Single Coil 06 – Write Single Register 15 – Write Multiple Coils 16 – Write Multiple Registers 22 – Mask Write Register 23 – Read/Write Multiple Registers
Polling (ms)	Communication period (ms)	100	0 to 3600000
Read Data Start Address	Initial address of the MODBUS read data	-	1 to 65536
Read Data Size	Size of MODBUS Read data	-	Depends on the function used
Read Data Range	MODBUS Read data address range	-	0 to 2147483646
Write Data Start Address	Initial address of the MODBUS write data	-	1 to 65536
Write Data Size	Size of MODBUS Write data	-	Depends on the function used
Write Data Range	MODBUS Write data address range	-	0 to 2147483647
Diagnostic Variable	Diagnostic variable name	-	Name of a variable declared in a program or GVL
Disabling Variable	Variable used to disable MODBUS relation	-	Field for symbolic variable used to disable, individually, MODBUS requests configured. This variable must be of type BOOL. The variable can be simple or array element and can be in structures.

Table 101: MODBUS Relations Configuration

Notes:

Setting: the number of factory default settings and the values for the column Options may vary according to the data type and MODBUS function (FC).

Function Code: MODBUS (FC) functions available are the following:

altus

Code		Description
DEC	**HEX**	
1	0x01	Read Coils (FC 01)
2	0x02	Read Input Status (FC 02)
3	0x03	Read Holding Registers (FC 03)
4	0x04	Read Input Registers (FC 04)
5	0x05	Write Single Coil (FC 05)
6	0x06	Write Single Holding Register (FC 06)
15	0x0F	Write Multiple Coils (FC 15)
16	0x10	Write Multiple Holding Registers (FC 16)
22	0x16	Mask Write Holding Register (FC 22)
23	0x17	Read/Write Multiple Holding Registers (FC 23)

Table 102: MODBUS Functions Supported by Nexto CPUs

Polling: this parameter indicates how often the communication set for this request must be performed. By the end of a communication will be awaited a time equal to the value configured in the field polling and after that, a new communication will be executed.

Read Data Start Address: field for the initial address of the MODBUS read data.

Read Data Size: the minimum value for the read data size is 1 and the maximum value depends on the MODBUS function (FC) used as below:

- Read Coils (FC 01): 2000
- Read Input Status (FC 02): 2000
- Read Holding Registers (FC 03): 125
- Read Input Registers (FC 04): 125
- Read/Write Multiple Registers (FC 23): 121

Read Data Range: this field shows the MODBUS read data range configured for each request. The initial address, along with the read data size will result in the range of read data for each request.

Write Data Start Address: field for the initial address of the MODBUS write data.

Write Data Size: the minimum value for the write data size is 1 and the maximum value depends on the MODBUS function (FC) used as below:

- Write Single Coil (FC 05): 1
- Write Single Register (FC 06): 1
- Write Multiple Coils (FC 15): 1968
- Write Multiple Registers (FC 16): 123
- Mask Write Register (FC 22): 1
- Read/Write Multiple Registers (FC 23): 121

Write Data Range: this field shows the MODBUS write data range configured for each request. The initial address, along with the read data size will result in the range of write data for each request.

Diagnostic Variable: The MODBUS request diagnostics configured by symbolic mapping or by direct representation, are stored in variables of type *T_DIAG_MODBUS_RTU_MAPPING_1* for Master devices and *T_DIAG_MODBUS_ETH_CLIENT_1* for Client devices and the mapping by direct representation are in 4-byte and 2-word, which are described in Table 88 ("n" is the value configured in the *%Q Start Address of Diagnostics Area* field).

Direct Representation Variable	Diagnostic Variable T_DIAG_MODBUS _ETH_MAPPING_1.*	Size	Description
Communication Status Bits:			
%QX(n).0	byStatus. bCommIdle	BIT	Communication idle (waiting to be executed).

Direct Representation Variable	Diagnostic Variable T_DIAG_MODBUS _ETH_MAPPING_1.*	Size	Description
%QX(n).1	byStatus. bCommExecuting	BIT	Active communication.
%QX(n).2	byStatus. bCommPostponed	BIT	Communication deferred, because the maximum number of concurrent requests was reached. Deferred communications will be carried out in the same sequence in which they were ordered to avoid indeterminacy. The time spent in this State is not counted for the purposes of time-out. The bCommIdle and bCommExecuting bits are false when the bCommPostponed bit is true.
%QX(n).3	byStatus. bCommDisabled	BIT	Communication disabled. The bCommIdle bit is restarted in this condition.
%QX(n).4	byStatus. bCommOk	BIT	Communication terminated previously was held successfully.
%QX(n).5	byStatus. bCommError	BIT	Communication terminated previously had an error. Check error code.
%QX(n).6	byStatus. bCommAborted	BIT	Previously terminated communication was interrupted due to connection failure.
%QX(n).7	byStatus. bDiag_7_reserved	BIT	Reserved.
Last error code (enabled when bCommError = true):			
%QB(n+1)	eLastErrorCode	MASTER_ERROR _CODE (BYTE)	Informs the possible cause of the last error in the MODBUS mapping. Consult Table 104 for further details.
Last exception code received by master:			
%QB(n+2)	eLastExceptionCode	MODBUS _EXCEPTION (BYTE)	NO_EXCEPTION (0) FUNCTION_NOT_SUPPORTED (1) MAPPING_NOT_FOUND (2) ILLEGAL_VALUE (3) ACCESS_DENIED (128)* MAPPING_DISABLED (129)* IGNORE_FRAME (255)*
Communication statistics:			
%QB(n+3)	byDiag_3_reserved	BYTE	Reserved.
%QW(n+4)	wCommCounter	WORD	Communications counter terminated, with or without errors. The user can test when communication has finished testing the variation of this counter. When the value 65535 is reached, the counter returns to zero.
%QW(n+6)	wCommErrorCounter	WORD	Communications counter terminated with errors. When the value 65535 is reached, the counter returns to zero.

Table 103: MODBUS Client Relations Diagnostics

altus

Notes:

Exception Codes: the exception codes show in this filed is the server returned values. The definitions of the exception codes 128, 129 and 255 are valid only with Altus slaves. For slaves from other manufacturers these exception codes can have different meanings.

Disabling Variable: field for the variable used to disable MODBUS requests individually configured within requests. The request is disabled when the variable, corresponding to the request, is equal to 1, otherwise the request is enabled.

Last Error Code: The codes for the possible situations that cause an error in the MODBUS communication can be consulted below:

Code	Enumerable	Description
1	ERR_EXCEPTION	Reply is in an exception code (see eLastExceptionCode = Exception Code).
2	ERR_CRC	Reply with invalid CRC.
3	ERR_ADDRESS	MODBUS address not found. The address that replied the request was different than expected.
4	ERR_FUNCTION	Invalid function code. The reply's function code was different than expected.
5	ERR_FRAME_DATA_COUNT	The amount of data in the reply was different than expected.
7	ERR_NOT_ECHO	The reply is not an echo of the request (FC 05 and 06).
8	ERR_REFERENCE_NUMBER	Invalid reference number (FC 15 and 16).
9	ERR_INVALID_FRAME_SIZE	Reply shorter than expected.
20	ERR_CONNECTION	Error while establishing connection.
21	ERR_SEND	Error during transmission stage.
22	ERR_RECEIVE	Error during reception stage.
40	ERR_CONNECTION_TIMEOUT	Application level time-out during connection.
41	ERR_SEND_TIMEOUT	Application level time-out during transmission.
42	ERR_RECEIVE_TIMEOUT	Application level time-out while waiting for reply.
43	ERR_CTS_OFF_TIMEOUT	Time-out while waiting CTS = false in transmission.
44	ERR_CTS_ON_TIMEOUT	Time-out while waiting CTS = true in transmission.
128	NO_ERROR	No error since startup.

Table 104: MODBUS Relations Error Codes

> **ATTENTION**
>
> Unlike other tasks of an application, when a mark is reached at MainTask debugging, the MODBUS Ethernet Client instance task or any other MODBUS task will stop being executed at the moment it tries to write in the memory area. This occurs in order to maintain data consistency of memory areas while MainTask is not running.

4.5.9. MODBUS Ethernet Server

This protocol is available for the Hadron Xtorm Series CPU on its Ethernet channels. By selecting this option in MasterTool Xtorm, the CPU becomes a MODBUS communication Server, allowing the connection with MODBUS Client devices. This protocol is only available when the CPU is in execution mode (Run mode).

To configure this protocol, the following steps must be performed:

1. Add the MODBUS Server Protocol instance to one of the available Ethernet channels (NET 1 ... NET 6). To perform this procedure, see the section Inserting a Protocol Instance.
2. Configure the Ethernet interface. To do this, see the section Ethernet Interfaces Configuration.
3. Configure the general parameters of the MODBUS Server protocol, such as: TCP port, protocol selection, IP filters for Write and Read (available on the Filters configuration button), and communication times (available on the Server Advanced Settings button).

altus

4. Add and configure the MODBUS mappings by specifying the variable name, data type, data start address, and data size.

The descriptions of each configuration are listed below in this section.

4.5.9.1. MODBUS Server Ethernet Protocol Configuration for Symbolic Mapping

4.5.9.1.1. MODBUS Server Protocol General Parameters – Configuration via Symbolic Mapping

The general parameters, found on the MODBUS protocol initial screen (figure below), are defined as.
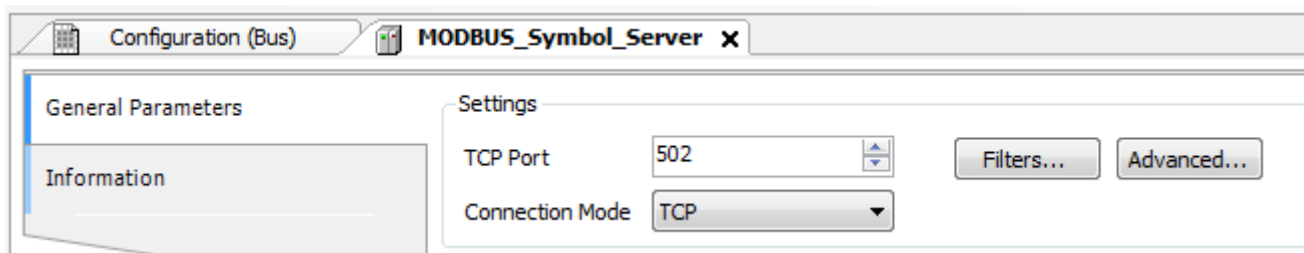
Figure 77: MODBUS Server General Parameters Configuration Screen

Configuration	Description	Default	Options
TCP Port	TCP port	502	2 to 65534
Connection Mode	Protocol selection	TCP	RTU via TCP TCP

Table 105: MODBUS Server General Configurations

Notes:

TCP Port: if there are multiple instances of the protocol added in a single Ethernet interface, different TCP ports must be selected for each instance. Some TCP ports, among the possibilities mentioned above, are reserved and therefore cannot be used. See table Reserved TCP/UDP ports.

The settings present on the *Filters...* button, described in table below, are relative to the TCP communication filters:

Configuration	Description	Default Value	Options
Write Filter IP Address	Specifies a range of IPs with write access in the variables declared in the MODBUS interface.	0.0.0.0	0.0.0.0 to 255.255.255.255
Write Filter Mask	Specifies the subnet mask in conjunction with the IP filter parameter for writing.	0.0.0.0	0.0.0.0 to 255.255.255.255
Read Filter IP Address	Specifies a range of IPs with read access in the variables declared in the MODBUS interface.	0.0.0.0	0.0.0.0 to 255.255.255.255
Read Filter Mask	Specifies the subnet mask in conjunction with the IP filter parameter for reading.	0.0.0.0	0.0.0.0 to 255.255.255.255

Table 106: IP Filters

Note:

Filters: filters are used to establish a range of IP addresses that have write or read access to MODBUS relations, being individually configured. The permission criteria is accomplished through a logical AND operation between the Write Filter Mask and the IP address of the client. If the result is the same as the Write Filter IP Address, the client is entitled to write. For example, if the Write Filter IP Address = 192.168.15.0 and the Write Filter Mask = 255.255.255.0, then only customers with IP address = 192.168.15.x shall be entitled. The same procedure is applied in the Read Filter parameters to define the read rights.

The communication times of the MODBUS server protocol, found on the *Advanced...* button of the configuration screen, are divided into: *Task Cycle* and *Connection Inactivity Time-out*.
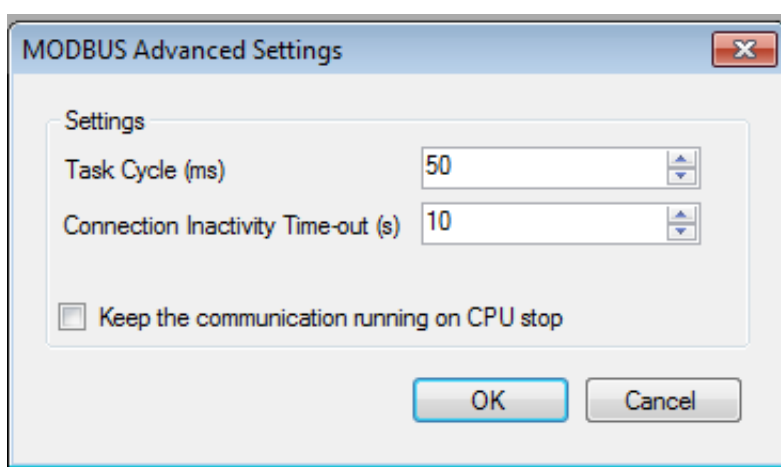
Figure 78: MODBUS Server Advanced Settings Configuration Screen

Configuration	Description	Default Value	Options
Task Cycle (ms)	Time for the instance execution within the cycle, without considering its own execution time	50	5 to 100
Connection Inactivity Time-out (s)	Maximum idle time between client and server before the connection is closed by the server	10	1 to 3600
Keep the communication running on CPU stop.	Enable the MODBUS Symbol Slave to run while the CPU is in STOP or after a breakpoint	Unmarked	Marked or Unmarked

Table 107: MODBUS Server Advanced Configurations

Notes:

Task Cycle: the user has to be careful when changing this parameter as it interferes directly in the answer time, data volume for scanning and mainly in the CPU resources balance between communications and other tasks.

Connection Inactivity Time-out: this parameter was created in order to avoid that the maximum quantity of TCP connections is reached, imagining that inactive connections remain open on account of the most different problems. It indicates how long a connection (client or server) can remain open without being used (without exchanging communication messages). If the specified time is not reached, the connection is closed releasing an input in the connection table.

4.5.9.1.2. *MODBUS Server Diagnostics – Configuration via Symbolic Mapping*

The diagnostics and commands of the MODBUS server protocol configured, either by symbolic mapping or by direct representation, are stored in variables of type *T_DIAG_MODBUS_ETH_SERVER_1* and the mapping by direct representation are in 4-byte and 8-word, which are described in table below (n is the value configured in the *%Q Start Address of Diagnostics Area* field):

Direct Representation Variable	Diagnostic Variable T_DIAG_MODBUS _ETH_SERVER_1 .*	Size	Description
Diagnostic bits:			
%QX(n).0	tDiag. bRunning	BIT	The server is running.
%QX(n).1	tDiag. bNotRunning	BIT	The server is not running (see bit bInterruptedByCommand).
%QX(n).2	tDiag. bInterruptedByCommand	BIT	The bit bNotRunning was enabled, because the server was interrupted by the user through the command bit.
%QX(n).3	tDiag. bConfigFailure	BIT	Discontinued diagnostic.
%QX(n).4	tDiag. bRXFailure	BIT	Discontinued diagnostic.
%QX(n).5	tDiag. bTXFailure	BIT	Discontinued diagnostic.
%QX(n).6	tDiag. bModuleFailure	BIT	Discontinued diagnostic.
%QX(n).7	tDiag. bDiag_7_reserved	BIT	Reserved.
%QB(n+1)	byDiag_1_reserved	BYTE	Reserved.
Command bits, restarted automatically:			
%QX(n+2).0	tCommand. bStop	BIT	Stop the server.
%QX(n+2).1	tCommand. bRestart	BIT	Restart the server.
%QX(n+2).2	tCommand. bResetCounter	BIT	Reset diagnostics statistics (counters).
%QX(n+2).3	tCommand. bDiag_19_reserved	BIT	Reserved.
%QX(n+2).4	tCommand. bDiag_20_reserved	BIT	Reserved.
%QX(n+2).5	tCommand. bDiag_21_reserved	BIT	Reserved.
%QX(n+2).6	tCommand. bDiag_22_reserved	BIT	Reserved.
%QX(n+2).7	tCommand. bDiag_23_reserved	BIT	Reserved.
%QB(n+3)	byDiag_3_reserved	BYTE	Reserved.
Communication statistics:			
%QW(n+4)	tStat. wActiveConnections	WORD	Number of established connections between client and server (0 to 64).

Direct Representation Variable	Diagnostic Variable T_DIAG_MODBUS _ETH_SERVER_1 .*	Size	Description
%QW(n+6)	tStat. wTimeoutClosedConnections	WORD	Connections counter, between the client and server, interrupted after a period of inactivity - time-out (0 to 65535).
%QW(n+8)	tStat. wClientClosedConnections	WORD	Connections counter interrupted due to customer request (0 to 65535).
%QW(n+10)	tStat. wRXFrames	WORD	Ethernet frames counter received by the server. An Ethernet frame can contain more than one request (0 to 65535).
%QW(n+12)	tStat. wRXRequests	WORD	Requests received by the server counter and answered normally (0 to 65535).
%QW(n+14)	tStat. wTXExceptionResponses	WORD	Requests received by the server counter and answered with exception codes (0 to 65535). The exception codes are listed below: 1: the function code (FC) is legal, but not supported. 2: relation not found in these data MODBUS. 3: illegal value for the address. 128: the master/client has no right to read or write. 129: MODBUS relation is disabled.
%QW(n+16)	tStat. wRXIllegalRequests	WORD	Illegal requests counter (0 to 65535).
%QW(n+18)	tStat. wDiag_18_Reserved	WORD	Reserved.

Table 108: MODBUS Server Diagnostics

Note:

Counters: all counters of the MODBUS Ethernet Server Diagnostics return to zero when the limit value 65535 is exceeded.

4.5.9.1.3.   Mapping Configuration – Configuration via Symbolic Mapping

The MODBUS relations configuration, showed on figure below, follows the parameters described on table below:

Figure 79: MODBUS Server Data Mappings Screen

Configuration	Description	Default Value	Options
Value Variable	Symbolic variable name	-	Name of a variable declared in a program or GVL
Data Type	MODBUS data type	-	Coil Input Status Holding Register Input Register
Data Start Address	Starting address of the MODBUS data	-	1 to 65536
Absolute Data Start Address	Start address of absolute data of Modbus as its type	-	-
Data Size	Size of the MODBUS data	-	1 to 65536
Data Range	Data range address configured	-	-

Table 109: MODBUS Ethernet Mappings Configuration

Notes:

Value Variable: this field is used to specify a symbolic variable in MODBUS relation.

Data Type: this field is used to specify the data type used in the MODBUS relation.

Data Start Address: data initial address of the MODBUS relation.

Absolute Data Start Address: absolute start address of the MODBUS data according to their type. For example, the Holding Register with address 5 has absolute address 400005. This field is read only and is available to assist in Client/Master MODBUS configuration that will communicate with this device. The values depend on the base address (offset) of each data type and allowed MODBUS address for each data type.

Data Size: the Data Size value sets the maximum amount of data that a MODBUS relation can access from the initial address. Thus, in order to read a continuous range of addresses, it is necessary that all addresses are declared in a single relation. This field varies according to the configured type of MODBUS data.

Data Range: is a read-only field and reports on the range of addresses that is being used by this mapping. It is formed by the sum of the fields *Data Start Address* and *Data Size*. There can be no range overlays with others mappings of the same *Data Type*.

> **ATTENTION**
>
> Unlike other tasks of an application, when a mark is reached at MainTask debugging, the MODBUS Ethernet Server instance task or any other MODBUS task will stop being executed at the moment it tries to write in the memory area. This occurs in order to maintain data consistency of memory areas while MainTask is not running.

4.5.10. DNP3 – Data Types

The table below shows the variable type supported by the HX3040 for each of the DNP3 protocol data types.

Mapping Type	IEC Variables Type	StaticGroup Valid Variation	EventGroup Valid Variation	Event Class Range
BI: Binary Input	BOOL BIT	g01v01 - Digital input in packed format g01v02 - Digital input with quality	g02v01 - Digital input event without time g02v02 - Digital input event with absolute time g02v03 - Digital input event with relative time	0..3

altus

Mapping Type	IEC Variables Type	StaticGroup Valid Variation	EventGroup Valid Variation	Event Class Range
DBI: Double Binary Input	DBP	g03v01 - Double digital input in packed format g03v02 - Double digital input with quality	g04v01 - Double digital input event without time g04v02 - Double digital input event with absolute time g04v03 - Double digital input event with relative time	0..3
BO: Binary Output	BOOL DBP BIT	g10v01 - Digital output in packed format g10v02 - Digital output with quality	-	0
CN: Counter	UINT	g20v02 - 16-bit counter with quality g20v06 - 16-bit counter without quality	g22v02 - 16-bit counter event with quality g22v06 - 16-bit counter event with quality and time	0..3
	UDINT	g20v01 - 32-bit counter with quality g20v05 - 32-bit counter without quality	g22v01 - 32-bit counter event with quality g22v05 - 32-bit counter event with quality and time	0..3
FCN: Frozen Counter	UINT	g21v02 - 16-bit frozen counter with quality g21v10 - 16-bit frozen counter without quality	g23v02 - 16-bit frozen counter event with quality g23v06 - 16-bit frozen counter event with quality and time	0..3
	UDINT	g21v01 - 32-bit frozen counter with quality g21v09 - 32-bit frozen counter without quality	g23v01 - 32-bit frozen counter event with quality g23v05 - 32-bit frozen counter event with quality and time	0..3
AI: Analog Input	INT	g30v02 - 16-bit analog input with quality g30v04 - 16-bit analog input without quality	g32v02 - 16-bit analog input event without time g32v04 - 16-bit analog input event with time	0..3
	DINT	g30v01 - 32-bit analog input with quality g30v03 - 32-bit analog input without quality	g32v01 - 32-bit analog input event without time g32v03 - 32-bit analog input event with time	0..3
	REAL	g30v05 - Simple precision analog input with quality	g32v05 - Simple precision analog input event without time g32v07 - Single precision analog input event in floating point with time	0..3
AO: Analog Output	INT	g40v02 - 16-bit analog output with quality	-	0
	DINT	g40v01 - 32-bit analog output with quality	-	0
	REAL	g40v03 - Simple precision analog output with quality	-	0

Table 110: Declaration of Variables for DNP3

altus

4.5.11. DNP3 Ethernet Client

This protocol is available for the Hadron Xtorm Series CPU on its Ethernet channels. By selecting this option in MasterTool Xtorm, the CPU becomes a client of the DNP3 communication. The maximum limit of DNP3 Client driver instances depends on how many protocol drivers are instantiated and supported by the HX3040 CPU in total. This protocol is enabled only when the CPU is in execution mode (RUN mode).

To configure this protocol, the following steps must be performed:

1. Add the DNP3 Client protocol instance to one of the available Ethernet channels (NET 1 .. NET 6). To perform this procedure, see the section Inserting a Protocol Instance.
2. Configure the Ethernet interface. To do this, see the section Ethernet Interfaces Configuration.
3. Configure the general parameters of the DNP3 Client protocol, with the Link Address mode.
4. Add and configure outstation devices, defining the appropriate parameters. For each driver instantiated, there is a maximum limit of 32 outstations.
5. Add and configure DNP3 mappings, specifying the variable name, static group, indexes, size, range, class, and quality.
6. Configure the link layer parameters, specifying the addressing and confirmation messages.
7. Configure application layer parameters, specifying application layer configurations, class scanning, and automatic requests.

The descriptions of each configuration are listed below in this section.

4.5.11.1. Configuration of the DNP3 Client Mappings

The configurations of the General Parameters of a DNP3 Client, shown in the figure below, follow the parameters described in the following table.
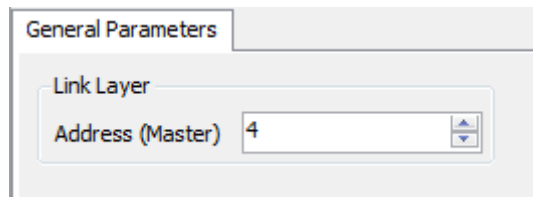
Figure 80: DNP3 Client General Parameters Screen

Configuration	Description	Default	Options
Address (Master)	Defines the address of the DNP3 master.	4	0 to 65519

Table 111: DNP3 Client General Parameter Configuration

The configuration of the DNP3 Client relations, shown in the figure below, follows the parameters described in the following table.

Figure 81: DNP3 Client Data Mapping Screen

Configuration	Description	Default	Options
Value Variable	Symbolic variable name.	-	Name of a variable declared in a program or GVL.
Static Group	Configuration of the static data.	-	g01 – Digital Input g03 – Double Digital Input g10 – Digital Output g20 – Counter g21 – Frozen Counter g30 – Analog Input* g40 – Analog Output*
Index	Index of the first DNP3 mapping point.	-	0 to 65535
Size	DNP3 data size.	-	- (automatically calculated)
Range	Address range of the configured data.	-	- (automatically calculated)
Quality Variable	Name of the symbolic variable that stores the mapping quality.	-	Variable of type QUALITY
Generate Quality Variable	This button automatically generates variables of type QUALITY in the GVL Qualities for each declared Value Variable. For more information, please see the section Quality Conversions	-	-

Table 112: Configuration of the DNP3 Client Mappings

Notes:

Value Variable: Name of the symbolic variable to be mapped. When a reading command is sent, the return sent in the response will be stored in this variable. When it is a writing command, the value written will be copied from this variable. The variable can be simple, array, array element, simple structure element, or structure array element.

Static Group: It is used to select in which DNP3 group the mapping will be created.

Index: The field is used to set the index of the first mapping point in the DNP3 protocol.

Size: This is a read-only field that tells how many points are being used by this mapping. The value of this field varies according to the type of variable set in the mapping.

Range: This is a read-only field that informs the address range that is being used by this mapping. It is formed by the sum of the "Indexes" and "Size" fields. There can be no range overlaps with other mappings of the same type.

Quality Variable: Name of the symbolic variable that stores the quality of the mapping, which is always of type QUALITY, available from LibDataTypes library. The variable can be simple, array or array element and may be in structures.

ATTENTION

When variables from group 30 (Analog Inputs) and group 40 (Analog Outputs) receive values from the Outstation that cannot be stored due to the difference between the variations of the data received and the storage variable mapped in DNP3 Client, the quality of that point will be INVALID and the value of the point will be the last valid value read by DNP3 Client. This will only occur if the DNP3 Server responds in a wider range than the one mapped in the DNP3 Client. In case the DNP3 Server responds in the same or lower range, the point value and quality will be stored according to the DNP3 Server response.

For example: Variable of type INT (16-bit) mapped in DNP3 Client and DNP3 Server responds in 32-bit variation. In this case, the value contained in the response of DNP3 Server will not be updated and the quality will be INVALID.

ATTENTION

The DNP3 Client driver does not generate events based on the values of its mappings. The events must be generated by the respective IED. Therefore, if the IED does not support event generation, the DNP3 Client driver will not be able to generate events.

4.5.11.2. Configuration of the DNP3 Client Link Layer

The configurations of the DNP3 Client link layer parameters, seen in the figure below, follow the parameters described in the following table.
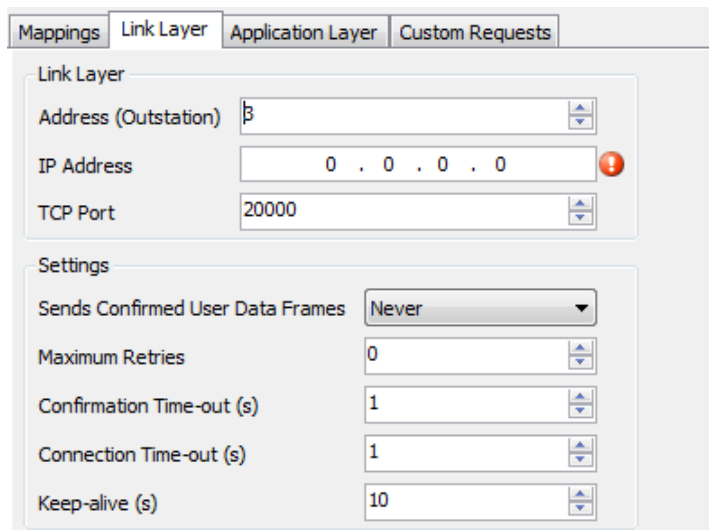
Figure 82: Configuration Screen of the DNP3 Client Link Layer

altus

Configuration	Description	Default	Options
Address (Outstation)	DNP3 source address of this Outstation.	3	0 to 65519
IP Address	IP address of this Outstation.	0.0.0.0	1.0.0.1 to 223.255.255.254
TCP Port	Listen port address to connect to this Outstation.	20000	0 to 65535
Sends Confirmed User Data Frames	Select the message confirmation mode.	Never	Never Sometimes Always
Maximum Retries	Number of times this Outstation will retransmit the message if it does not receive the confirmation from the client.	0	0 to 10
Confirmation Time-Out(s)	Time for this Outstation to send the confirmation message again.	1	1 to 86400
Connection Time-Out(s)	Maximum time for this Outstation to connect to the DNP3 Client through the TCP protocol.	1	1 to 10
Keep Alive (s)	Link test time.	10	1 to 60

Table 113: Configuration of the DNP3 Client Link Layer

Notes:

Address (Outstation): Source DNP3 address of this outstation.

IP Address: IP address of this outstation.

Porta TCP: Listen port address for connection to this outstation.

Sends Confirmed User Data Frames: Sends messages requesting the outstation to confirm receipt or not. The available link layer confirmation modes are as follows:

	Options	Description
Link Layer Confirmation Mode	Never	Never requests confirmation of messages
	Always	Always requests confirmation of messages
	Sometimes	Requests confirmation only for multifragmented messages

Table 114: Confirmation Modes of the DNP3 Client Link Layer

Notes:

Maximum Retries: The number of times the messages will be transmitted if outstation confirmation is not received. This is enabled when, in the *"Link Layer Confirmation Mode"* option, either *"Always"* or *"Sometimes"* is selected.

Confirmation Time-Out(s): Determines the time (in seconds) to wait for the outstation to send the confirmation message. This is enabled when, in the *"Link Layer Confirmation Mode"* option, either *"Always"* or *"Sometimes"* is selected.

Keep Alive(s): Time interval for sending link test messages (*"TCP keep-alive"* or *"serial link status"*) after last received DNP3 message. If no response is received for the link test message, the current connection will be terminated.

4.5.11.3. Configuration of the DNP3 Client Application Layer

The application layer configuration of a DNP3 Server added under a DNP3 Client is shown in the figure below. The parameters configured in this screen are described in the following table.
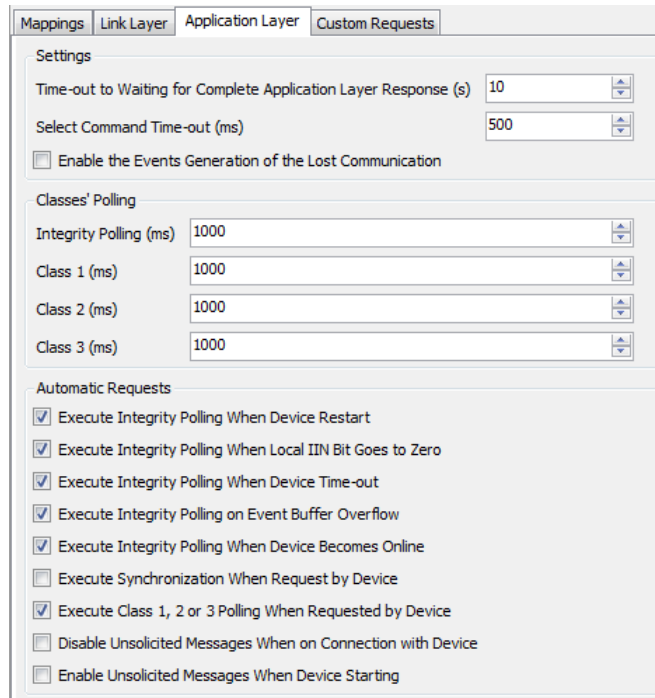
Figure 83: Configuration Screen of the DNP3 Client Application Layer

Configuration	Description	Default	Options
Time-out to Waiting for Complete Application Layer Response (s)	Maximum time that the outstation will wait for a complete response from the application layer.	10	1 to 86400
Select Command Time-out (ms)	Maximum time for the execution of the selection command.	500	100 to 10000
Enable the Events Generation of the Lost Communication	Enables the generation of events in situations of communication loss and reestablishment of communication with the outstation.	Disabled	Enabled Disabled
Integrity Polling (ms)	Determines the time (in milliseconds) in which the scan of all mapped points is executed.	1000	0 to 86400000
Class 1 (ms)	Time the scan of the points mapped to "Class 1" is executed.	1000	0 to 86400000
Class 2 (ms)	Time the scan of the points mapped to "Class 2" is executed.	1000	0 to 86400000
Class 3 (ms)	Time the scan of the points mapped to "Class 3" is executed.	1000	0 to 86400000
Execute Integrity polling When Device Restart	Updates the values of all points in the outstation device and also reads all events when the DEVICE_RESTART bit of the IIN is set to one.	Enabled	Enabled Disabled
Execute Integrity Polling When Local IIN Bit Goes to Zero	Runs integrity scan when the LOCAL bit of the IIN switches from one to zero.	Enabled	Enabled Disabled
Execute Integrity Polling When Device Time-out	Runs the integrity scan when the outstation is operating in local mode and is not accessible by the master.	Enabled	Enabled Disabled

Configuration	Description	Default	Options
Execute Integrity Polling on Event Buffer Overflow	Runs the integrity scan after an overflow in the outstation event queue, indicated by the EVENT_BUFFER_OVERFLOW bit of the IIN.	Enabled	Enabled Disabled
Execute Integrity Polling When Device Becomes Online	Runs the integrity scan when communication with the master is re-established after a time-out.	Enabled	Enabled Disabled
Execute Synchronization When Request By Device	Allows the master to send the time setting command when the outstation requests it.	Disabled	Enabled Disabled
Execute Class 1,2 or 3 Polling When Requested by Device	Performs the reading of the events of the three classes when one of the CLASS_1_EVENTS, CLASS_2_EVENTS and CLASS_3_EVENTS bits of the IIN is set to one.	Enabled	Enabled Disabled
Disable Unsolicited Messages When on Connection with Device	Sends the command to disable unsolicited messages when the outstation connects. When this field is checked, the "Enable unsolicited messages when device starts" checkbox is deselected and disabled.	Disabled	Enabled Disabled
Enable Unsolicited Messages When Device Starting	Sends the command to enable unsolicited messages when the outstation connects.	Disabled	Enabled Disabled

Table 115: Configuration of the DNP3 Client Application Layer

4.5.11.4.   Custom Requests of the DNP3 Client

The configurations of the DNP3 Client custom relations, visualized in the figure below, follow the parameters described in the following table.
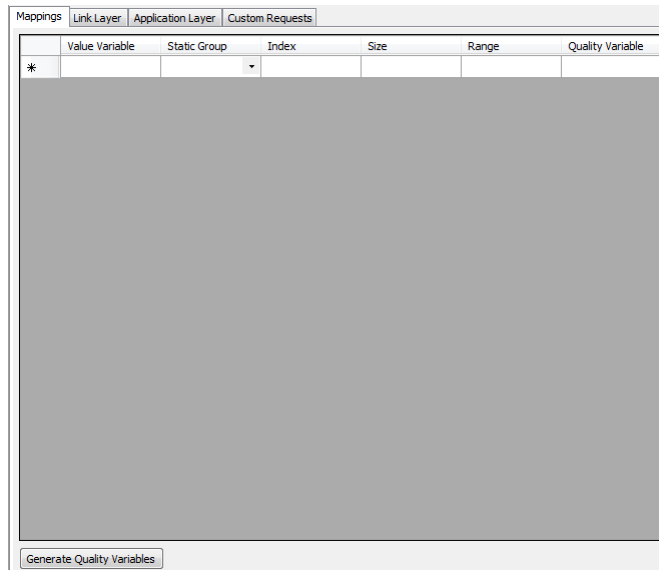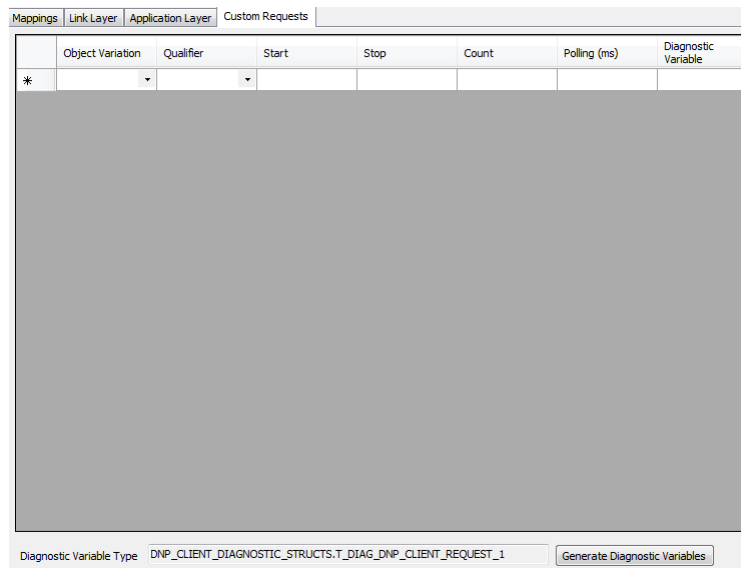

Figure 84: Message Screen of DNP3 Client Custom Requests

Configuration	Description	Padrão	Opções
Object Variation	Defines the group and the variation of the objects to be required.	-	g01v00 - Binary Input Default Variation. g01v01 - Binary Input Packed Format. g01v02 - Binary Input With Flags. g02v00 - Binary Input Event Default Variation. g02v01 - Binary Input Event without Time. g02v02 - Binary Input Event with Absolute Time. g02v03 - Binary Input Event with Relative Time. g03v00 - Double-bit Binary Input Default Variation. g03v01 - Double-bit Binary Input Packed Format. g03v02 - Double-bit Binary Input With Flags. g04v00 - Double-bit Binary Input Event Default Variation. g04v01 - Double-bit Binary Input Event without Time. g04v02 - Double-bit Binary Input Event with Absolute Time. g04v03 - Double-bit Binary Input Event with Relative Time. g10v00 - Binary Output Default Variation. g10v01 - Binary Output Packed Format. g10v02 - Binary Output Status with Flags. g20v00 - Counter Default Variation. g20v01 - Counter 32 bits with flags. g20v02 - Counter 16 bits with flags. g20v05 - Counter 32 bits without flags. g20v06 - Counter 16 bits without flags. g21v00 - Frozen Counter Default Variation. g21v01 - Frozen Counter 32 bits with Flag. g21v02 - Frozen Counter 16 bits with Flag. g21v09 - Frozen Counter 32 bits without Flag. g21v10 - Frozen Counter 16 bits without Flag. g22v00 - Counter Event Default Variation. g22v01 - Counter Event 32 bits with Flag. g22v02 - Counter Event 16 bits with Flag. g22v05 - Counter Event 32 bits with Flag and Time. g22v06 - Counter Event 16 bits with Flag and Time. g23v00 - Frozen Counter Event Any Variation. g23v01 - Frozen Counter Event 32 bits with Flag. g23v02 - Frozen Counter Event 16 bits with Flag. g23v05 - Frozen Counter Event 32 bits with Flag and Time. g23v06 - Frozen Counter Event 16 bits with Flag and Time. g30v00 - Analog Input Default Variation. g30v01 - Analog Input 32 bits with Flag. g30v02 - Analog Input 16 bits with Flag. g30v03 - Analog Input 32 bits without Flag. g30v04 - Analog Input 16 bits without Flag. g30v05 - Analog Input Single Precision, Floating-point with Time g32v00 - Analog Input Event Default Variation. g32v01 - Analog Input Event 32 bits without Time. g32v02 - Analog Input Event 16 bits without Time. g32v03 - Analog Input Event 32 bits with Time. g32v04 - Analog Input Event 16 bits with Time. g32v05 - Analog Input Event Single Precision, Floating-point without Time.
			g32v07 - Analog Input Event Single Precision, Floating-point with Time. g40v00 - Analog Output Status Default Variation. g40v01 - Analog Output Status 32 bits with Flag. g40v02 - Analog Output Status 16 bits with Flag. g40v03 - Analog Output Status Single Precision, Floating-point with Flag. g60v01 - Class 0 Data. g60v02 - Class 1 Data. g60v03 - Class 2 Data. g60v04 - Class 3 Data.
Qualifier	Identifies the type of request to be held.	-	Range All Counter

altus

Configuration	Description	Padrão	Opções
Start	Defines the first point of a range that will be required. Can only be filled in if the "Qualifier" column has the "Range" value selected.	0	0 to 65535
Stop	Defines the last point of a range that will be required. Can only be filled in if the "Qualifier" column has the value "Range" selected.	0	0 to 65535
Count	Defines how many points will be required. Can only be filled in if the "Qualifier" column has the "Counter" value selected.	0	0 to 65535
Polling (ms)	Sets the period (in milliseconds) that the request occurs.	1000	0 to 86400000
Diagnostic Variable	Name of the symbolic variable that will receive the diagnosis of the request.	-	Name of a variable declared in a program or GVL

Table 116: Configuration of DNP3 Client Custom Requests

Notes:

Generated Diagnostic Variables: Diagnostic symbolic variables can be generated automatically using the *"Generate Diagnostic Variables"* button. Clicking on this button will create a variable of the type "T_DIAG_DNP_CLIENT_ REQUEST_1", in the GVL "ReqDiagnostics", for each of the requests.

Qualifier: Identifies the type of request that will be performed. The available options depend on the type of Object Variation chosen. The table describes the Qualifier types and the possible qualifier values for each of the request groups.

Function Type	Configuration	Description
Qualifier	All	All points of the group and variation are required.
	Range	Only will be requested a range of points defined by the columns "Start" and "Stop.
	Counter	To request a defined number of events in the "Counter" column, for example, "read 10 events".

Table 117: Qualifier Types in DNP3 Client Parameters

Object Variation	Allowed Qualifier
g01v00 - Binary Input Default Variation. g01v01 - Binary Input Packed Format. g01v02 - Binary Input With Flags.	All Range
g02v00 - Binary Input Event Default Variation. g02v01 - Binary Input Event without Time. g02v02 - Binary Input Event with Absolute Time. g02v03 - Binary Input Event with Relative Time.	All Counter
g03v00 - Double-bit Binary Input Default Variation. g03v01 - Double-bit Binary Input Packed Format. g03v02 - Double-bit Binary Input With Flags.	All Range

altus

Object Variation	Allowed Qualifier
g04v00 - Double-bit Binary Input Event Default Variation. g04v01 - Double-bit Binary Input Event without Time. g04v02 - Double-bit Binary Input Event with Absolute Time. g04v03 - Double-bit Binary Input Event with Relative Time.	All Counter
g10v00 - Binary Output Default Variation. g10v01 - Binary Output Packed Format. g10v02 - Binary Output Status with Flags.	All Range
g20v00 - Counter Default Variation. g20v01 - Counter 32 bits with flags. g20v02 - Counter 16 bits with flags. g20v05 - Counter 32 bits without flags. g20v06 - Counter 16 bits without flags.	All Range
g21v00 - Frozen Counter Default Variation. g21v01 - Frozen Counter 32 bits with Flag. g21v02 - Frozen Counter 16 bits with Flag. g21v09 - Frozen Counter 32 bits without Flag. g21v10 - Frozen Counter 16 bits without Flag.	All Range
g22v00 - Counter Event Default Variation. g22v01 - Counter Event 32 bits with Flag. g22v02 - Counter Event 16 bits with Flag. g22v05 - Counter Event 32 bits with Flag and Time. g22v06 - Counter Event 16 bits with Flag and Time.	All Counter
g23v00 - Frozen Counter Event Any Variation. g23v01 - Frozen Counter Event 32 bits with Flag. g23v02 - Frozen Counter Event 16 bits with Flag. g23v05 - Frozen Counter Event 32 bits with Flag and Time. g23v06 - Frozen Counter Event 16 bits with Flag and Time.	All Counter
g30v00 - Analog Input Default Variation. g30v01 - Analog Input 32 bits with Flag. g30v02 - Analog Input 16 bits with Flag. g30v03 - Analog Input 32 bits without Flag. g30v04 - Analog Input 16 bits without Flag. g30v05 - Analog Input Single Precision, Floating-point with Time	All Range
g32v00 - Analog Input Event Default Variation. g32v01 - Analog Input Event 32 bits without Time. g32v02 - Analog Input Event 16 bits without Time. g32v03 - Analog Input Event 32 bits with Time. g32v04 - Analog Input Event 16 bits with Time. g32v05 - Analog Input Event Single Precision, Floating-point without Time. g32v07 - Analog Input Event Single Precision, Floating-point with Time.	All Counter
g40v00 - Analog Output Status Default Variation. g40v01 - Analog Output Status 32 bits with Flag. g40v02 - Analog Output Status 16 bits with Flag. g40v03 - Analog Output Status Single Precision, Floating-point with Flag.	All Range
g60v01 - Class 0 Data. g60v02 - Class 1 Data. g60v03 - Class 2 Data. g60v04 - Class 3 Data.	All Counter

Table 118: Possible Qualifier Values for All Requests Groups

altus

4.5.11.4.1.   DNP3 Client Diagnostics

The diagnostics and commands of the configured DNP3 Client protocol are stored in variables of type T_DIAG_DNP_CLIENT_REQUEST_1 which are described in the table below.

Diagnostic Variable T_DIAG_DNP _CLIENT_1 .*	Size	Description
Command Bits:		
tCommand. bStop	BIT	Disable Driver.
tCommand. bStart	BIT	Enable Driver.
tCommand. bDiag_01_Reserved	BIT	Reserved.
tCommand. bDiag_02_Reserved	BIT	Reserved.
tCommand. bDiag_03_Reserved	BIT	Reserved.
tCommand. bDiag_04_Reserved	BIT	Reserved.
tCommand. bDiag_05_Reserved	BIT	Reserved.
tCommand. bDiag_06_Reserved	BIT	Reserved.
Diagnostic Bits:		
bRunning	BOOL	DNP Client is running.
Communication Statistics:		
tStat. wRXFrames	WORD	Number of frames received.
tStat. wTXFrames	WORD	Number of frames sent.
tStat. wRequestsSent	WORD	Indicates the number of requests sent. Does not count "automatic requests".
tStat. wCommErrors	WORD	Counter of communication errors including Physical Layer, Link Layer and Transport Layer errors.
tStat. dwReserved_0	DWORD	Reserved.
tStat. dwReserved_1	DWORD	Reserved.
dwReserved_0	DWORD	Reserved.
dwReserved_1	DWORD	Reserved.

Table 119: DNP3 Client Diagnostics

The diagnostics of the DNP3 Client requests are described in the table below.

Diagnostic Variable T_DIAG_DNP_ CLIENT_REQUEST_1.*	Size	Description
Request Diagnostics:		
eConnection Status. CLOSED	BYTE	Closed communication with the IED.
eConnection Status. CONNECTED	BYTE	Operative communication with the IED.
eRequestStatus. SUCCESS	BYTE	Indication of successful request.
eRequestStatus. PROCESSING	BYTE	Indicates that a response has been received, but the command has not yet been completed. This could mean that the response is part of a multi-fragmented response and does not have the final bit set, or it could be a request such as "select operate" that requires multiple responses requests.
eRequestStatus. FAILURE	BYTE	Indicates that there was a failure in the transmission of the request.
eRequestStatus. MISMATCH	BYTE	The response to a "select" or "operate" command indicates errors.
eRequestStatus. STATUS_CODE_CHANGED	BYTE	The status of a response to a "select" command indicates errors.
eRequestStatus. IIN_ERROR	BYTE	The response to a request has IIN bits indicating error.
eRequestStatus. TIMEOUT	BYTE	Time synchronization is necessary.
eRequestStatus. CANCELED	BYTE	Indicates that a second request was made and, therefore, the first request is cancelled.
tINN. ALL_STATIONS	BIT	Indicates that a message was received from all stations.
tINN. CLASS_1_EVENTS	BIT	Indicates that there are class 1 events in the outstation queue to be reported.
tINN. CLASS_2_EVENTS	BIT	Indicates that there are class 2 events in the outstation queue to be reported.
tINN. CLASS_3_EVENTS	BIT	Indicates that there are class 3 events in the outstation queue to be reported.
tINN. NEED_TIME	BIT	Indicates that the IED wants to perform time synchronization using DNP3.
tINN. LOCAL_CONTROL	BIT	One or more points of the outstation are in local control mode.
tINN. DEVICE_TROUBLE	BIT	Outstation faulty.
tINN. DEVICE_RESTART	BIT	Outstation was restarted.
tINN. NO_FUNC_CODE_SUPPORT	BIT	Outstation does not support the function code.

altus

Diagnostic Variable T_DIAG_DNP_ CLIENT_REQUEST_1.*	Size	Description
tINN. OBJECT_UNKNOWN	BIT	Outstation does not support the command for the objects of the request.
tINN. PARAMETER_ERROR	BIT	A parameter error was detected.
tINN. EVENT_BUFFER_OVERFLOW	BIT	An overflow condition exists in the outstation queue and at least one unconfirmed event has been missed.
tINN. ALREADY_EXECUTING	BIT	The requested operation is already running.
tINN. CONFIG_CORRUPT	BIT	The outstation has a wrong configuration.
tINN. RESERVED_2	BIT	Reserved.
tINN. RESERVED_1	BIT	Reserved.
dwReserved_0	BIT	Reserved.
dwReserved_1	BIT	Reserved.

Table 120: DNP3 Client Request Diagnostics

Note:

Request Diagnostics: From all the bits available in the tIIN structure, only three of them are actually related to the request: NO_FUNC_CODE_SUPPORT, OBJECT_UNKNOWN and PARAMETER_ERROR. All the others are related to the Outstation to which the request is addressed. Thus, if the user wishes to evaluate the diagnostics related to an Outstation, the diagnostic bits of a request that is active and periodically executed should be consulted.

4.5.11.5. DNP3 Commands for IEDs

In this section, the user commands available to be sent via DNP3 to IEDs are listed.

Implemented commands:

- Function Code 03: Select
- Function Code 04: Operate
- Function Code 05: Direct Operate
- Function Code 06: Direct Operate – No Acknowledgement
- Function Code 13: Cold Restart

Due to the characteristics of the protocol, sending simultaneous commands is not supported, even if they are addressed to different points or even to different IEDs. If a command is triggered by the user application while another command is executing, the new command will wait for the end of the first one for a period of time defined in the functional block *udiCommandTimeOut*. If the time for the end of the execution of the first command is longer than *udiCommandTimeOut*, the new command will return an error message *DNP3_COMMAND_ERROR_STATUS_TIMEOUT*. That is, a command starts counting timeout time from the moment it was triggered by the user application, not just when it was sent by the DNP3 Client.

4.5.11.5.1. *Command for Digital Output*

The commands for the digital outputs are defined in the function **DNP3_BinaryCommand**. In the tables below the input and output parameters of the functional block for the digital outputs can be seen.

Parameter	Type	Description
bRequest	BOOL	When TRUE, executes the command.
dwVariableAddr	DWORD	Address of the variable that will receive the command.
eCommand	ENUM (BYTE)	Command that will be sent: DNP3_COMMAND_TYPE_SELECT (0) DNP3_COMMAND_TYPE_OPERATE (1) DNP3_COMMAND_TYPE_DIRECT_OPERATE (2) DNP3_COMMAND_TYPE_DIRECT_OPERATE_ NO_ACK (3)
eOpType	ENUM (BYTE)	DNP3_OP_TYPE_CTRL_NUL (0) DNP3_OP_TYPE_CTRL_PULSE_ON (1) DNP3_OP_TYPE_CTRL_PULSE_OFF (2) DNP3_OP_TYPE_CTRL_LATCH_ON (3) DNP3_OP_TYPE_CTRL_LATCH_OFF (4)
eTripClose	ENUM (BYTE)	DNP3_TRIP_CLOSE_CTRL_NUL (0) DNP3_TRIP_CLOSE_CTRL_PAIRED_CLOSE (64) DNP3_TRIP_CLOSE_CTRL_PAIRED_TRIP (128)
byCount	BYTE	This is the number of times the client must perform the operation.
udiOnTime	UNSIGNED DOUBLE INT	This is the duration, expressed in milliseconds, for the digital output to remain on.
udiOffTime	UNSIGNED DOUBLE INT	This is the duration, expressed in milliseconds, for the digital output to remain off.
udiCommandTimeOut	UNSIGNED DOUBLE INT	Time for the Time Out of this command (milliseconds).

Table 121: DNP3_BinaryCommand Block Input Parameters

Parameter	Type	Description
eExecStatus	ENUM (BYTE)	Execution status, indicates if a new command can be sent, values are as follows: DNP3_COMMAND_EXEC_STATUS_DONE (0) DNP3_COMMAND_EXEC_STATUS_RUNNING (1)
eErrorStatus	ENUM (BYTE)	Indicates whether the last command was successful, the values are as follows: DNP3_COMMAND_ERROR_STATUS_NO_ERROR (0) DNP3_COMMAND_ERROR_STATUS_TIMEOUT (1) DNP3_COMMAND_ERROR_STATUS_NO_SELECT (2) DNP3_COMMAND_ERROR_STATUS_FORMAT_ERROR (3) DNP3_COMMAND_ERROR_STATUS_NOT_SUPPORTED (4) DNP3_COMMAND_ERROR_STATUS_ALREADY_ACTIVE (5) DNP3_COMMAND_ERROR_STATUS_HARDWARE_ERROR (6) DNP3_COMMAND_ERROR_STATUS_UNDEFINED (7) DNP3_COMMAND_ERROR_STATUS_INVALID_ADDRESS (8) DNP3_COMMAND_ERROR_STATUS_WRONG_TIMEOUT (9) DNP3_COMMAND_ERROR_STATUS_INVALID_VALUE (10) DNP3_COMMAND_ERROR_STATUS_UNINITIALIZED (11)

Table 122: DNP3_BinaryCommand Block Output Parameters

Note:

eTripClose: The trip/close commands are unique for digital output modules that support this functionality. For example, Hadron Xtorm Series HX2320 module. If the user is using Hadron Xtorm Series mixed expansion bus with Nexto Series, and uses a digital output module that does not support this feature, the CPU returns the following error code: "DNP3_COMMAND_ERROR_STATUS_NOT_SUPPORTED".

4.5.11.5.2. Command for o Analog Outputs

The commands for the analog outputs are defined in two functions: **DNP3_AnalogCommandInt** and **DNP3 _Analog-CommandReal**. In the tables below, we can see the input and output parameters for each of the function blocks for the analog outputs.

Function Block Name: **DNP3_AnalogCommandInt**, used for numbers of integer type (16 or 32 bits). Used for the G41V1 and G41V2 groups.

Parameter	Type	Description
bRequest	BOOL	When TRUE, executes the command.
dwVariableAddr	DWORD	Command variable address to send.
eCommand	ENUM (BYTE)	Command that will be sent: DNP3_COMMAND_TYPE_SELECT (0) DNP3_COMMAND_TYPE_OPERATE (1) DNP3_COMMAND_TYPE_DIRECT_OPERATE (2) DNP3_COMMAND_TYPE_DIRECT_OPERATE_ NO_ACK (3)
dValue	DOUBLE INT	Analog value to be written: INT or DINT.
udiCommandTimeOut	UNSIGNED DOUBLE INT	Time for the Time Out of this command (milliseconds).

Table 123: DNP3_AnalogCommandInt Block Input Parameters

Parameter	Type	Description
eExecStatus	ENUM (BYTE)	Execution status, indicates if a new command can be sent, values are as follows: DNP3_COMMAND_EXEC_STATUS_DONE (0) DNP3_COMMAND_EXEC_STATUS_RUNNING (1)
eErrorStatus	ENUM (BYTE)	Indicates whether the last command was successful, the values are as follows: DNP3_COMMAND_ERROR_STATUS_NO_ERROR (0) DNP3_COMMAND_ERROR_STATUS_TIMEOUT (1) DNP3_COMMAND_ERROR_STATUS_NO_SELECT (2) DNP3_COMMAND_ERROR_STATUS_FORMAT_ERROR (3) DNP3_COMMAND_ERROR_STATUS_NOT_SUPPORTED (4) DNP3_COMMAND_ERROR_STATUS_ALREADY_ACTIVE (5) DNP3_COMMAND_ERROR_STATUS_HARDWARE_ERROR (6) DNP3_COMMAND_ERROR_STATUS_UNDEFINED (7) DNP3_COMMAND_ERROR_STATUS_INVALID_ADDRESS (8) DNP3_COMMAND_ERROR_STATUS_WRONG_TIMEOUT (9) DNP3_COMMAND_ERROR_STATUS_INVALID_VALUE (10) DNP3_COMMAND_ERROR_STATUS_UNINITIALIZED (11)

Table 124: DNP3_AnalogCommandInt Block Output Parameters

Function Block Name: **DNP3_AnalogCommandReal**, used for numbers of type real (32 bits). Used for the G41V3 group.

altus

Parameter	Type	Decription
bRequest	BOOL	When TRUE, executes the command.
dwVariableAddr	DWORD	Command variable address to send.
eCommand	ENUM (BYTE)	Command that will be sent: DNP3_COMMAND_TYPE_SELECT (0) DNP3_COMMAND_TYPE_OPERATE (1) DNP3_COMMAND_TYPE_DIRECT_OPERATE (2) DNP3_COMMAND_TYPE_DIRECT_OPERATE_ NO_ACK (3)
rValue	REAL	Analog value to be written: REAL.
udiCommandTimeOut	UNSIGNED DOUBLE INT	Time for the Time Out of this command (milliseconds).

Table 125: DNP3_AnalogCommandReal Block Input Parameters

Parameter	Type	Description
eExecStatus	ENUM (BYTE)	Execution status, indicates if a new command can be sent, values are as follows: DNP3_COMMAND_EXEC_STATUS_DONE (0) DNP3_COMMAND_EXEC_STATUS_RUNNING (1)
eErrorStatus	ENUM (BYTE)	Indicates whether the last command was successful, the values are as follows: DNP3_COMMAND_ERROR_STATUS_NO_ERROR (0) DNP3_COMMAND_ERROR_STATUS_TIMEOUT (1) DNP3_COMMAND_ERROR_STATUS_NO_SELECT (2) DNP3_COMMAND_ERROR_STATUS_FORMAT_ERROR (3) DNP3_COMMAND_ERROR_STATUS_NOT_SUPPORTED (4) DNP3_COMMAND_ERROR_STATUS_ALREADY_ACTIVE (5) DNP3_COMMAND_ERROR_STATUS_HARDWARE_ERROR (6) DNP3_COMMAND_ERROR_STATUS_UNDEFINED (7) DNP3_COMMAND_ERROR_STATUS_INVALID_ADDRESS (8) DNP3_COMMAND_ERROR_STATUS_WRONG_TIMEOUT (9) DNP3_COMMAND_ERROR_STATUS_INVALID_VALUE (10) DNP3_COMMAND_ERROR_STATUS_UNINITIALIZED (11)

Table 126: DNP3_AnalogCommandReal Block Output Parameters

4.5.11.5.3. Cold Restart Command

Function Block Name: **DNP3_ColdCommand**.

Parameter	Type	Description
bRequest	BOOL	When TRUE, executes the command.
dwVariableAddr	DWORD	Command variable address to send.
udiCommandTimeOut	UNSIGNED DOUBLE INT	Time for the Time Out of this command (milliseconds).

Table 127: DNP3_ColdCommand Block Input Parameters

altus

Parameter	Type	Description
eExecStatus	ENUM (BYTE)	Execution status, indicates if a new command can be sent, values are as follows: DNP3_COMMAND_EXEC_STATUS_DONE (0) DNP3_COMMAND_EXEC_STATUS_RUNNING (1)
eErrorStatus	ENUM (BYTE)	Indicates whether the last command was successful, the values are as follows: DNP3_COMMAND_ERROR_STATUS_NO_ERROR (0) DNP3_COMMAND_ERROR_STATUS_TIMEOUT (1) DNP3_COMMAND_ERROR_STATUS_NO_SELECT (2) DNP3_COMMAND_ERROR_STATUS_FORMAT_ERROR (3) DNP3_COMMAND_ERROR_STATUS_NOT_SUPPORTED (4) DNP3_COMMAND_ERROR_STATUS_ALREADY_ACTIVE (5) DNP3_COMMAND_ERROR_STATUS_HARDWARE_ERROR (6) DNP3_COMMAND_ERROR_STATUS_UNDEFINED (7) DNP3_COMMAND_ERROR_STATUS_INVALID_ADDRESS (8) DNP3_COMMAND_ERROR_STATUS_WRONG_TIMEOUT (9) DNP3_COMMAND_ERROR_STATUS_INVALID_VALUE (10) DNP3_COMMAND_ERROR_STATUS_UNINITIALIZED (11)

Table 128: DNP3_ColdCommand Block Output Parameters

Application Example:

Below is an example application using the ST language, in which the user sends the command *Select Before Operate* to an IED via DNP3.

```
PROGRAM UserPrg
VAR
  iDNPClientAO:    INT;
  iDNPClientAOCmd:    LibDNP3.DNP3_AnalogCommandInt;
END_VAR

iDNPClientAOCmd.dwVariableAddr:= ADR(iDNPClientAO);
iDNPClientAOCmd();
IF  iDNPClientAOCmd.eExecStatus = LibDNP3.DNP3_COMMAND_EXEC_STATUS_DONE AND
  iDNPClientAOCmd.bRequest = TRUE THEN
    IF iDNPClientAOCmd.eCommand = LibDNP3.DNP3_COMMAND_TYPE_SELECT AND
       iDNPClientAOCmd.eErrorStatus = LibDNP3.DNP3_COMMAND_ERROR_STATUS_NO_ERROR
    THEN
            iDNPClientAOCmd.eCommand:= LibDNP3.DNP3_COMMAND_TYPE_OPERATE;
    ELSE
            iDNPClientAOCmd.bRequest:= FALSE;
    END_IF
END_IF
```

altus

4.5.11.6. Status Codes for DNP Commands

DNP3 commands return error status (*andErrorStatus*) when executed. The table below shows the list and description of possible values for the error status.

Code	Status	Description
0	DNP3_COMMAND_ERROR_STATUS_NO_ERROR	Request accepted, started or queued.
1	DNP3_COMMAND_ERROR_STATUS_TIMEOUT	Request not accepted, because the message was received after the select command time out.
2	DNP3_COMMAND_ERROR_STATUS_NO_SELECT	Request not accepted, because the select command was not sent previously.
3	DNP3_COMMAND_ERROR_STATUS_FORMAT_ERROR	Request not accepted, because there were formatting errors in the control request (select, operate or direct operate).
4	DNP3_COMMAND_ERROR_STATUS_NOT_SUPPORTED	Request not accepted, because a control operation is not supported by this point.
5	DNP3_COMMAND_ERROR_STATUS_ALREADY_ACTIVE	Request not accepted, because the control queue is already full or the point is already active.
6	DNP3_COMMAND_ERROR_STATUS_HARDWARE_ERROR	Request not accepted, because of hardware problems.
7	DNP3_COMMAND_ERROR_STATUS_UNDEFINED	Request not accepted, because of some other undefined reason.
8	DNP3_COMMAND_ERROR_STATUS_INVALID_ADDRESS	Request not accepted, because the variable address is invalid.
9	DNP3_COMMAND_ERROR_STATUS_WRONG_TIMEOUT	Request not accepted, because time out value is invalid.
10	DNP3_COMMAND_ERROR_STATUS_INVALID_VALUE	Request not accepted, because the value is not valid.
11	DNP3_COMMAND_ERROR_STATUS_UNINITIALIZED	Request not accepted, because the functional block was not started correctly.

Table 129: Status Codes for DNP3 Commands

4.5.12. DNP3 Ethernet Server

This protocol is available for the Hadron Xtorm Series CPU on its Ethernet channels. By selecting this option in MasterTool Xtorm, the CPU becomes the DNP3 communication server, allowing connection with up to five DNP3 client devices. For each client, the driver has a unique event queue with the following characteristics:

- Size: 4500 events.
- Overflow policy: keep the most recent one.

altus

To configure this protocol, the following steps must be performed:

1. Add the DNP3 Server protocol instance to one of the available Ethernet channels (NET 1 ... NET 6). To perform this procedure, see the section Inserting a Protocol Instance.
2. Configure the Ethernet interface. To do this, see the section Ethernet Interfaces Configuration.
3. Configure the DNP3 Server protocol general parameters, with Port or IP connection mode, and the TCP port number when the selected connection mode is IP mode.
4. Add and configure devices by setting appropriate parameters.
5. Add and configure DNP3 mappings by specifying the variable name, static group variation, event group variation, index, size, range, event class, deadband, and deadband type.
6. Configure the parameters of the link layer, specifying the addressing and confirmation messages.
7. Configure the application layer parameters, specifying response fragmentation, confirmation messages, integrity polling, synchronism, commands, and event queue.
8. Configure the parameters of the unsolicited messages, specifying the control parameters and trigger conditions.

The descriptions of each configuration are listed below in this section.

4.5.12.1. Configuration of the DNP3 Server Mappings

The settings of the General Parameters of a DNP3 Server, as shown in the figure below, follow the parameters described in the following table.
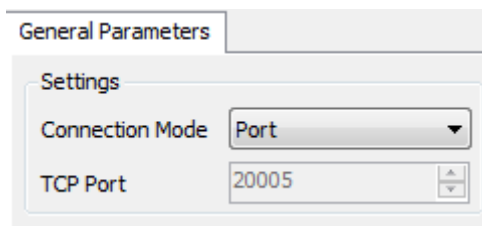
Figure 85: DNP3 Server General Parameter Screen

Configuration	Description	Default	Options
Connection Mode	Sets the connection mode with the Connected Client modules. The available connection modes are "IP" or "Port".	Port	Port IP
TCP Port	Sets the TCP port number of the RTU to be used for communication with the Connected Client modules, if "IP" is selected in the "Connection Mode" field.	20005	1 to 65535

Table 130: DNP3 Server General Parameter Configuration

The configurations of the DNP3 Server relations, seen in the figure below, follow the parameters described in the following table.

Figure 86: DNP3 Server Mapping Configuration Screen

Configuration	Description	Default	Options
Value Variable	Symbolic variable name.	-	Name of a variable declared in a program or GVL.
Static Group and Default Variation	Static data variation configuration.	-	g10v01 - Binary Output Packed Format. g10v02 - Binary Output Status with Flags. g01v01 - Binary Input Packed Format. g01v02 - Binary Input With Flags. g20v01 - Counter 32 bits with flags. g20v02 - Counter 16 bits with flags. g20v05 - Counter 32 bits without flags. g20v06 - Counter 16 bits without flags. g21v01 - Frozen Counter 32 bits with Flag. g21v02 - Frozen Counter 16 bits with Flag. g21v09 - Frozen Counter 32 bits without Flag. g21v10 - Frozen Counter 16 bits without Flag. g30v01 - Analog Input 32 bits with Flag. g30v02 - Analog Input 16 bits with Flag. g30v03 - Analog Input 32 bits without Flag. g30v04 - Analog Input 16 bits without Flag. g30v05 - Analog Input Single Precision, Floating-point with Flag. g03v01 – Double-bit Binary Input Packed Format. g03v02 – Double-bit Binary Input With Flags. g40v01 – Analog Output Status 32 bits with Flag. g40v02 – Analog Output Status 16 bits with Flag. g40v03 – Analog Output Status Single Precision, Floating-point with Flag.
Event Group and Default Variation	Variation configuration for events.	-	g22v01 – Counter Event 32 bits with Flag. g22v02 – Counter Event 16 bits with Flag. g22v05 – Counter Event 32 bits with Flag and Time. g22v06 – Counter Event 16 bits with Flag and Time. g23v01 – Frozen Counter Event 32 bits with Flag. g23v02 – Frozen Counter Event 16 bits with Flag. g23v05 - Frozen Counter Event 32 bits with Flag and Time. g23v06 - Frozen Counter Event 16 bits with Flag and Time. g02v01 – Binary Input Event without Time. g02v02 – Binary Input Event with Absolute Time. g02v03 – Binary Input Event with Relative Time. g32v01 - Analog Input Event 32 bits without Time. g32v02 - Analog Input Event 16 bits without Time. g32v03 - Analog Input Event 32 bits with Time. g32v04 - Analog Input Event 16 bits with Time. g32v05 - Analog Input Event Single Precision, Floating-point without Time. g04v01 – Double-bit Binary Input Event without Time. g04v02 – Double-bit Binary Input Event with Absolute Time. g04v03 – Double-bit Binary Input Event with Relative Time. None

Configuration	Description	Default	Options
Event Class	Event Class Configuration.	-	Class 0 Class 1 Class 2 Class 3
Index	Index of the first DNP3 mapping point.	-	0 to 65535
Size	DNP3 data size.	-	- (automatically calculated)
Range	Address range of the data set.	-	- (automatically calculated)
Dead Band Variable	Name of the symbolic variable that will contain the deadband data. This field applies only to analog input variable mapping.	-	Name of a variable declared in a program or GVL.
Dead Band Type	Defines the type of Deadband to be used in the mapping.	Disabled	Absolute Disabled Integrated

Table 131: Configuration of the DNP3 Server Mappings

Notes:

Value Variable: Name of the symbolic variable to be mapped. When a reading command is sent, the return sent in response will be stored in this variable. In case of a writing command, the written value is also stored in this variable. The variable may be simple, array or array element and may be in structures.

Static Group and Default Variation: The field is used to configure the variation of static data that will be returned if the DNP3 client does not specify a variation in the request data, or in the case of unsolicited messages.

Event Group and Default Variation: The field is used to set the range for the events that will be returned if the DNP3 client does not specify a variation in the request data, or in the case of unsolicited messages.

Event Class: The field is used to set in which class the events of this mapping will be stored. For points configured as class 0, no events will be reported. For groups 10 and 40, the value of this field is fixed at zero, since it deals with digital and analog outputs respectively.

Index: The field is used to set the index of the first mapping point of the DNP3 protocol.

Size: This field shows the maximum amount of data that the DNP3 relation can access, starting at the starting address. Therefore, to read a continuous address range, it is necessary that all addresses are declared in a single relation. This field varies according to the type of DNP3 data configured, whether single variable or array, and is calculated automatically by the configuration tool from the declaration of the *Value Variable*.

Range: This field shows the user the range of memory addresses used by the DNP3 relation, calculated automatically by the configuration tool from the *Index* and *Size* fields.

Dead Band Variable: This field is used to specify a variable that will contain the DNP3 relation Dead Band data. It must have the same type and size as the variable mapped in the Value Variable column. New values of the deadband variable will be considered only when the analog input variable changes value.

Dead Band Type: The available Dead Band configuration types are as follows:

Function Type	Configurations	Description
Dead Band Type	Disabled	In this option, any value variation in a point of the group, however small, generates an event for this point.
	Absolute	In this option, if the module of the variation of the value of a point of the group is greater than the value configured by the variable in the "Deadband" field, an event is generated for this point.
	Integrated	In this option, if the module of the integral of the variation of the value of a point of the group is greater than the value configured by the variable in the "Deadband" field, an event is generated for this point. The integration interval is one second.

Table 132: Deadband Types in DNP3 Server Parameters

ATTENTION

The dead band set to variables mapped in DNP3 Server is not applied when these points are derived from an IED (through the DNP3 Client driver, IEC 61850 GOOSE Subscriber, etc...), because in these case the event detection occurs in the IED itself

4.5.12.2. Configuration of the DNP3 Server Link Layer

The DNP3 Server's link layer parameter configurations, shown in the figure below, follow the parameters described in the following table.
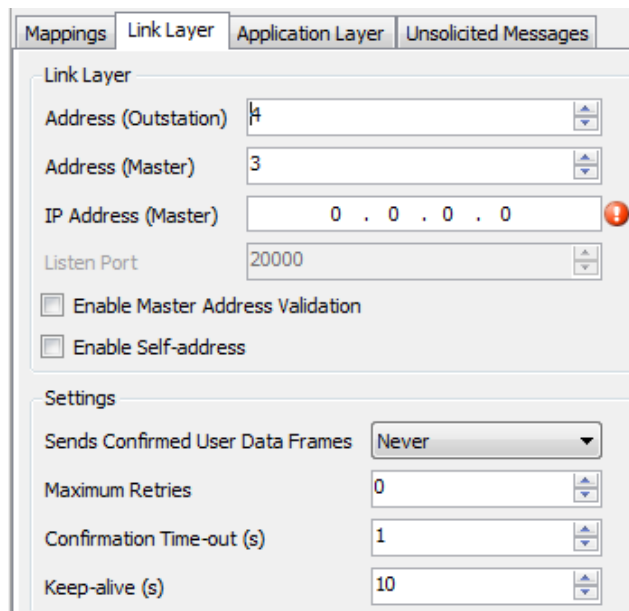
Figure 87: DNP3 Server Link Layer Configuration Screen

Configuration	Description	Default	Options
Address (Outstation)	DNP3 address of this server.	4	0 to 65519
Address (Master)	DNP3 address of the connected client.	3	0 to 65519
IP Address (Master)	IP of the connected client, used when the client connection is IP.	0.0.0.0	1.0.0.1 to 223.255.255.254
Listen Port	Listen port address for client connection. Used when the client connection is not IP.	20000	1 to 65535
Enable Master Address Validation	Option to Enable/Disable the validation of the Origin address.	Disabled	Enabled Disabled
Enable Self-address	Option to Enable/Disable Auto Addressing.	Disabled	Enabled Disabled
Sends Confirmed User Data Frames	Selects the confirmation mode for messages sent to the client.	Never	Never Always Sometimes
Maximum Retries	Maximum message retransmission attempts.	3	0 to 10
Confirmation Time-out (s)	Waiting time to send new attempt.	3	1 to 86400
Keep Alive (s)	Waiting time to send link test messages after receiving the DNP message.	10	1 to 86400

Table 133: DNP3 Server Link Layer Configuration

Notes:

Address (Outstation): The field is used to set the DNP3 address of this server.

Address (Master): The field is used to set the DNP3 address of the client.

IP Address (Master): The field is used to set the IP address of the connected client, used when the client connection is over IP.

Listen Port: Listen port address for client connection. Used when the client connection is not through IP addressing.

Enable Master Address Validation: If this option is checked, the DNP3 Server will only accept messages whose source and destination addresses match the values configured respectively in the "Source Address" and "Destination Address" fields. In other words, the "Source Address" field of the message must match the value in the "Destination Address" field configured for the Client. And the "Destination Address" field of the message must match the value in the "Source Address" field configured for the client.

Enable Self-address: Enables receiving messages with address 65532 from the DNP3 Client.

Sends Confirmed User Data Frames: The available link layer confirmation modes are as follows:

Function Type	Configurations	Description
Link Layer Confirmation Mode	Never	Never requests message confirmation.
	Always	Always requests message confirmation.
	Sometimes	Requests confirmation only for multifragmented messages.

Table 134: Tipos de Modo de Confirmação da Camada de Enlace nos Parâmetros do DNP3 Servidor

Maximum Retries: Number of times the DNP3 Server will retransmit the message if it does not receive confirmation from the client.

Confirmation Time-Out (s): Waiting time to send a new attempt.

Keep Alive (s): Time interval for sending link test messages ("TCP keep-alive" or "serial link status") after last received DNP3 message. If no response is received for the link test message, the current connection will be terminated. This test mechanism must be set up correctly, as it is through this mechanism that connection problems are identified.

altus

4.5.12.3.   Configuration of the of DNP3 Server Application Layer

The configurations of the DNP3 Server relations, visualized in the figure below, follow the parameters described in the following table.
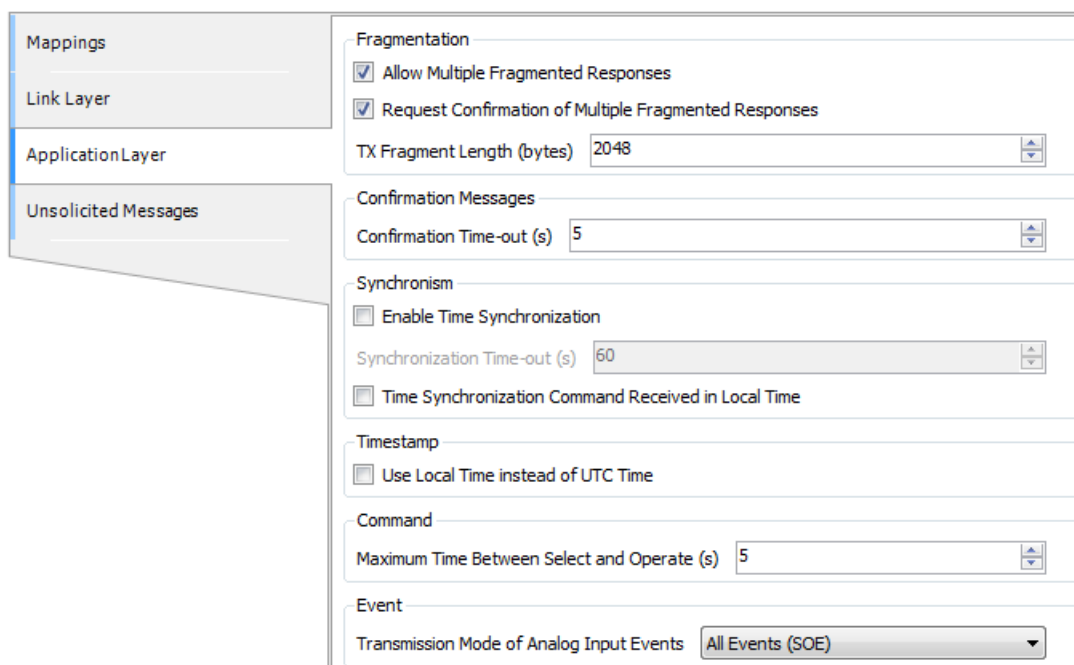
Figure 88: DNP3 Server Application Layer Configuration Screen

Configuration	Description	Default	Opções
Allow Multiple Fragmented Responses	Option to Enable/Disable Response Transmission with Multiple Fragments.	Enabled	Enabled Disabled
Request Confirmation of Multiple Fragmented Responses	Option to Enable/Disable Response Confirmation Request with Multiple Fragments.	Enabled	Enabled Disabled
TX Fragmented Length (bytes)	Maximum size of the fragments.	2048	249 to 2948
Confirmation Time-out (s)	Determines the maximum waiting time for the confirmation messages.	5	1 to 86400
Enable Time Synchronization	Option to Enable/Disable Time Synchronization Request.	Disabled	Enabled Disabled
Time Synchronization Command Received in Local Time	Option to Enable/Disable the treatment of the synchronization command in local time.	Disabled	Enabled Disabled
Synchronization Time-out (s)	Sets the time that the local clock will remain valid after it has received a time synchronization command.	60	1 to 86400
Use Local Time instead of UTC Time	Option to Enable/Disable the time stamp in local time for events.	Disabled	Enabled Disabled
Maximum Time Between Selected and Operate (s)	Maximum time that the select command will remain active waiting for the Operate command.	5	1 to 86400

altus

Configuration	Description	Default	Opções
Transmission Mode of Analog Inputs Events	Selects the mode of transmission of analog input events.	All Events (SOE)	All Events (SOE) Most Recent Events

Table 135: DNP3 Server Application Layer Configuration

Notes:

Allow Multiple Fragmented Responses: If this option is checked, the DNP3 Server will enable the splitting of an application message into smaller messages and send them one by one.

Request Confirmation of Multiple Fragmented Responses: If this option is checked, the DNP3 server will request the client a confirmation that the fragmented message was received.

TX Fragmented Length (bytes): Maximum length of the fragmented messages.

Confirmation Time-out (s): The confirmation message time-out is the time the server will wait for a confirmation that the message was received by the client.

Enable Time Synchronization: If this option is checked, the DNP3 Server is enabled to request time synchronization from the client. This request is made in the first message transmitted to the client after the time set in the "Synchronization Time-Out" field has elapsed.

Synchronization Time-out (s): This field is used to set the time interval for time synchronization requests.

Maximum Time Between Selected and Operate (s): The period in which the selection command will remain active (starts counting when the selection command is received).

Transmission Mode of Analog Input Events: The available transmission modes of the analog input events are as follows:

Function Type	Configurations	Description
Transmission Mode of Analog Inputs Events	All Events(SOE)	All analog events generated are sent.
	Most Recent Events	Only the most recent event of each analog point is sent.

Table 136: Configuration of Transmission Modes of Analog Input Events in DNP3 Server

> **ATTENTION**
>
> When the time synchronization option is checked on more than one outstation ("Enable Synchronization" parameter) the times received on the different outstations will be overwritten on the system clock in a short period of time, which can cause undesirable behavior due to delays in the message broadcasting and system load.

4.5.12.4. Configuration of the DNP3 Server Unrequested Messages

The configurations of the DNP3 Server relations, visualized in the figure below, follow the parameters described in the following table.

altus
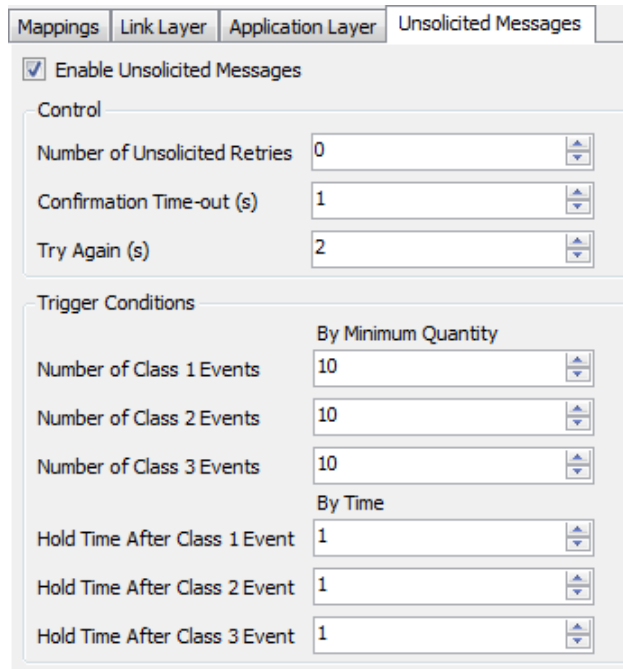
Figure 89: DNP3 Server Unsolicited Messages Configuration Screen

Configuration	Description	Default	Options
Enable Unsolicited Messages	Option to Enable/Disable unsolicited messages.	Enabled	Enabled Disabled
Number of Unsolicited Retries	Number of attempts to transmit the message.	3	0 to 65535
Confirmation Time-out (s)	Interval to wait after a time out.	5	1 to 86400
Try Again (s)	Interval to wait after reaching the maximum number of attempts.	30	1 to 86400
Number of Class 1 Events	Minimum number of events in Class 1 to trigger an unsolicited message.	10	1 to 255
Number of Class 2 Events	Minimum number of events in Class 2 to trigger an unsolicited message.	10	1 to 255
Number of Class 3 Events	Minimum number of events in Class 3 to trigger an unsolicited message.	10	1 to 255
Hold Time After Class 1 Event	Maximum time that an event can be stored in class 1 without being sent by an unsolicited message.	1	1 to 86400
Hold Time After Class 2 Event	Maximum time that an event can be stored in class 2 without being sent by an unsolicited message.	1	1 to 86400
Hold Time After Class 3 Event	Maximum time that an event can be stored in class 3 without being sent by an unsolicited message.	1	1 to 86400

Table 137: DNP3 Server Unsolicited Messages Configuration

Notes:

Enable Unsolicited Messages: If this option is checked, the DNP3 Server will allow Unsolicited Messages.

Number of Unsolicited Retries: Number of times the DNP3 Server will try to retransmit the message.

Confirmation Time-out(s): Specifies the time to be waited after a time-out occurs for the unsolicited message that was transmitted. After this interval has elapsed the message will be retransmitted. Then the retransmission time-out will be given by the parameters : **"Time-out de Mensagens de Confirmação" + "Time-Out"**.

Try Again (s): After the maximum number of attempts to transmit the message is reached (see *"Number of Unsolicited Retries"* parameter), the remaining retransmissions will be based on this time parameter. Therefore, the retransmission time will now be given by the parameters: **"Time-out de Mensagens de Confirmação" + "Tentar Novamente"**.

Number of Class 1 Events: Minimum number of events in class 1 to trigger the unsolicited message.

Number of Class 2 Events: Minimum number of events in class 2 to trigger the unsolicited message.

Number of Class 3 Events: Minimum number of events in class 3 to trigger the unsolicited message.

Hold Time After Class 1 Event: Maximum time that a class 1 event can be stored without being sent by an unsolicited message.

Hold Time After Class 2 Event: Maximum time that a class 2 event can be stored without being sent by an unsolicited message.

Hold Time After Class 3 Event: Maximum time that a class 3 event can be stored without being sent by an unsolicited message.

4.5.12.5. DNP3 Server Diagnostics

The diagnostics and commands of the configured DNP3 Server protocol, are stored in variables of type *T_DIAG_DNP_ SERVER_1* which are described in the table below.

Diagnostic Variable T_DIAG_DNP_SERVER_1.*	Size	Description
Command bits, automatically reset:		
tCommand. bStop	BIT	Disable Driver.
tCommand. bStart	BIT	Enable Driver.
tCommand. bDiag_01_Reserved	BIT	Reserved.
tCommand. bDiag_02_Reserved	BIT	Reserved.
tCommand. bDiag_03_Reserved	BIT	Reserved.
tCommand. bDiag_04_Reserved	BIT	Reserved.
tCommand. bDiag_05_Reserved	BIT	Reserved.
tCommand. bDiag_06_Reserved	BIT	Reserved.
Diagnostics:		
tClient_X. bRunning	BOOL	DNP Server is running.
tClient_X. eConnectionStatus. CLOSED	ENUM (BYTE)	Communication channel is closed. Server will not accept connection request. ENUM value (0).
tClient_X. eConnectionStatus. LISTENING	ENUM (BYTE)	Server is listening on the configured port, and there are no connected clients. ENUM value (1).

altus

Diagnostic Variable T_DIAG_DNP_SERVER_1.*	Size	Description
tClient_X. eConnectionStatus. CONNECTED	ENUM (BYTE)	Client connected. ENUM value (2).
tClient_X. tQueueDiags. bOverflow	BOOL	The client's queue is overflowing.
tClient_X. tQueueDiags. wSIZE	WORD	Configured queue size.
tClient_X. tQueueDiags. wUSAGE	WORD	Number of events in the queue.
tClient_X. tQueueDiags. dwUnsolEna	DWORD	Mask that indicates for which classes are enabled the unsolicited messages.
tClient_X. tQueueDiags. dwReserved_1	DWORD	Reserved.
tClient_X. tStats. wRXFrames	WORD	Number of frames received.
tClient_X. tStats. wTXFrames	WORD	Number of frames sent.
tClient_X. tStats. wCommErrors	WORD	Communication error counter including errors in the Physical Layer, Link Layer and Transport Layer.
tClient_X. tStats. dwReserved_0	DWORD	Reserved.
tClient_X. tStats. dwReserved_1	DWORD	Reserved.

Table 138: DNP3 Server Diagnostics

Note:

dwUnsolEna: The bits 0, 1 and 2 correspond respectively to the bits indicating the enabling status of unsolicited messages of classes 1, 2 and 3. When turned on they indicate that the sending of unsolicited messages has been enabled for the corresponding class.

altus

4.5.12.6. Association of Objects with IEC 60870-5-104 Protocol

The table below shows allowed associations between DNP3 and IEC 60870-5-104 protocols.

DNP3 Group and Variation	Type of Object IEC 60870-5-104
g01v01 - Binary Input Packed Format. g01v02 - Binary Input With Flags.	Single Point Information (M_SP_NA)
g03v01 - Double-bit Binary Input Packed Format. g03v02 - Double-bit Binary Input With Flags.	Double Point Information (M_DP_NA)
g10v01 - Binary Output Packed Format. g10v02 - Binary Output Status with Flags.	Single Command (C_SC_NA) Double Command (C_DC_NA)
g30v02 - Analog Input 16 bits with Flag. g30v04 - Analog Input 16 bits without Flag.	Measured Value, normalized value (M_ME_NA) Measured Value, scaled value (M_ME_NB)
g30v01 - Analog Input 32 bits with Flag. g30v03 - Analog Input 32 bits without Flag.	Measured Value, short floating value (M_ME_NC)
g30v05 - Analog Input Single Precision, Floating-point with Flag	Measured Value, short floating value (M_ME_NC)
g40v02 - Analog Output Status 16 bits with Flag.	Setting Point Command, normalized Value (C_SE_NA) Setting Point Command, scaled Value (C_SE_NA)
g40v01 - Analog Output Status 32 bits with Flag.	-
g40v03 - Analog Output Status Single Precision, Floating-point with Flag.	Setting Point Command, short floating point Value (C_SE_NC)

Table 139: Association of DNP3 Object Types with Their Correspondents in the IEC 60870-5-104

> **ATTENTION**
>
> g20 (Counters) e g21 (Frozen Counters): it is not allowed to associate a variable used as a counter, simultaneously in the Protocols DNP3 and IEC 60870-5-104.
> g40v01 (Analog Output Status 32 bits with Flag): there is no equivalent object for IEC 60870-5-104 protocol.

4.5.12.7. Cold Restart Command

The Cold Restart command forces a reboot of the remote. The internal queue events are not lost, that is, the values of the retentive and persistent variables are not changed. However, the events of the drivers queue (due to overflow), will be lost. Once the remote is reset, all settings made during the Runtime will be lost, and the system will return to the original settings made through the MasterTool software.

4.5.12.8. Commands for Output Points and Counters

The commands received by the DNP3 Server driver may have the following destinations:

- Internal Points
- Output Module (e.g. HX2320)
- Driver DNP3 Client
- Interception (CommandReceiver function block)

The CPU only supports commands to freeze counters in a certain range of points, if they are of internal type. If this is not the case (points redirected to an IED connected to a DNP3 Client driver for example), the command will not be accepted. In such cases, freeze each point individually.

4.5.12.8.1.   Internal Points

Internal points are those ones represented by variables which are not associated to any of the others destinations described above. For this type of point, the CPU only supports commands for simple points. If the DNP3 server receives a trip/close command to an internal point, it returns the error code 0x04 (not supported), unless the command interceptor is used.

4.5.12.8.2.   Output Module

If the variable mapped to the DNP3 output point is associated to an output card (e.g. HX2320), the command will be redirected and executed by the card itself. In case of trip/close commands, this redirection will work only if the variable is the of DBP type. If a variable of type BOOL is used, the DNP3 server returns the error code 0x03 (wrong format) for pulsed commands, as well as when a Latched command is directed to a BPD variable mapped in the output card. With the interceptation and treatment of these commands by the application it can be avoided the return of failure messages.

4.5.12.8.3.   DNP3 Client Driver

If the variable is associated to an IED point in a DNP3 Client driver, the command is redirected and sent to be executed by the IED itself and the value of the variable stored in the CPU memory is not updated. In case of trip/close commands, typically the variable must be of type DBP. However, the redirection works in the same way even if a variable of type BOOL is used.

Notice that due to protocol features, the DNP3 client driver is able to execute only one command at a time. That is, while a command to a particular point is not finalized (response or time-out), you cannot trigger a new command, even to a different point. Thus, if a command is redirected from the DNP3 Server to an IED associated to a busy DNP3 Client driver, it returns the error code 0x05 (already active).

4.5.12.8.4.   IEC 60870-5-104 Client Driver

If the variable is associated with an IED point in an IEC 60870-5-104 Client driver, the command will be redirected and sent to be executed by the IED itself. In the case of trip/close commands, typically the variable should be of type DBP, however redirection will work the same way even if a variable of type BOOL is used.

4.5.12.8.5.   Commands Interception

The interception of commands received by the IEC 61850 Server driver is accomplished through the *CommandReceive* function block as described in the Interception of Commands from Control Center section. This feature allows the interception of selection and operation commands for the following CDCs (Common Data Classes): SPC, DPC, INC, BSC, ISC.

Below is an example of an application using the ST language showing the interception of a command to the *BlkOpn* object of an XCBR Logical Node, responding *"SUCCESS"* to the IEC 61850 Client.

4.5.13.   IEC 60870-5-104 – Data Types

The table below shows the variable type supported by the HX3040 for each of the IEC 60870-5-104 protocol data types.

Type of Object	Types of IEC Variables
Single Point Information (M_SP_NA)	BOOL
	BIT
Double Point Information (M_DP_NA)	DBP
Step Position Information (M_ST_NA)	USINT
Measured Value, normalized value (M_ME_NA)	INT
Measured Value, scaled value (M_ME_NB)	INT
Measured Value, short floating point value (M_ME_NC)	INT
	UINT
	DINT
	UDINT
	REAL
Integrated Totals (M_IT_NA)	INT
	DINT

Type of Object	Types of IEC Variables
Bitstring Information (M_BO_NA)	DWORD
Single Command (C_SC_NA)	BOOL
	BIT
Double Command (C_DC_NA)	DBP
Regulating Step Command (C_RC_NA)	DBP
Setting Point Command, normalized Value (C_SE_NA)	INT
Setting Point Command, scaled Value (C_SE_NB)	INT
Setting Point Command, short floating point Value (C_SE_NC)	REAL
Bitstring Command (C_BO_NA)	DWORD

Table 140: Declaration of Variables for IEC 60870-5-104

Note:

Regulating Step Command: The states *Lower* and *Higher* of the C_RC_NA object are associated respectively with the states *OFF* and *ON* of the DBP internal type.

Step Position Information: As defined in item 7.3.1.5 of the IEC 60870-5-101 standard, this 8-bit variable is composed of two fields: Value (defined by the 7 least significant bits of the variable) and Transient (defined as the most significant bit, which indicates when the measured device is transitioning).

Below is an example of code for manipulating the fields in a variable of type USINT. This code does not consist if the input value is within the range (between -64 and + 63), so this consistency is up to the user.

```
PROGRAM UserPrg
VAR
  usiVTI: USINT;  // Value transient indication (client mapping)
  siValue:  SINT;   // Value to convert to VTI
  bTransient: BOOL;   // Transient to be converted to VTI
END_VAR

usiVTI:= SINT_TO_USINT(siValue) AND 16#3F;
IF siValue < 0 THEN
  usiVTI:= usiVTI OR 16#40;
END_IF
IF bTransient THEN
  usiVTI:= usiVTI OR 16#80;
END_IF
```

4.5.14. IEC 60870-5-104 Client

This protocol is available for the Hadron Xtorm Series CPU in its Ethernet interfaces (NET1 .. NET6). By selecting this option in MasterTool Xtorm, the CPU becomes a client of the IEC 60870-5-104 communication. The maximum limit of IEC 60870-5-104 Client driver instantiation depends on how many protocol drivers are instantiated and supported by the HX3040 UCP.

The IEC 60870-5-104 Client protocol is active only when the CPU is in RUN mode.

To configure this protocol, the following steps must be performed:

1. Select the Ethernet interface (NET1 ... NET6) where the IEC 60870-5-104 Client will be instantiated, and configure this interface. To do this, see the section Ethernet Interfaces Configuration.
2. Add an instance of the IEC 60870-5-104 Client protocol below the selected Ethernet interface. To perform this procedure, see the section Inserting a Protocol Instance.
3. Add and configure *Controlled Station* devices below the IEC 60870-5-104 Client instance, as shown in the figure 90. There is a maximum limit of 32 Controlled Stations under each instance of the IEC 60870-5-104 Client.

altus

4. Configure the link layer parameters of each Controlled Station device.
5. Add and configure sectors below the Controlled Station device, as shown in the figure 91. There is a maximum limit of 5 sectors for each device.
6. Perform the following settings for each added sector:

   ■ Add and configure IEC 60870-5-104 mapping of the sector;

   ■ Configure the link layer parameters of the sector;

   ■ Configure the application layer parameters of the sector.

The descriptions of each configuration are listed below in this section.
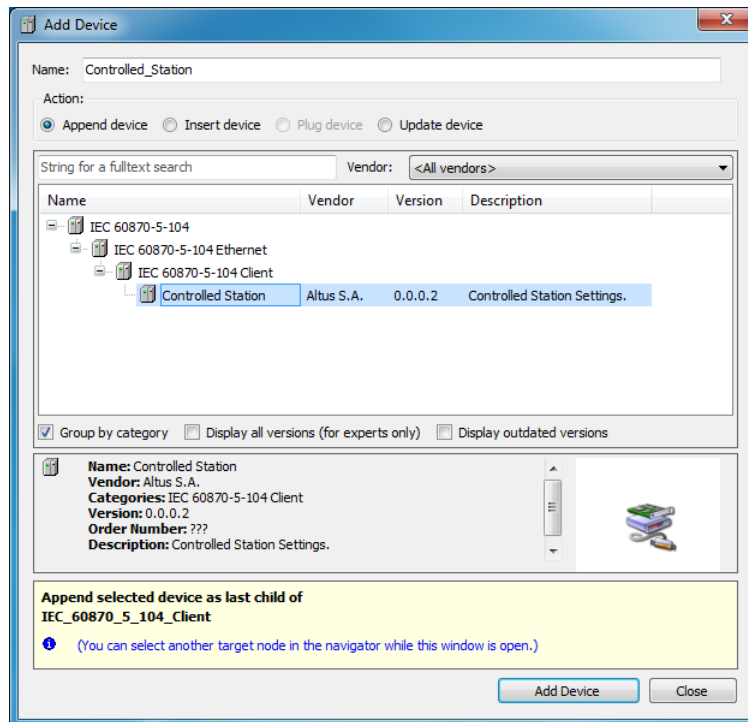
Figure 90: Insertion Screen of a Controlled Station Device below an IEC 60870-5-104 Client
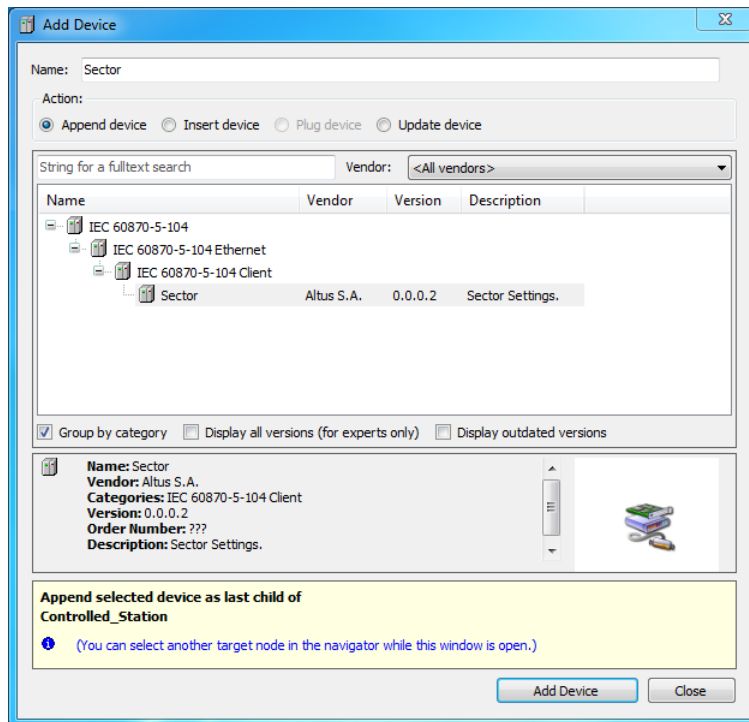
Figure 91: Insert screen of a Sector below a Controlled Station IEC 60870-5-104 Device

4.5.14.1. Data Link Layer configuration of a Controlled Station

The configuration of the link layer parameters of a Controlled Station, added under an IEC 60870-5-104 Client, is shown in the figure below. The following table describes the fields that must be filled in on this screen.

Figure 92: IEC 60870-5-104 Server Link Layer Configuration Screen

Configuration	Description	Default	Options
Port Number	Address of the TCP listen port for connection to this Controlled Station.	2404	0 to 65535
IP Address	IP address of this Controlled Station.	0.0.0.0	1.0.0.1 to 223.255.255.254
Time-out t1(s)	Period of time (in seconds) the device waits to receive a confirmation message after sending an APDU type I or U message (data) before closing the connection.	15	1 to 180
Time-out t2(s)	Period of time (in seconds) the device waits to send a supervisory message (S-Frame) confirming receipt of data frames.	10	1 to 180
Time-out t3(s)	Period of time (in seconds) that the device will send a link test message if there is no transmission on both sides.	20	1 to 180
Parameter k (APDUs)	Maximum number of data messages (I-Frame) transmitted and not confirmed.	12	1 to 12
Parameter w (APDUs)	Maximum number of data messages (I-Frame) received and not confirmed.	8	1 to 8

Table 141: Configuration of the Link Layer of a Controlled Station

4.5.14.2. Mappings Configuration of an IEC 60870-5-104 Sector

The configuration of the mappings of an IEC 60870-5-104 sector, added below a Controlled Station, is shown in the figure below. The following table describes the columns that must be filled in for each mapping.
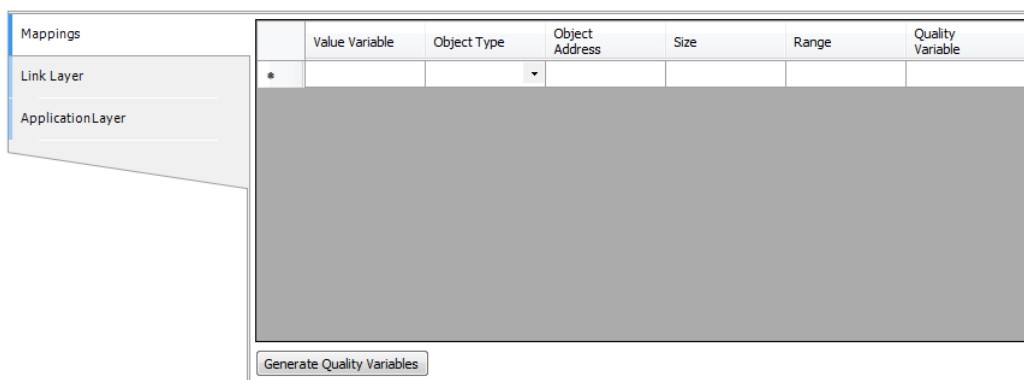
Figure 93: Data Mapping Screen of an IEC 60870-5-104 Sector

COnfiguration	Description	Default	Options
Value Variable	Name of the symbolic variable.	-	Name of a variable declared in a program or GVL.
Object Type	IEC 60870-5-104 object type configuration.	-	Single Point Information. Double Point Information. Step Position Information. Measured Value (Normalized). Measured Value (Scaled). Measured Value (Short Floating Point). Integrated Totals. Single Command. Double Command. Regulating Step Command. Setting Point Command (Normalized). Setting Point Command (Scaled). Setting Point Command (Short Floating Point).
Object Address	Index of the first point of the IEC 60870-5-104 mapping.	-	1 to 65535
Size	Specifies the maximum amount of data that an IEC 60870-5-104 mapping can access.	-	- (automatically calculated)
Range	Address range of the configured data.	-	- (automatically calculated)
Quality Variable	Name of the symbolic variable that will store the quality of the mapping.	-	Variable of type QUALITY
Generate Quality Variable	This button automatically generates in GVL Qualities variables of type QUALITY for each declared Value Variable. For more information please see the section Quality Conversions.	-	-

Table 142: Configuration of Data Mappings of an IEC 60870-5-104 Sector

Notes:

Value Variable: Name of the symbolic variable to be mapped. When a reading command is sent, the return sent in the response will be stored in this variable. When it is a writing command, the value written will be copied from this variable. The variable can be simple, array, array element, simple structure element, or structure array element. The symbolic variable can only be mapped in one sector and must not be mapped in any other client protocol.

Object Type: Selects the IEC 60870-5-104 object type linked to the mapping.

Object Address: The field is used to set the address of the first mapping point in the IEC 60870-5-104 protocol.

Size: This is a read-only field that informs how many points are being used by this mapping. The value of this field is calculated automatically according to the type of variable configured in the mapping.

Range: This is a read-only field that informs the address range that is being used by this mapping. The value of this field is automatically calculated by summing the "Object Address" and "Size" fields. There can be no range overlaps with other mappings.

Quality Variable: Name of the symbolic variable that will store the quality of the mapping, this is always of type QUALITY, available from the LibDataTypes library. The variable can be simple, array or array element and can be in structures. The symbolic variable can only be mapped in one sector and must not be mapped in any other client protocol. It is not allowed to map quality variables to IEC 60870-5-104 command types.

altus

> **ATTENTION**
>
> The IEC 60870-5-104 Client driver does not generate events based on the values of its mappings. Events must be generated by the respective IED. Therefore, if the IED does not support event generation, the IEC 60870-5-104 Client driver will not be able to generate them.

4.5.14.3. Data Link Layer configuration of an IEC 60870-5-104 Sector

The configuration of the link layer parameters of an IEC 60870-5-104 sector, added below a Controlled Station, is shown in the figure below. The following table describes the fields that must be filled in on this screen.
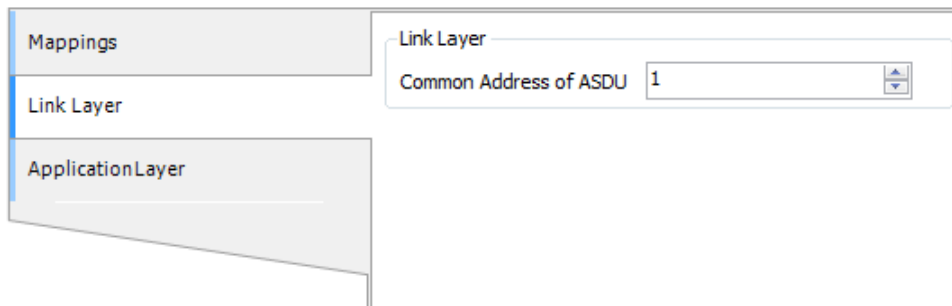
Figure 94: Data Link Layer Configuration Screen of an IEC 60870-5-104 Sector

Configuration	Description	Default	Options
Common Address of ADSU	IEC 60870-5-104 address of this sector.	1	1 to 65534

Table 143: Configuration of the Link Layer of an IEC 60870-5-104 Sector

4.5.14.4. Application Layer Configuration of an IEC 60870-5-104 Sector

The configuration of the application layer parameters of an IEC 60870-5-104 sector, added below a Controlled Station, is shown in the figure below. The following table describes the fields that must be filled in on this screen.
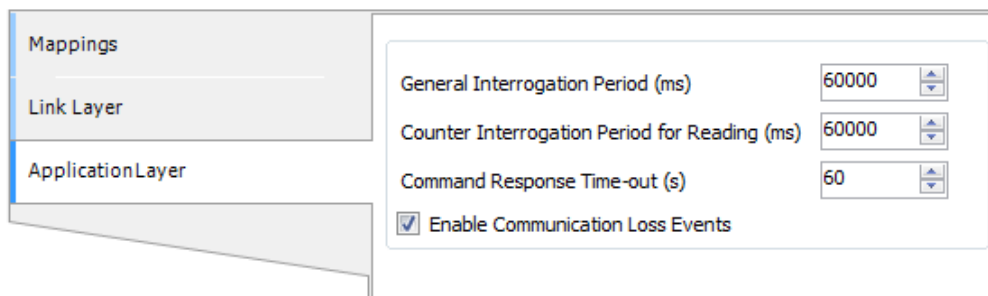
Figure 95: Application Layer Configuration of an IEC 60870-5-104 Sector

Configuration	Description	Default	Options
General Interrogation Period (ms)	Period for execution of the "General Interrogation" command that reads all mapped points (except counters). The special value 0 disables the execution of the command.	60000	0 disable. 1 to 86400000. [milliseconds]
Counter Interrogation Period for Reading (ms)	Period for execution of the "Counter Interrogation" command to read all mapped counters. The special value 0 disables the execution of the command.	60000	0 disable. 1 to 86400000. [milliseconds]
Command Response Time-out (s)	Maximum time for receiving the response of a command. If this time is exceeded, the command error diagnostic counter is incremented.	60	1 to 3600. [seconds]
Enable Communication Loss Events	Enables the generation of communication loss and reestablishment events with the Controlled Stations.	Enabled	Enabled. Disabled.

Table 144: IEC 60870-5-104 Client Application Layer Configuration

4.5.14.5. IEC 60870-5-104 Client Diagnostics

The diagnostics of the configured IEC 60870-5-104 Client protocol are divided into two structures, one for the Client and one for each Controlled Station declared below it. The diagnostics for the client are stored in variables of type T_DIAG_IEC104_CLIENT_1 which are described in table 145. The diagnostics for Controlled Station are stored in variables of type T_DIAG_IEC104_CONTROLLED_STATION_1, which are described in table 146.

Diagnostic Variable T_DIAG_IEC104_CLIENT_1.	Size	Description
Command bits, automatically reset:		
tCommand. bStop	BIT	Disable Driver.
tCommand. bStart	BIT	Enable Driver.
tCommand. bDiag_01_Reserved	BIT	Reserved.
tCommand. bDiag_02_Reserved	BIT	Reserved.
tCommand. bDiag_03_Reserved	BIT	Reserved.
tCommand. bDiag_04_Reserved	BIT	Reserved.
tCommand. bDiag_05_Reserved	BIT	Reserved.

altus

Diagnostic Variable T_DIAG_ IEC104_CLIENT_1.	Size	Description
tCommand. bDiag_06_Reserved	BIT	Reserved.
Diagnostics:		
bRunning	BOOL	IEC 60870-5-104 Client is running.

Table 145: IEC 60870-5-104 Client Diagnostics

Diagnostic Variable T_DIAG_IEC104_ CONTROLLED_STATION_1.	Size	Description
Connection Diagnostics:		
eConnectionStatus. CLOSED	BYTE	Closed communication with the IED.
eConnectionStatus. CONNECTED	BYTE	Communication active with the IED.
Communication Statistics:		
tStat. wRXFrames	WORD	Number of frames received.
tStat. wTXFrames	WORD	Number of frames sent.
tStat. wCommErrors	WORD	Counter of communication errors including Physical Layer, Link Layer and Transport Layer errors.
tStat. wGeneralInterrogationErrors	WORD	Counter of errors in the cyclic "General Interrogation" command.
tStat. wCounterInterrogationErrors	WORD	Counter of errors in cyclic "Counter Interrogation" command.
tStat. dwReserved_0	DWORD	Reserved.
tStat. dwReserved_1	DWORD	Reserved.

Table 146: Controlled Station IEC 60870-5-104 Diagnostics

4.5.14.6. IEC 60870-5-104 Commands for IEDs

In this section are listed the user commands available to be sent via IEC 60870-5-104 to IEDs. These commands are executed with the help of functional blocks, described below.

Due to the characteristics of the IEC 60870-5-104 Client protocol, it is not possible to send simultaneous user commands to the same Sector of an IED, even if they are addressed to different points in the Sector. Thus, when a command is sent to an IED's Sector, the next command to this same IED Sector will only be sent after the execution of the first command is finished.

Then, if a command is triggered by the user application while there is another command executing for the same IED Sector, the new command will wait for the first command to finish for a period of time defined in the functional block *udiCommandTimeOut*. And if the time for the end of execution of the first command is longer than *udiCommandTimeOut*, the new command will return an error message (IEC104_COMMAND_ERROR_STATUS_TIMEOUT). That is, a command starts counting timeout time from the moment it was triggered by the user application, not only when it was sent by the IEC 60870-5-104 Client.

4.5.14.6.1. *Commands for Digital Outputs*

The commands for the digital outputs are executed with the **IEC104_DigitalCommand** functional block. This functional block allows to execute commands on the following types of objects:

altus

- Single Command (C_SC_NA)
- Double Command (C_DC_NA)
- Regulating Step Command (C_RC_NA)

In the following tables, the input and output parameters of this functional block are described.

Parameter	Type	Description
bRequest	BOOL	When TRUE, executes the command.
dwVariableAddr	DWORD	Command variable address to send.
eCommand	ENUM (BYTE)	Command that will be sent: IEC104_COMMAND_TYPE_SELECT (0) IEC104_COMMAND_TYPE_EXECUTE (1) IEC104_COMMAND_TYPE_DEACTIVATE (2)
eQualifier	ENUM (BYTE)	IEC104_QUALIFIER_DEFAULT (0) IEC104_QUALIFIER_SHORT_PULSE (1) IEC104_QUALIFIER_LONG_PULSE (2) IEC104_QUALIFIER_PERSISTENT (3)
eValue	ENUM (BYTE)	IEC104_VALUE_OFF (0) IEC104_VALUE_ON (1) IEC104_VALUE_LOWER (2) IEC104_VALUE_HIGHER (3)
udiCommandTimeOut	UDINT	Time for the Time Out of this command (milliseconds).

Table 147: IEC104_DigitalCommand Functional Block Output Parameters

Parameter	Type	Description
eExecStatus	ENUM (BYTE)	Execution status, indicates if a new command can be sent, the values are as follows: IEC104_COMMAND_EXEC_STATUS_DONE (0) IEC104_COMMAND_EXEC_STATUS_RUNNING (1)
eErrorStatus	ENUM (BYTE)	Indicates whether the last command was successful. IEC104_COMMAND_ERROR_STATUS_NO_ERROR (0) IEC104_COMMAND_ERROR_STATUS_TIMEOUT (1) IEC104_COMMAND_ERROR_STATUS_FORMAT_ERROR (2) IEC104_COMMAND_ERROR_STATUS_NOT_SUPPORTED (3) IEC104_COMMAND_ERROR_STATUS_INTERNAL_ERROR (4) IEC104_COMMAND_ERROR_STATUS_STATION_ERROR (5) IEC104_COMMAND_ERROR_STATUS_INVALID_ADDRESS (6) IEC104_COMMAND_ERROR_STATUS_WRONG_TIMEOUT (7) IEC104_COMMAND_ERROR_STATUS_UNINITIALIZED (8)

Table 148: IEC104_DigitalCommand Functional Block Input Parameters

Notes:

- **eValue:**
  - The values IEC104_VALUE_ON and IEC104_VALUE_OFF shall be used exclusively for objects of the types Single Command (C_SC_NA) and Double Command (C_DC_NA).
  - The values IEC104_VALUE_LOWER and IEC104_VALUE_HIGHER shall be used exclusively for objects of type Regulating Step Command (C_RC_NA).

altus

The commands for the analog outputs are executed with the **IEC104_AnalogCommandInt** and **IEC104_ AnalogCommandReal** function blocks.

The **IEC104_AnalogCommandInt** functional block allows to execute commands on the following types of objects:

- Setting Point Command, normalized Value (C_SE_NA)
- Setting Point Command, scaled Value (C_SE_NB)

In the following tables the input and output parameters of the **IEC104_AnalogCommandInt** functional block are described.

Parameter	Type	Description
bRequest	BOOL	When TRUE, executes the command.
dwVariableAddr	DWORD	Command variable address to send.
eCommand	ENUM (BYTE)	Command that will be sent: IEC104_COMMAND_TYPE_SELECT (0) IEC104_COMMAND_TYPE_EXECUTE (1) IEC104_COMMAND_TYPE_DEACTIVATE (2)
iValue	INT	Analog value to be written: INT.
udiCommandTimeOut	UDINT	Time for the Time Out of this command (milliseconds).

Table 149: IEC104_AnalogCommandInt Block Input Parameters

Parameter	Type	Description
eExecStatus	ENUM (BYTE)	Execution status, indicates if a new command can be sent, the values are as follows: IEC104_COMMAND_EXEC_STATUS_DONE (0) IEC104_COMMAND_EXEC_STATUS_RUNNING (1)
eErrorStatus	ENUM (BYTE)	Indicates whether the last command was successful. IEC104_COMMAND_ERROR_STATUS_NO_ERROR (0) IEC104_COMMAND_ERROR_STATUS_TIMEOUT (1) IEC104_COMMAND_ERROR_STATUS_FORMAT_ERROR (2) IEC104_COMMAND_ERROR_STATUS_NOT_SUPPORTED (3) IEC104_COMMAND_ERROR_STATUS_INTERNAL_ERROR (4) IEC104_COMMAND_ERROR_STATUS_STATION_ERROR (5) IEC104_COMMAND_ERROR_STATUS_INVALID_ADDRESS (6) IEC104_COMMAND_ERROR_STATUS_WRONG_TIMEOUT (7) IEC104_COMMAND_ERROR_STATUS_UNINITIALIZED (8)

Table 150: IEC104_AnalogCommandInt Block Output Parameters

The **IEC104_AnalogCommandReal** functional block allows to execute commands on the following types of objects:

- Setting Point Command, short floating point Value (C_SE_NC).

The following tables describe the input and output parameters of the **IEC104_AnalogCommandReal** functional block.

altus

Parameter	Type	Description
bRequest	BOOL	When TRUE, executes the command.
dwVariableAddr	DWORD	Address of the command variable to send.
eCommand	ENUM (BYTE)	Command that will be sent: IEC104_COMMAND_TYPE_SELECT (0) IEC104_COMMAND_TYPE_EXECUTE (1) IEC104_COMMAND_TYPE_DEACTIVATE (2)
rValue	REAL	Analog value that will be written: REAL.
udiCommandTimeOut	UDINT	Time for the Time Out of this command (milliseconds).

Table 151: IEC104_AnalogCommandReal Block Input Parameters

Parameter	Type	Description
eExecStatus	ENUM (BYTE)	Execution status, indicates if a new command can be sent, values are as follows: IEC104_COMMAND_EXEC_STATUS_DONE (0) IEC104_COMMAND_EXEC_STATUS_RUNNING (1)
eErrorStatus	ENUM (BYTE)	Indicates whether the last command was successful, the values are as follows: IEC104_COMMAND_ERROR_STATUS_NO_ERROR (0) IEC104_COMMAND_ERROR_STATUS_TIMEOUT (1) IEC104_COMMAND_ERROR_STATUS_FORMAT_ERROR (2) IEC104_COMMAND_ERROR_STATUS_NOT_SUPPORTED (3) IEC104_COMMAND_ERROR_STATUS_INTERNAL_ERROR (4) IEC104_COMMAND_ERROR_STATUS_STATION_ERROR (5) IEC104_COMMAND_ERROR_STATUS_INVALID_ADDRESS (6) IEC104_COMMAND_ERROR_STATUS_WRONG_TIMEOUT (7) IEC104_COMMAND_ERROR_STATUS_UNINITIALIZED (8)

Table 152: IEC104_AnalogCommandReal Block Output Parameters

4.5.14.6.3. *Reset Process Command*

The **IEC104_ResetProcess** functional block allows to reset an IEC 60870-5-104 Server device. In the following tables the input and output parameters of the functional block can be seen.

Parameter	Type	Description
bRequest	BOOL	When TRUE, executes the command.
dwVariableAddr	DWORD	Address of any variable mapped on the server, used only to identify the server.
udiCommandTimeOut	UDINT	Time for the Time Out of this command (milliseconds).

Table 153: IEC104_ResetProcess Block Input Parameters

Parameter	Type	Description
eExecStatus	ENUM (BYTE)	Execution status, indicates if a new command can be sent, values are as follows: IEC104_COMMAND_EXEC_STATUS_DONE (0) IEC104_COMMAND_EXEC_STATUS_RUNNING (1)
eErrorStatus	ENUM (BYTE)	Indicates whether the last command was successful, the values are as follows: IEC104_COMMAND_ERROR_STATUS_NO_ERROR (0) IEC104_COMMAND_ERROR_STATUS_TIMEOUT (1) IEC104_COMMAND_ERROR_STATUS_FORMAT_ERROR (2) IEC104_COMMAND_ERROR_STATUS_NOT_SUPPORTED (3) IEC104_COMMAND_ERROR_STATUS_INTERNAL_ERROR (4) IEC104_COMMAND_ERROR_STATUS_STATION_ERROR (5) IEC104_COMMAND_ERROR_STATUS_INVALID_ADDRESS (6) IEC104_COMMAND_ERROR_STATUS_WRONG_TIMEOUT (7) IEC104_COMMAND_ERROR_STATUS_UNINITIALIZED (8)

Table 154: IEC104_ResetProcess Block Output Parameters

4.5.14.6.4. *Synchronize IED Command*

Using the **IEC104_SynchronizeIED** functional block it is possible to command a clock synchronization of an IED (IEC 60870-5-104 Server).

In the following tables the input and output parameters of the functional block can be seen.

Parameter	Type	Description
bRequest	BOOL	When TRUE, executes the command.
dwVariableAddr	DWORD	Address of any variable mapped on the server, used only to identify the server.
udiCommandTimeOut	UDINT	Time for the Time Out of this command (milliseconds).

Table 155: IEC104_SynchronizeIED Block Input Parameters

Parameter	Type	Description
eExecStatus	ENUM (BYTE)	Execution status, indicates if a new command can be sent, values are as follows: IEC104_COMMAND_EXEC_STATUS_DONE (0) IEC104_COMMAND_EXEC_STATUS_RUNNING (1)
eErrorStatus	ENUM (BYTE)	Indicates whether the last command was successful, the values are as follows: IEC104_COMMAND_ERROR_STATUS_NO_ERROR (0) IEC104_COMMAND_ERROR_STATUS_TIMEOUT (1) IEC104_COMMAND_ERROR_STATUS_FORMAT_ERROR (2) IEC104_COMMAND_ERROR_STATUS_NOT_SUPPORTED (3) IEC104_COMMAND_ERROR_STATUS_INTERNAL_ERROR (4) IEC104_COMMAND_ERROR_STATUS_STATION_ERROR (5) IEC104_COMMAND_ERROR_STATUS_INVALID_ADDRESS (6) IEC104_COMMAND_ERROR_STATUS_WRONG_TIMEOUT (7) IEC104_COMMAND_ERROR_STATUS_UNINITIALIZED (8)

Table 156: IEC104_SynchronizeIED Block Output Parameters

altus

4.5.14.6.5.   General Interrogation Command

In the application layer configuration of the IEC 60870-5-104 Client it is possible to configure the periodic execution of General Interrogation commands for reading only all mapped points (except counters).

This command can also be executed through the functional block **IEC104_GeneralInterrogation**.

In the following tables the input and output parameters of the functional block are described.

Parameter	Type	Description
bRequest	BOOL	When TRUE, executes the command.
dwVariableAddr	DWORD	Address of any variable mapped on the server, used only to identify the server.
udiCommandTimeOut	UNSIGNED DOUBLE INT	Time for the Time Out of this command (milliseconds).

Table 157: IEC104_GeneralInterrogation Block Input Parameters

Parameter	Type	Description
eExecStatus	ENUM (BYTE)	Execution status, indicates if a new command can be sent, values are as follows: IEC104_COMMAND_EXEC_STATUS_DONE (0) IEC104_COMMAND_EXEC_STATUS_RUNNING (1)
eErrorStatus	ENUM (BYTE)	Indicates whether the last command was successful, the values are as follows: IEC104_COMMAND_ERROR_STATUS_NO_ERROR (0) IEC104_COMMAND_ERROR_STATUS_TIMEOUT (1) IEC104_COMMAND_ERROR_STATUS_FORMAT_ERROR (2) IEC104_COMMAND_ERROR_STATUS_NOT_SUPPORTED (3) IEC104_COMMAND_ERROR_STATUS_INTERNAL_ERROR (4) IEC104_COMMAND_ERROR_STATUS_STATION_ERROR (5) IEC104_COMMAND_ERROR_STATUS_INVALID_ADDRESS (6) IEC104_COMMAND_ERROR_STATUS_WRONG_TIMEOUT (7) IEC104_COMMAND_ERROR_STATUS_UNINITIALIZED (8)

Table 158: IEC104_GeneralInterrogation Block Output Parameters

4.5.14.6.6.   Counter Interrogation Command

In the application layer configuration of the IEC 60870-5-104 Client it is possible to configure the periodic execution of Counter Interrogation commands only for counter reads.

Through the **IEC104_CounterInterrogation** functional block it is possible to execute other options of the *Counter Interrogation* command:

- Ler
- Freeze
- Reset
- Reset e Freeze

In the following tables the input and output parameters of the functional block are described.

altus

Parameter	Type	Description
bRequest	BOOL	When TRUE, executes the command.
dwVariableAddr	DWORD	Address of any variable mapped on the server, used only to identify the server.
eCommand	ENUM (BYTE)	Command that will be sent: IEC104_COMMAND_READ (0) IEC104_COMMAND_FREEZE (1) IEC104_COMMAND_RESET (2) IEC104_COMMAND_FREEZE_RESET (3)
udiCommandTimeOut	UDINT	Time for the Time Out of this command (milliseconds).

Table 159: IEC104_CounterInterrogation Block Input Parameters

Parameter	Type	Description
eExecStatus	ENUM (BYTE)	Execution status, indicates if a new command can be sent, values are as follows: IEC104_COMMAND_EXEC_STATUS_DONE (0) IEC104_COMMAND_EXEC_STATUS_RUNNING (1)
eErrorStatus	ENUM (BYTE)	Indicates whether the last command was successful, the values are as follows: IEC104_COMMAND_ERROR_STATUS_NO_ERROR (0) IEC104_COMMAND_ERROR_STATUS_TIMEOUT (1) IEC104_COMMAND_ERROR_STATUS_FORMAT_ERROR (2) IEC104_COMMAND_ERROR_STATUS_NOT_SUPPORTED (3) IEC104_COMMAND_ERROR_STATUS_INTERNAL_ERROR (4) IEC104_COMMAND_ERROR_STATUS_STATION_ERROR (5) IEC104_COMMAND_ERROR_STATUS_INVALID_ADDRESS (6) IEC104_COMMAND_ERROR_STATUS_WRONG_TIMEOUT (7) IEC104_COMMAND_ERROR_STATUS_UNINITIALIZED (8)

Table 160: IEC104_CounterInterrogation Block Output Parameters

4.5.14.6.7. Utilization Example of a Functional Block for IEC 60870-5-104 Command

Below is an example of an application using the ST language, in which the user sends the command *Select Before Execute* to an IED through the IEC 60870-5-104 protocol.

```
PROGRAM UserPrg
VAR
    iIec104ClientAO:        INT;
    iIec104ClientAOCmd:     LibIEC104.IEC104_AnalogCommandInt;
END_VAR

iIec104ClientAOCmd.dwVariableAddr:= ADR(iIec104ClientAO);
iIec104ClientAOCmd();
IF  iIec104ClientAOCmd.eExecStatus = LibIEC104.IEC104_COMMAND_EXEC_STATUS_DONE
    AND
  iIec104ClientAOCmd.bRequest = TRUE THEN
    IF iIec104ClientAOCmd.eCommand = LibIEC104.IEC104_COMMAND_TYPE_SELECT AND
       iIec104ClientAOCmd.eErrorStatus = LibIEC104.
    IEC104_COMMAND_ERROR_STATUS_NO_ERROR THEN
```

```
            iIec104ClientAOCmd.eCommand:= LibIEC104.IEC104_COMMAND_TYPE_EXECUTE;
    ELSE
            iIec104ClientAOCmd.bRequest:= FALSE;
    END_IF
  END_IF
```

4.5.14.6.8. Error codes for IEC 60870-5-104 Commands

IEC 60870-5-104 commands return error status (*eErrorStatus*) when executed. The table below shows the list and description of possible values for the error status.

Code	Status	Description
0	IEC104_COMMAND_ERROR_STATUS_NO_ERROR	Command accepted, started or queued.
1	IEC104_COMMAND_ERROR_STATUS_TIMEOUT	Command not accepted because the message was received after the command time out.
2	IEC104_COMMAND_ERROR_STATUS_FORMAT_ERROR	Command not accepted because there are formatting errors in the request.
3	IEC104_COMMAND_ERROR_NOT_SUPPORTED	Command not accepted because the operation is not supported by this point.
4	IEC104_COMMAND_ERROR_INTERNAL_ERROR	Command not accepted because of an internal failure.
5	IEC104_COMMAND_ERROR_STATION_ERROR	Command not accepted because of an unknown Controlled Station fault.
6	IEC104_COMMAND_ERROR_INVALID_ADDRESS	Command not accepted because the variable address is invalid.
7	IEC104_COMMAND_ERROR_WRONG_TIMEOUT	Command not accepted because the time out value is not valid.
8	IEC104_COMMAND_ERROR_UNINITIALIZED	Command not accepted because the functional block was not started correctly.

Table 161: Error Status for IEC 60870-5-104 Commands

4.5.15. IEC 60870-5-104 Servidor

This protocol is available for the Hadron Xtorm Series CPU on its Ethernet channels. By selecting this option in MasterTool Xtorm, the CPU becomes a server for IEC 60870-5-104 communication, allowing connection with up to five IEC 60870-5-104 client devices. For each client, the driver has a unique event queue with the following characteristics:

- Size: 4500 events.
- Overflow policy: keep the most recent one.

To configure this protocol, the following steps must be performed:

1. Add the IEC 60870-5-104 Server protocol instance to one of the available Ethernet channels (NET 1 .. NET 6). To perform this procedure, see section Inserting a Protocol Instance.
2. Configure the Ethernet interface. To do this, see the section Ethernet Interfaces Configuration.

3. Configure the general parameters of the IEC 60870-5-104 Server protocol, with Port or IP connection mode, and the TCP port number when the selected connection mode is IP mode.
4. Add and configure devices by setting the appropriate parameters.
5. Add and configure IEC 60870-5-104 mappings by specifying the variable name, object type, object address, size, range, deadband, and deadband type.
6. Configure the parameters of the link layer, specifying addressing, communication time-outs, and communication parameters.
7. Configure the application layer parameters, synchronism configuration, commands, as well as the transmission mode of Integrated Totals objects.

The descriptions of each configuration are listed below in this section.

4.5.15.1. General Parameters

The General Parameter settings of the IEC 60870-5-104 Server as shown in the figure below, follow the parameters described in the following table:
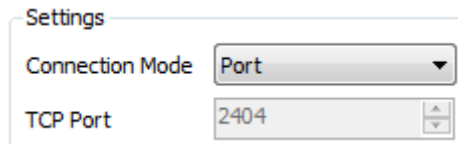
Figure 96: IEC 60870-5-104 Server General Parameters Screen

Configuration	Description	Default	Options
Connection Mode	Configures the connection mode with the Connected Client modules.	Port	Port IP
TCP Port	Defines which RTU TCP port number will be used for communication with the Connected Client modules, if "IP" is selected in the "Connection Mode" field.	2404	1 to 65535

Table 162: IEC 60870-5-104 Server General Parameter Configuration

4.5.15.2. Data Mapping

For the configuration of the IEC 60870-5-104 Server data relations, visualized in the figure below, follow the parameters described in the following table.

Figure 97: Data Mapping Screen of IEC 60870-5-104 Server

Configuration	Description	Default	Options
Value Variable	Name of the symbolic variable.	-	Name of a variable declared in a program or GVL.
Object Type	IEC 60870-5-104 object type configuration.	-	Single Point Information Double Point Information Step Position Information Measured Value (Normalized) Measured Value (Scaled) Measured Value (Short Floating Point) Integrated Totals Bitstring Information (32 Bits) Single Command Double Command Regulating Step Command Setting Point Command (Normalized) Setting Point Command (Scaled) Setting Point Command (Short Floating Point) Bitstring Command (32 Bits)
Object Address	Index of the first point of the IEC 60870-5-104 mapping.	-	1 to 65535
Size	Specifies the maximum amount of data that an IEC 60870-5-104 mapping can access.	-	- (automatically calculated)
Range	Address range of the configured data.	-	- (automatically calculated)
Counter Variable	Name of the symbolic variable that will store the counter variable data.	-	Name of a variable declared in a program, GVL, or counter module.
Dead Band Variable	Name of the symbolic variable that will store the deadband data.	-	Name of a variable declared in a program or GVL.
Dead Band Type	Defines the type of Deadband to be used in the mapping.	Disabled	Absolute Disabled Integrated
Select Required	Defines whether or not a previous select is required to execute a command.	False	True False
Short Pulse	Defines the short pulse time of an IEC 60870-5-104 digital command, in milliseconds.	1000	1 to 86400000
Long Pulse	Defines the long pulse time of an IEC 60870-5-104 digital command, in milliseconds.	2000	1 to 86400000

Table 163: Configuration of the IEC 60870-5-104 Server Mappings

Notes:

Value Variable: Name of the symbolic variable to be mapped. When a reading command is sent, the return sent in the response will be stored in this variable. When it is a writing command, the value written will be copied from this variable. The variable can be simple, array or array element, and can be in structures.

Counter Variable: This field only applies when mapping objects of type Integrated Totals, this being the counter variable, to be manipulated in the process. It must have the same type and size as the variable mapped in the Value Variable column, whose value will be read by the client or reported to the client in case of events.

> **ATTENTION**
>
> When the Counter Variable has a quality variable associated with it, for this quality to be transferred to the frozen variable in the *freeze* command, a quality variable must be associated with the frozen variable. This can be done through the *Internal Points* tab.

Dead Band Variable: This field only applies to the mapping of analog input variables (Measured Value objects). It must have the same type and size as the variable mapped in the Value Variable column. New values of the deadband variable will be considered only when the analog input variable changes its value.

Counter Variable: The available Deadband configuration types are as follows:

Function Type	Configuratios	Description
Dead Band Type	Disabled	In this option, any value variation in a group's point, however small, generates event to this point.
	Absolute	In this option, if the value variation module of a group's point is greater than the value set by the variable in the field "Dead Band", an event is generated for this point.
	Integrated	In this option, if the magnitude of the value variation integral in a group's point is greater than the value set by the variable in the field "Dead Band", an event is generated for this point. The integration interval is 1 second.

Table 164: Dead Band Types in IEC 60870-5-104 Server Parameters

Note:

Short Pulse and Long Pulse: When defining the time length of the short and long pulses the limits supported by the device that will handle the command must be taken into account. For example, if the destination is an output card, you should check in the module's CE which are the minimum and maximum times, as well as the resolution, for the execution of pulsed commands.

> **ATTENTION**
>
> The dead band set to variables mapped in IEC 60870-5-104 Server is not applied when these points are derived from an IED (through the DNP3 Client driver, IEC 61850 GOOSE Subscriber, etc...), because in these case the event detection occurs in the IED itself.

4.5.15.3. Link Layer

For the configuration of the IEC 60870-5-104 Server's link layer parameters, seen in the figure below, the parameters described in the following table are used.
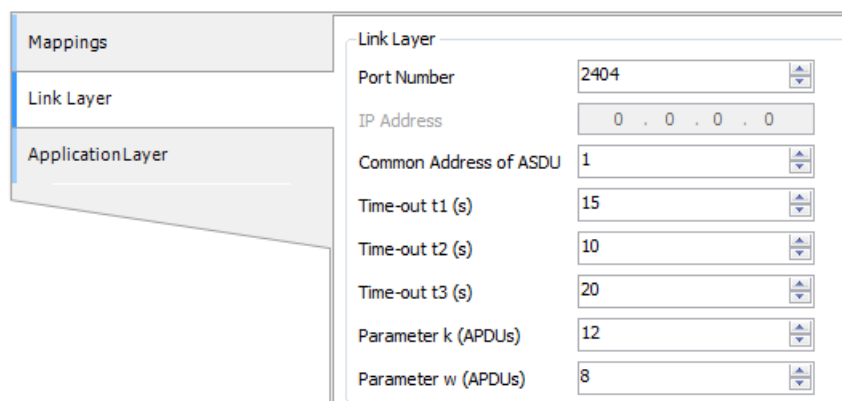
Figure 98: IEC 60870-5-104 Server Data Link Layer Configuration Screen

Configuration	Description	Default	Options
Port Number	Listen port address for client connection. Used when the client connection is not IP.	2404	1 to 65535
IP Address (Master)	IP of the connected client. Used when the client connection is not IP.	0.0.0.0	1.0.0.1 to 223.255.255.254
Common Address of ASDU	IEC 60870-5-104 address of the server.	1	1 to 65535
Time-out t1(s)	Period of time (in seconds) that the device waits to the receipt of a confirmation message after sending a APDU message of the type I or U (data), before closing the connection.	15	1 to 180
Time-out t2(s)	Period of time (in seconds) that the device waits to send a supervisory message (SFrame) confirming receipt of data frames.	10	1 to 180
Time-out t3(s)	Period of time (in seconds) that will be sent a message to test if there is no link transmission by both sides.	20	1 to 180
Parameter k (APDUs)	Maximum number of data messages (I-Frame) transmitted and not confirmed.	12	1 to 12
Parameter w (APDUs)	Maximum number of data messages (I-Frame) received and not confirmed.	8	1 to 8

Table 165: IEC 60870-5-104 Server Link Layer Configuration

Note:

The fields **"Time-out t1 (s)"**, **"Time-out t2 (s)"** and **"Time-out t3 (s)"** are dependent on each other and must be set in such a way that **"Time-out t1 (s)"** is greater than **"Time-out t2 (s)"** and **"Time-out t3 (s)"** is greater than **"Time-out t1 (s)"**. If any of these rules are not respected, error messages will be displayed when compiling the project.

4.5.15.4. Application Layer

For the configuration of the IEC 60870-5-104 Server application layer parameters, seen in the figure below, the parameters described in the following table are used.
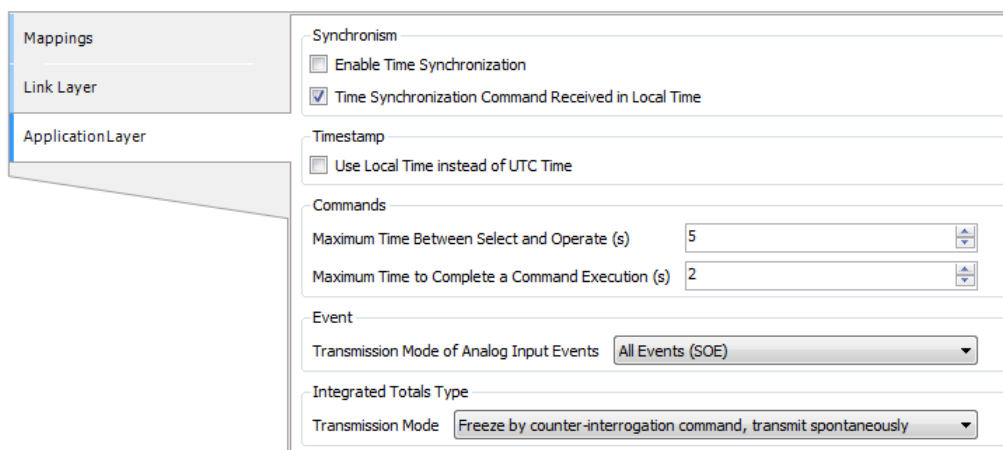
Figure 99: IEC 60870-5-104 Server Application Layer Configuration Screen

Configuration	Description	Default	Options
Enable Time Synchronization	Option to Enable/Disable time synchronization request.	Disabled	Enabled. Disabled.
Time Synchronization Command Received in Local Time	Option to Enable/Disable local time synchronization command handling.	Enabled	Enabled. Disabled.
Use Local Time instead of UTC Time	Option to Enable/Disable local time stamping for events.	Disabled	Enabled. Disabled.
Maximum Time Between Select and Operate (s)	Time period in which the selection command will remain active (starts counting from the confirmation of receipt of the selection command) waiting for the Operate command.	5	1 to 180
Maximum Time to Complete a Command Execution (s)	Maximum time to handle a command when transferred to a Client communication driver or even an output card, before it is answered failure by time-out.	2	1 to 180
Transmission Mode of Analog Inputs Events	Transmission mode of the analog input events.	All Events (SOE)	All Events (SOE). Most Recent Events.
Transmission Mode	Transmission mode of the counters' data.	Freeze by counter-interrogation command, transmit spontaneously.	Freeze by counter-interrogation command, transmit spontaneously. Freeze and transmit by counter-interrogation command.

Table 166: IEC 60870-5-104 Server Application Layer Configuration

Notes:

Enable Time Synchronization: When enabled, it allows the IEC 60870-5-104 Server to adjust the CPU clock when a synchronization command is received.

Time Synchronization Command Received in Local Time: When enabled, the IEC 60870-5-104 Server sets the CPU clock treating the time received in the synchronization command as local time. Otherwise this time is considered UTC.

Use Local Time Instead of UTC Time: When enabled, the time stamp of the events generated by the IEC 60870-5-104 Server will be sent according to the local time of the CPU.

> **ATTENTION**
>
> When the time synchronization option is checked on more than one server ("Enable Time Synchronization" parameter) the times received on the different servers will be overwritten in the system clock in a short space of time, which can cause undesirable behavior due to time delays in message propagation and system load.

Maximum Time to Complete a Command Execution: This parameter does not apply when the command is intercepted by the command intercept function block (CommandReceiver). In this case the maximum time to process the command is set by a parameter of the function block.

Transmission Mode of Analog Input Events: The available transmission mode of analog input Events are as follows:

Function Type	Configurations	Description
Transmission Mode of Analog Inputs Events	All Events(SOE)	All analog events generated are sent.
	Most Recent Events	Sent only the latest in each analog point.

Table 167: Transmission Modes of Analog Input Events in IEC 60870-5-104 Server

Function Type	Configurations	Description
Transmission Mode	Freeze by counter-interrogation command, transmit spontaneously.	It is equivalent to Mode D of counters acquisition (integrated totals) defined by IEC 60870-5-104. In this mode, counters' interrogation commands (freeze) from the control stations will freeze the counters. If the frozen values have changed, they will be reported through events
	Freeze and transmit by counter-interrogation command.	It is equivalent to Mode C of counters acquisition (integrated totals) defined by IEC 60870-5-104. In this mode, counters' interrogation commands (freeze) from the control stations will freeze the counters. Subsequentials counters interrogation commands (read) are sent through control station to gather frozen values.

Table 168: Transmission Modes of IEC 60870-5-104 Server Frozen Counters

> **ATTENTION**
>
> The IEC60870-5-104 standard, section "Transmission control using Start/Stop", proposes the use of STARTDT and STOPDT commands to control the data traffic between cliente and server, using simple and multiples connections. Even that Xtorm supports this commands, their utilization is not recommended to control data transmission, mainly with redundant CPUs, because such commands are not synchronized betweeen both redundant CPUs. Instead of using multiple connections between client and Xtorm server, it is suggested the utilization of NIC Teaming resource to supply redundant Ethernet channels (physically) and preserve the CPU resourses (control centers by CPU).

4.5.15.5. Server Diagnostics

The diagnostics and commands of the IEC 60870-5-104 configured server protocol are stored in variables of type *T_DIAG_IEC104_SE* which are described in the table below.

altus

Diagnostic Variable T_DIAG_ IEC104_SERVER_1.	Size	Description
Command bits, automatically reset:		
tCommand. bStop	BIT	Disable Driver.
tCommand. bStart	BIT	Enable Driver.
tCommand. bDiag_01_Reserved	BIT	Reserved.
tCommand. bDiag_02_Reserved	BIT	Reserved.
tCommand. bDiag_03_Reserved	BIT	Reserved.
tCommand. bDiag_04_Reserved	BIT	Reserved.
tCommand. bDiag_05_Reserved	BIT	Reserved.
tCommand. bDiag_06_Reserved	BIT	Reserved.
Diagnostics:		
tClient_X. bRunning	BOOL	IEC 60870-5-104 Server is running.
tClient_X. eConnectionStatus. CLOSED	ENUM (BYTE)	Communication channel closed. Server will not accept connection request. ENUM value (0).
tClient_X. eConnectionStatus. CONNECTED	ENUM (BYTE)	Server is listening on the configured port, and there are no connected clients. ENUM value (1).
tClient_X. eConnectionStatus. CONNECTED	ENUM (BYTE)	Client is connected. ENUM value (2).
tClient_X. tQueueDiags. bOverflow	BOOL	The client's queue is overflowing.
tClient_X. tQueueDiags. wSize	WORD	Configured queue size.
tClient_X. tQueueDiags. wUsage	WORD	Number of events in the queue.
tClient_X. tQueueDiags. dwReserved_0	DWORD	Reserved.
tClient_X. tQueueDiags. dwReserved_1	DWORD	Reserved.
tClient_X. tStats. wRXFrames	WORD	Number of frames received.

altus

Diagnostic Variable T_DIAG_ IEC104_SERVER_1.	Size	Description
tClient_X. tStats. wTXFrames	WORD	Number of frames sent.
tClient_X. tStats. wCommErrors	WORD	Communication error counter including Physical Layer, Link Layer and Transport Layer errors.
tClient_X. tStats. dwReserved_0	DWORD	Reserved.
tClient_X. tStats. dwReserved_1	DWORD	Reserved.

Table 169: IEC 60870-5-104 Server Diagnostics

4.5.15.6. Association of the objects with the DNP3 protocol

The table shows allowed associations between the IEC 60870-5-104 and DNP3 protocols.

IEC 60870-5-104 Object Type	DNP3 Group
Single Point Information (M_SP_NA)	g01 - Digital input
Double Point Information (M_DP_NA)	g03 - Double digital input
Step Position Information (M_ST_NA)	-
Measured Value, normalized value (M_ME_NA)	g30 - Analog input
Measured Value, scaled value (M_ME_NB)	g30 - Analog input
Measured Value, short floating point value (M_ME_NC)	g30 - Analog input
Single Command (C_SC_NA)	g10 - Digital output
Double Command (C_DC_NA)	g10 - Digital output
Regulating Step Command (C_RC_NA)	g10 - Digital output
Setting Point Command, normalized Value (C_SE_NA)	g40 - Analog output
Setting Point Command, scaled Value (C_SE_NB)	g40 - Analog output
Setting Point Command, short floating point Value (C_SE_NC)	g40 - Analog output

Table 170: Association of IEC 60870-5-104 Object Types with Their Correspondents in the DNP3

> **ATTENTION**
>
> Integrated Totals (M_IT_NA): It is not allowed to associate a variable used as a frozen counter simultaneously in the IEC 60870-5-104 and DNP3 protocols.
> Step Position Information (M_ST_NA): There is no equivalent group for the DNP3 protocol.

4.5.15.7. Commands for Output Points and Counters

The commands received by the IEC 60870-5-104 Server driver can have the following destinations:

- Internal Points
- Output Module (e.g. HX2320)
- Driver DNP3 Client
- Driver IEC 60870-5-104 Client
- Interception (CommandReceiver function block)

4.5.15.7.1.   Internal Points

Internal points are those ones represented by variables which are not associated to any of the others destinations described above. For this type of point, are not supported pulsed commands. If the IEC 60870-5-104 server receives a pulsed command (single or double) to an internal point, it return a failure code (negative confirmation), unless it is used the command intercepter.

4.5.15.7.2.   Output Module

If the variable mapped to the IEC 60870-5-104 output point is associated to an output card (e.g. HX2320) of the Xtorm Series, the command will be redirected and executed by the card itself. In the case of pulsed commands, this redirection will only work if the variable is of type DBP. If a variable of type BOOL is used, the server will return a negative confirmation (failure) message for pulsed commands, just as when a persistent command is directed to a mapped DBP variable on the output card. By intercepting and handling these commands the application can avoid returning negative (failure) messages.

> **ATTENTION**
>
> Commands intended for variables mapped onto a Nexto Series output card (for example an NX2020 module) are treated as commands for internal points.

4.5.15.7.3.   DNP3 Client Driver

If the variable is associated with an IED point in a DNP3 Client driver, the command will be redirected and sent to be executed by the IED itself and the variable value stored in the memory of the CPU is not updated. It is important to note that the DNP3 Client does not support persistent commands for double points, in this case a negative confirmation message is returned.

It should be considered that, due to the characteristics of the protocol, the DNP3 Client driver is able to execute only one command at a time. That is, as long as a command for a determined point is not finished (response or time-out), it is not possible to trigger a new command, even if it is for a different point. Thus, if a command is redirected from the IEC 60870-5-104 Server or even from the DNP3 Server to an IED associated with a DNP3 Client driver and the latter is busy executing a command, a negative confirmation message will be returned.

4.5.15.7.4.   IEC 60870-5-104 Client Driver

If the variable is associated with an IED point in an IEC 60870-5-104 Client driver, the command will be redirected and sent to be executed by the IED itself. In the case of trip/close commands, typically the variable should be of type DBP, however redirection will work the same way even if a variable of type BOOL is used.

4.5.15.7.5.   Commands Interception

The interception of commands received by the IEC 60870-5-104 Server driver is accomplished through the *CommandReceiver* function block as described in the Interception of Commands from Control Center section. This feature allows the interception of selection and execution commands.

4.5.15.7.6.   Commands Qualifier

The IEC 60870-5-104 standard provides four different command qualifiers for the Single Command, Double Command, and Regulating Step objects, all of which are supported by the Xtorm Server.

Each object type has a specific behavior for each command qualifier, as can be seen in the table below.

altus

	IEC 60870-5-104 protocol object type		
Qualifier	**Single Command**	**Double Command**	**Regulating Step Command**
No additional definition (standard)	Same behavior of persistent qualifier.	Same behavior of short pulse qualifier.	Same behavior of short pulse qualifier.
Short pulse duration **Long pulse duration**	Requires interception of the command for treatment by the application, otherwise it will return a negative confirmation message (failure).	The output corresponding to the command (ON or OFF) will be turned on for the time corresponding to the duration of the pulse (short or long) configured for the point.	The output corresponding to the command (LOWER or HIGHER) will be turned on for the time corresponding to the pulse duration (short or long) configured for the point.
Persistent output	The output will be turned ON or OFF, and will remain so until a new command, according to the value commanded by the client.	Requires command interception for treatment by the application, otherwise it will return a negative confirmation message (failure).	Requires command interception for treatment by the application, otherwise it will return a negative confirmation message (failure).

Table 171: IEC 60870-5-104 Server Commands Qualifier

4.5.15.7.7. Command Interception

For more information about interception of IEC 60870-5-104 client commands, see the section Interception of Commands from Control Center, implemented through the function block *CommandReceiver*.

4.5.16. IEC 61850 Servidor

This protocol is available for the Hadron Xtorm Series CPU on its Ethernet channels. By selecting this option in MasterTool Xtorm, the CPU becomes the IEC 61850 communication server, allowing the connection with IEC 61850 clients through the MMS protocol and also enabling the sending and receiving of fast messages via the GOOSE protocol.

The following table describes the maximum configuration limits related to the IEC 61850 Server protocol.

	IEC 61850 Server
Ethernet Interfaces Instances (NETs)	1
Instances per CPU	1
Maximum Number of Clients	10
Maximum Number of Logical Nodes	255
Maximum Number of Datasets	100
Maximum Number of Dynamic Datasets	10
Maximum Number of members in a Dataset	300
Maximum Number of GOOSE Control Blocks	100
Maximum Number of GOOSE Subscriptions	100
Maximum Number of Report Control Blocks	100
Maximum Number of Mappings	1.000

Table 172: IEC 61850 Server Protocol Limits

To configure this protocol, the following steps must be performed:

1. Add the IEC 61850 Server protocol instance to one of the available Ethernet channels (NET 1 ... NET 6). To perform this procedure, see section Inserting a Protocol Instance.
2. Configure the Ethernet interface. To do this, see the section Ethernet Interfaces Configuration.
3. Add a Logical Device.

4. Add the desired Logical Nodes, including optional objects and attributes if necessary.
5. Create and configure Datasets, selecting elements from the previously configured Logical Nodes.
6. Create and configure Reports (Buffersized and non-buffered), using the Datasets defined previously.
7. Create and configure the sending of GOOSE messages (GOOSE Publisher), using the Datasets defined previously.
8. Configure the reception of GOOSE (GOOSE Subscriber) messages by importing .CID files.
9. Configure the mapping of RTU communication points to attributes of the Logical Nodes.

The descriptions of each configuration are listed below in this section.

4.5.16.1. Implementation of the IEC 61850 Data Model

One of the main differences of the IEC 61850 standard when compared to the other known protocols (DNP3, MODBUS, etc...) is in the data model used for communication. In IEC 61850, data are set as large symbolic data structures that represent a physical or logical element present in the system. These structures are called Logical Nodes, which are formed by Objects that in turn have a set of Attributes. Finally, Attributes can be simple or structured. The figure below illustrates the hierarchical structure of this data model. The Logical Nodes are represented by the orange symbols, while the objects are green and the attributes are blue:
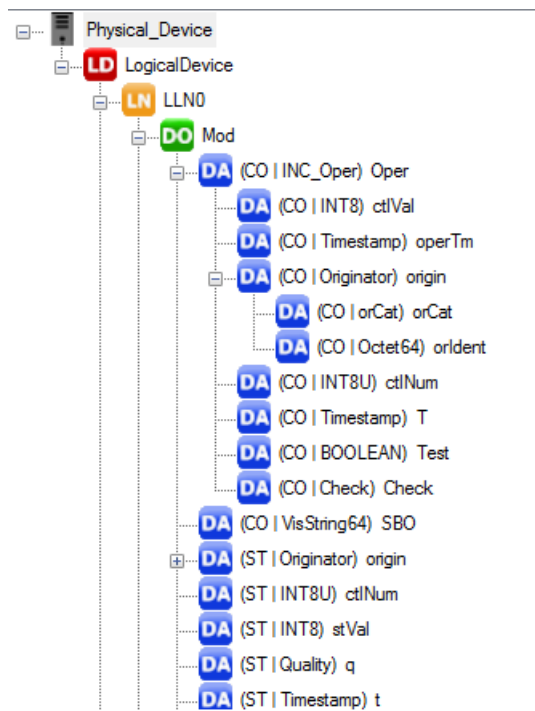
Figure 100: Structured Data Model

Although they are represented in the form of a data structure, Logical Nodes often have an associated algorithm, which can be executed on a specific IED or also distributed among multiple IEDs.

The implementation of this IEC 61850 data structure in the IEC 61131-3 programming environment of MasterTool Xtorm is accomplished through POUs of type Functional Block. Logical Devices, Logical Nodes and Objects are represented by a Functional Block, while Attributes become symbolic variables of elementary types according to IEC 61131-3. All this data are declared in a global variable list (GVL) automatically created by the MasterTool Xtorm tool and are stored in the "IEC61850 Generated POUs" located in the project tree.

The figure below shows an example of the POUs generated from an IEC 61850 configuration:
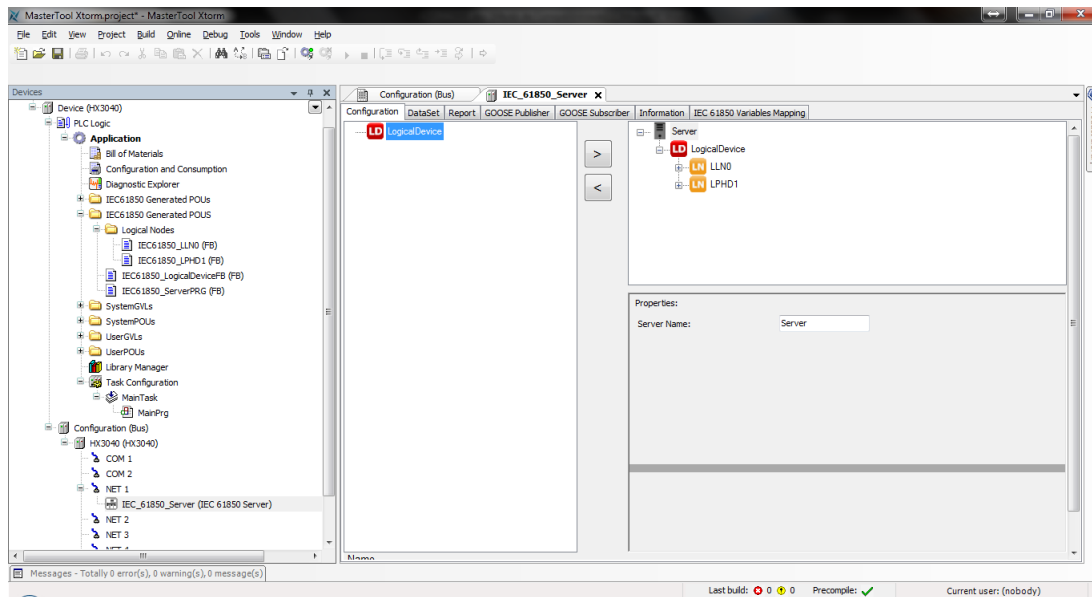
Figure 101: Data Model Implementation

For each instance of a Logical Node present in the IEC 61850 configuration, a corresponding type of Functional Block (stored in the "Logical Nodes" folder) is created. This is necessary because each Logical Node can be customized by adding optional Objects and Attributes.

As mentioned earlier, each Attribute becomes a symbolic variable in the IEC 61131-3 programming environment. The following table shows the correspondence between the IEC 61850 basic data types and the IEC 61131-3 elementary data types:

IEC 61850 Basic Type	IEC 61131-3 Elementary Type
BOOLEAN	BOOL
INT8	SINT
INT16	INT
INT32	DINT
INT8U	USINT
INT16U	UINT
INT32U	UDINT
FLOAT32	REAL
FLOAT64	LREAL
ENUMERATED	INT
CODED ENUM	INT
OCTET STRINGnn VISIBLE STRINGnn UNICODE STRINGnn	STRING(nn)

Table 173: Correspondence of IEC 61850 vs IEC 61131-3 Data Types

The format of *Timestamp* follows the IEC61850 standard, in which it uses a DATE_AND_TIME type structure that is standard in IEC61131-3 and one more variable for the fraction of seconds *dwFracsec*.

The *dwFracsec* variable is a fractional part of a 24-bit number. The least significant bit represents $1/2^{24} = 59.60$ ns. To arrive at the fractional value in decimal, the present value in *dwFracsec* must be divided by 256 (to reduce from 32 to 24 bits) and multiply by 59.60 ns.

The enumerated data types have a set of elements whose values are given a name.

The following table describes the values (in decimal base) that represent in the RTU memory each of the element names:

CDC (Common Data Class)	Attribute	Identifier	Value (decimal)
-	origin.OrCat	not-supported	0
		bay-control	1
		station-control	2
		remote-control	3
		automatic-bay	4
		automaticstation	5
		automatic-remote	6
		maintenance	7
		process	8
ACD	dirGeneral dirPhsA dirPhsB dirPhsC dirNeut	unknown	0
		forward	1
		backward	2
		both	3
BSC	ctlVal	stop	0
		lower	16384
		higher	-32768
		reseved	-16384
CMV	angRef	V	0
		A	1
		Other	2
CURVE	setCharact	ANSI Extremly Inverse	1
		ANSI Very Inverse	2
		ANSI Normal Inverse	3
		ANSI Moderate Inverse	4
		ANSI Definite Time	5
		Long-Time Extremely Inverse	6
		Long-Time Very Inverse	7
		Long-Time Inverse	8
		IEC Normal Inverse	9
		IEC Very Inverse	10
		IEC Inverse	11
		IEC Extremely Inverse	12
		IEC Short-Time Inverse	13
		IEC Long-Time Inverse	14
		IEC Definite Time	15
		Polynom 1 to 16	17 to 32
		Multiline 1 to 16	33 to 48
DPC DPS	stVal	intermediate	0
		off	16384
		on	-32768
		BAD	-16384
SEQ MV CMV	Sev range	normal	0
		high	1
		low	2
		high-high	3
		low-low	4
SEQ	phsRef	A	0
		B	1
		C	2

altus

CDC (Common Data Class)	Attribute	Identifier	Value (decimal)
SEQ	T	pos-neg-zero	0
		dir-quad-zero	1
WYE	angRef	Va	0
		Vb	1
		Vc	2
		Aa	3
		Ab	4
		Ac	5
		Vab	6
		Vbc	7
		Vca	8
		Vother	9
		Aother	10

Table 174: Elements represented in the RTU memory

Due to the flexibility of size of the STRING attributes, the implementation of these elements in the IEC 61131-3 environment is done differently from all others. Instead of storing directly the character set of the string, the symbolic variable that represents the attribute stores only a reference to a table where the string itself is actually located. Thus, all attributes of type string have a suffix "_REF". This reference is actually the index to access a string table implemented in the form of a GVL. There are in total 7 string tables separated by type:

GVL Name (Table)	Description
gaVisSTRING32	VISIBLE STRING32
gaVisSTRING64	VISIBLE STRING64
gaVisSTRING129	VISIBLE STRING129
gaVisSTRING255	VISIBLE STRING255
gaUCSTRING255	UNICODE STRING255
gaOCTED64	OCTET STRING64
gaOCTED255	OCTET STRING255

Table 175: Tables for access to STRING type attributes

For example, look at the configuration in the figure below:
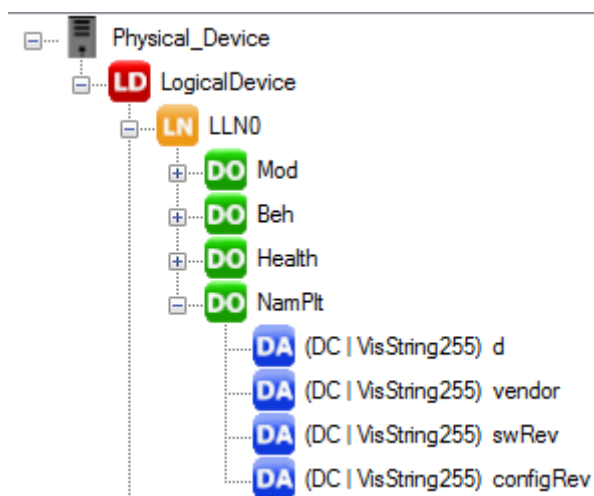
Figure 102: Configuration example with STRING Type attributes

altus

In this example, the Attribute "d" of Object *NamPlt* is of type *VisString255* and its corresponding symbolic variable is:

```
gfbIEC61850_LogicalDevice.LLN0.fbNamPlt.d_REF
```

Thus, to assign the value "TEST" to this attribute in ST language:

```
gaVisSTRING255[gfbIEC61850_LogicalDevice.GGIO4.fbNamPlt.d_REF]:= 'TEST';
```

4.5.16.2. Database Configuration

The configuration of the IEC 61850 Server protocol database is done through the Configuration tab by adding a Logical Device and its respective Logical Nodes:
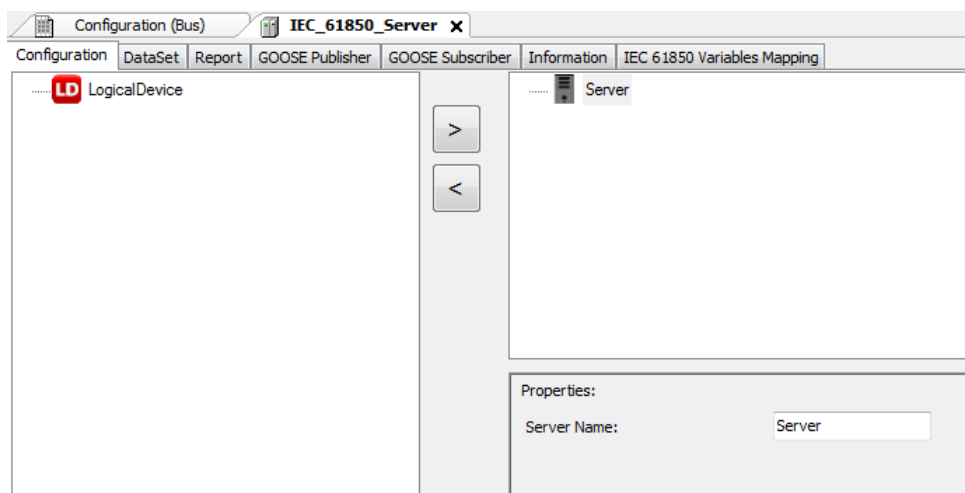
Figure 103: Configuring the Physical Device Name

This screen is divided into two parts. The right side represents the current configuration of the device. On the left side is the list of available objects that can be added/removed according to the object that was selected in the configuration.

The first step on this screen is to define the name of the Physical Device through the *Server Name* field. It is recommended to use a specific name according to the system design where this RTU will be used, because this is the name that will identify this device when configuring the communication with Client MMS devices and also when configuring the GOOSE message exchange with other IEDs.

The next step is to add a Logical Device. To do this, select the *LogicalDevice* object in the left part of the screen and then click the ">" button:

altus
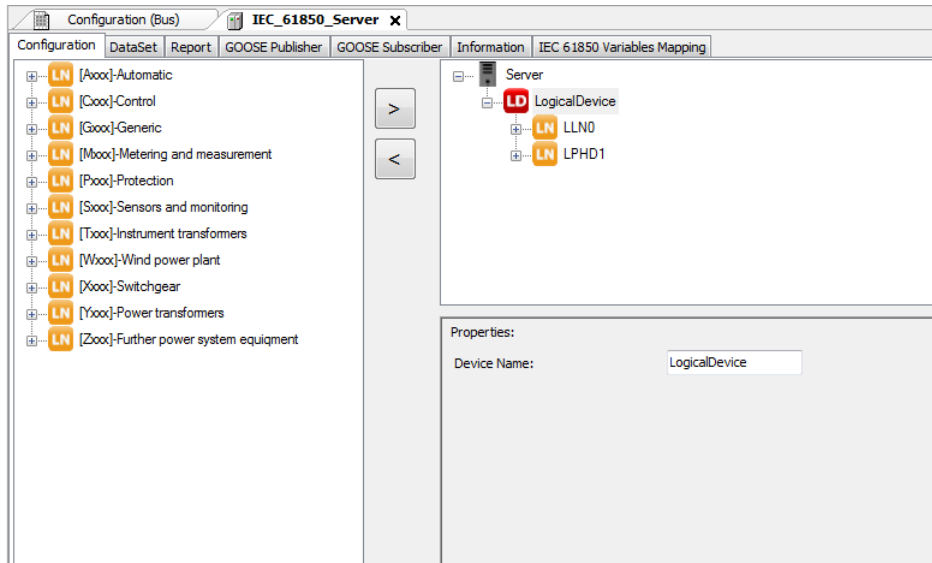
Figure 104: Adding and Configuring the Logical Device Name

The Logical Device is an abstraction level of the data model defined in IEC 61850, and is intended for data organization. The IEC 61850 Server driver supports only one Logical Device, and its name can be changed via the Device Name field. The Logical Device comes by default with two Logical Nodes that store configuration and device information (LLN0 and LPHD1), which cannot be removed.

From this point on, the user can add Logical Nodes as needed by their project. To instantiate new Logical Nodes, simply select it from the list that appears on the left side of the screen:
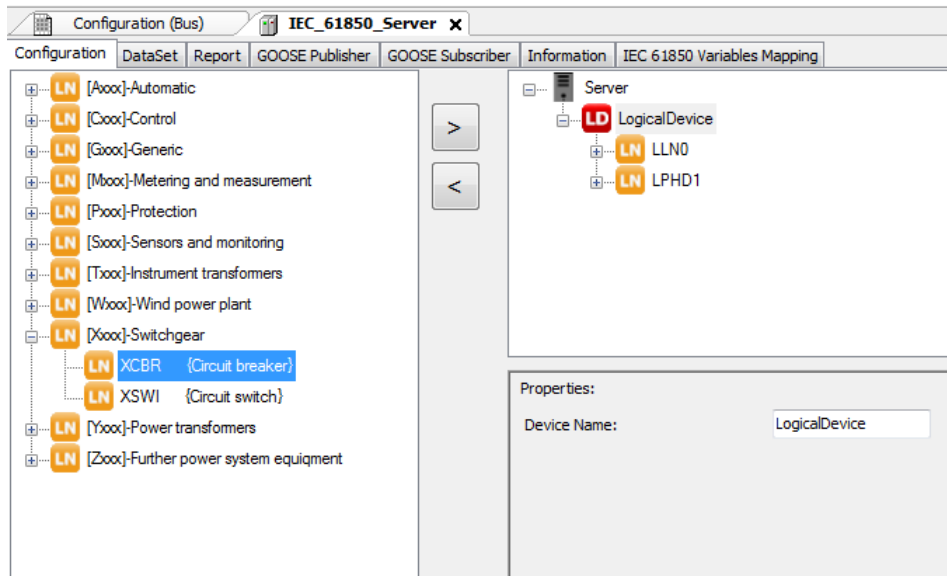
Figure 105: Adding a Logical Node

By clicking on a configured Logical Node, it is possible to modify its prefix and index through the Node Prefix and Logical Node Index fields. In addition, it is also possible to add and remove optional Data Objects. Similarly, by clicking on a Data Object, it is possible to add and remove optional Data Attributes.

Figure 106: Configuring Logical Node

The table below shows all the configuration fields present on this screen, informing their default values and the configuration possibilities:

Configuration	Description	Default	Options
Server Name	Define the Physical Device name.	Server	Set of up to 32 characters (letters, numbers and underline).
Device Name	Define the Logical Device name.	LogicalDevice	Set of up to 32 characters (letters, numbers and underline).
Node Prefix	Define the Logical Node prefix.	-	A set of characters (letters only) whose amount plus the number of characters of the index cannot exceed 07 characters.
Logical Node Index	Define the Logical Node index.	-	A set of characters (numbers only) whose amount plus the number of characters of the prefix cannot exceed 07 characters.

Table 176: Database Parameters Configuration

Notes:

The sum of the size of the **Node Prefix** and **Logical Node Index** fields must not exceed 12 characters.

The limit of the path of an attribute, defined by the sum of the length of the fields **Device Name, Node Prefix, Logical Node Index**, as well as all the separators defined in the standard and the attribute name itself, must not exceed 64 characters.

4.5.16.3. Dataset Configuration

Datasets are lists of variables that are used in communication via GOOSE and MMS messages (Reports, etc...). In other words, they consist of a subset of data that contains references to Attributes and/or Objects of the database configured in the driver.

The configuration of Datasets is done through the Dataset tab:

altus
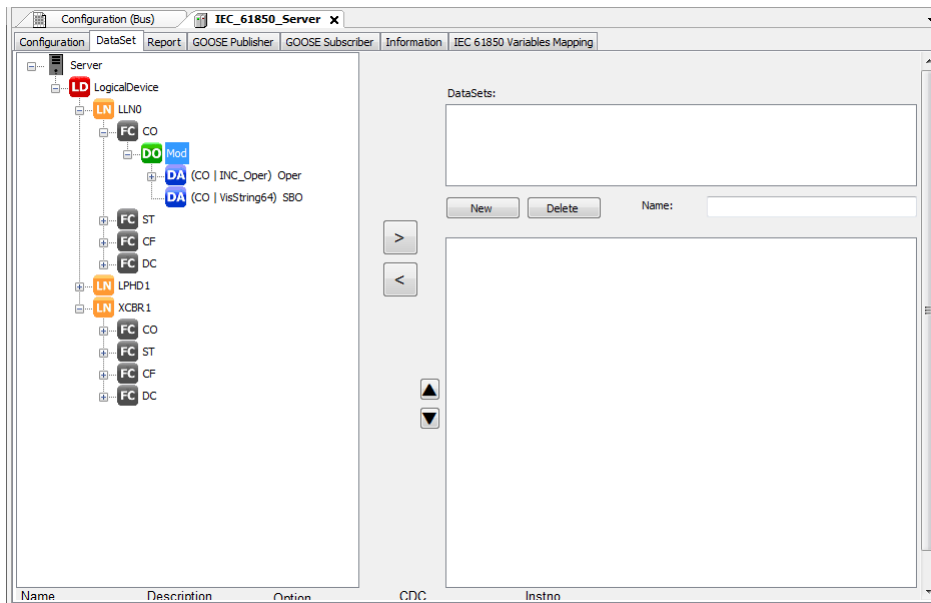
Figure 107: Configuring Datasets

This screen is divided into two parts. On the right side is the list of Datasets currently configured on the device, where there are also the commands to create, remove and rename Datasets. On the left side is the list of available elements that can be added/removed in each Dataset from the database that has been configured.

Additionally, Hadron Xtorm RTU supports the Dynamic Datasets feature, which allows the creation of Datasets in runtime by the IEC 61850 Client (SCADA, etc...). After being created, it is possible to associate them with an existing Report Control Block. The configuration of Dynamic Datasets is volatile, that is, it will be lost in case of power-off or reconfiguration of the RTU.

4.5.16.4. Reports Configuration

The Report is a mechanism for sending a Dataset in an unsolicited manner to an IEC 61850 Client. Its settings (associated Dataset, trigger conditions, etc...) are all stored in an entity called Report Control Block (RCB).

The configuration of the reports is performed through the Report tab:
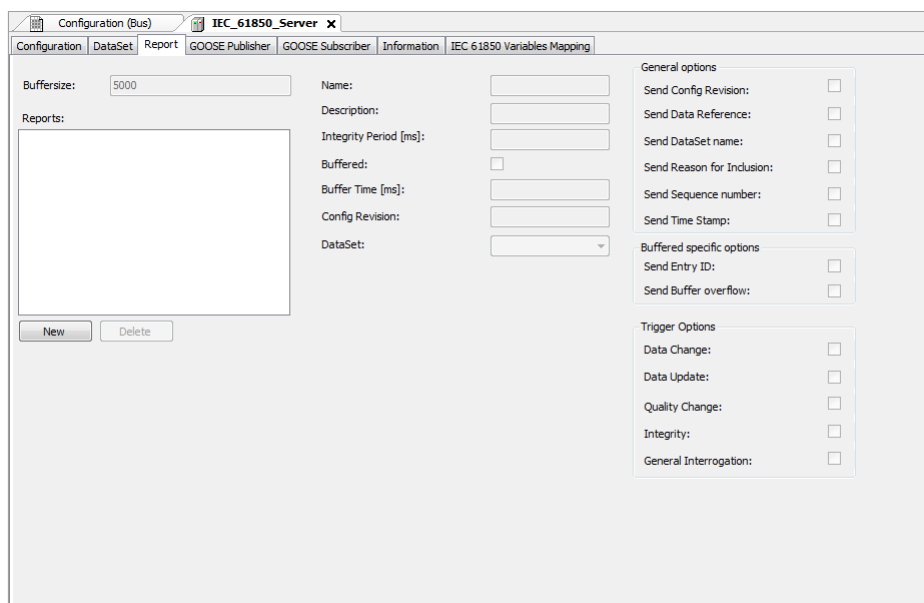
Figure 108: Configuring Reports

altus

On the left side of the screen is the list of RCBs, where there are also commands for creating and removing reports. On the right side are the RCB settings.

The Report operating principle is based on the transmission of information to the MMS Client whenever there is any change in the elements contained in the Dataset of that Control Block. Additionally, the Report can be configured to be sent periodically from the *Integrity Period* parameter. The Report can also be configured in buffered mode, where messages are stored in an internal buffer in the RTU for later sending if the MMS Client is not connected. This buffer has a fixed size of 20Kbytes for each Report Control Block, and the overflow policy is "keep oldest" (that is, if the buffer is full and a new report is generated, the reports stored in the buffer will be kept, and the new report will be discarded).
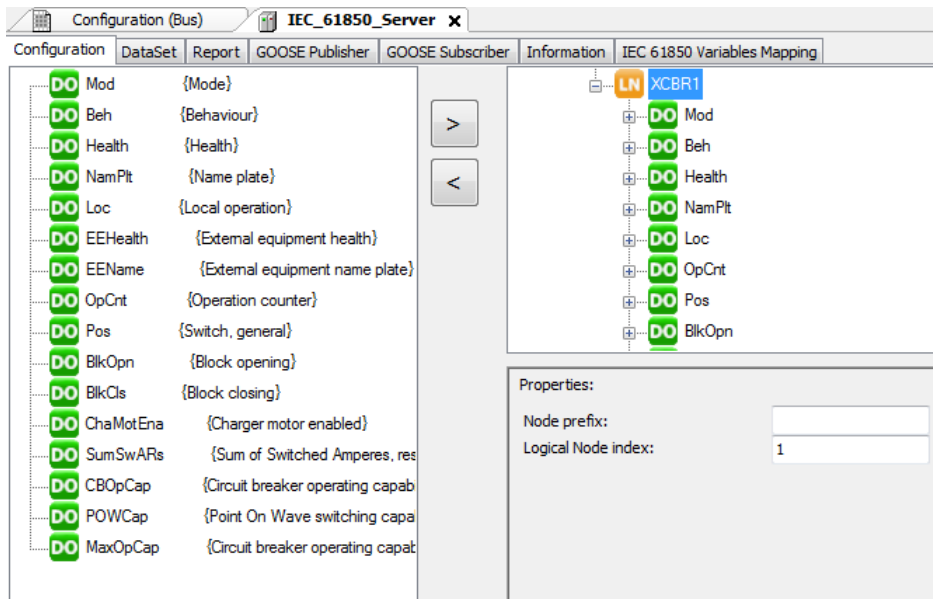
The table below shows all the configuration fields present on this screen, informing their default values and the configuration possibilities:

Configuration	Description	Default	Options
Buffersize	Sets the size (in bytes) of the buffer for storing events. However, this parameter is not supported by the driver, which uses a fixed size of 20 Kbytes for each Buffered Report Control Block regardless of the value set on the screen.	-	-
Name	Name that will identify this Report Control Block for configuration purposes and for communication with other devices.	RCB_n	Set of up to 32 characters (letters, numbers and underscores).
Description	Description of the Report Control Block, to assist in project documentation.	-	Set of up to 80 characters (letters, numbers and underscores).
Integrity Period (ms)	Period in which the Report will be triggered automatically if none of the other trigger conditions have occurred.	0	0 – Disabled (never send) 1 to 65535
Buffered	Selects the Buffers mode, in which Reports are stored internally in the RTU for later sending if the MMS Client is not connected.	Disabled	Habilitado/Disabled
Buffer Time (ms)	Time that will be waited after an event occurs for the Report to be transmitted.	100	0 to 65535
Config. Revision	Number that will be used to identify the revision of the Report Control Block configuration.	1	0 to 65535
Dataset	Dataset that will be monitored and transmitted by this Report Control Block.	-	-
Send Config Revision	If selected, sends the Configuration Revision parameter together with the Report.	Enabled	Enabled/Disabled
Send Data Reference	If selected, sends the data reference along with the Report.	Enabled	Enabled/Disabled
Send Dataset Name	If selected, sends the Dataset name with the Report.	Enabled	Enabled/Disabled
Send Reason for Inclusion	If selected, sends the reason for inclusion with the Report.	Enabled	Enabled/Disabled

altus

Configuration	Description	Default	Options
Send Sequence Number	If selected, sends the sequence number with the Report.	Enabled	Enabled/Disabled
Send Time Stamp	If selected, sends the time stamp with the Report.	Enabled	Enabled/Disabled
Send Entry ID	If selected, sends the Input ID with the Report.	Disabled	Enabled/Disabled
Send Buffer Overflow	If selected, sends the Buffer Overflow information with the Report.	Disabled	Enabled/Disabled
Data Change	If selected, enables the Report trigger condition depending on the value change.	Enabled	Enabled/Disabled
Data Update	Enables the Report to be triggered on update of the variable value (regardless of whether the value has changed). However, this option is not supported by the driver and, if enabled, will have no effect.	-	-
Quality Change	If selected, enables the report trigger condition depending on the quality change.	Enabled	Enabled/Disabled
Integrity	If selected, enables the report trigger condition if Integrity Polling is requested by the MMS Client.	Enabled	Enabled/Disabled
General Interrogation	Enables the Report trigger condition depending on the receipt of a General Interrogation command sent by the MMS Client. However, this option is not supported by the driver and, if enabled, will have no effect.	-	-

Table 177: Reports Configuration Parameters

4.5.16.5. GOOSE Configuration

GOOSE (Generic Object Oriented Substation Events) is a form of communication defined by IEC 61850 designed to enable the fast exchange of information between IEDs. GOOSE communication uses the Publisher/Subscriber model, i.e. messages transmitted by one IED are received by all other devices present in the network, and it is up to them to evaluate if they want to store and use the data contained in the message. GOOSE messages are encapsulated directly over the Ethernet frame using Multicast or Broadcast MAC addresses, and have no relation to the TCP/IP layers.

The configuration of the GOOSE messages to be transmitted by the Xtorm RTU is done in the GOOSE Publisher tab:
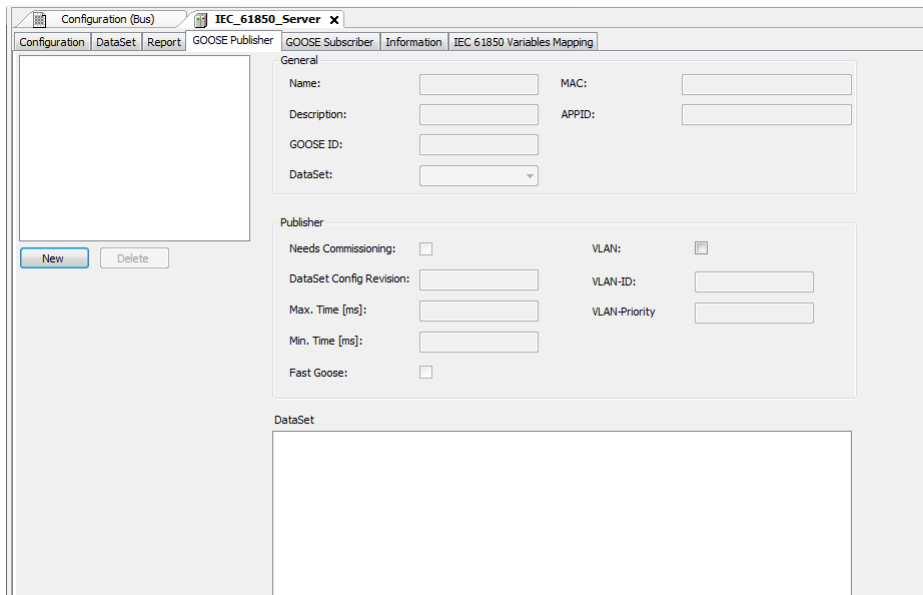
Figure 109: Setting the GOOSE Publisher Communication

The settings of a GOOSE message to be transmitted by the Xtorm RTU (associated Dataset, sending times, etc...) are all stored in an entity called *GOOSE Control Block (GCB)*. On the left side of the screen is the list of GCBs, where there are also commands for creating and removing GCBs. On the right side are the GCB settings. On the bottom of the screen it is possible to see the elements that constitute the Dataset that has been associated with this message.

The operating principle of GOOSE Publisher is based on sending the message when there is any change in the elements contained in the Dataset of that Control Block. After the message is sent, retransmissions are performed using the time base defined by the "Min. Time" parameter. The time interval for the first retransmission is 1X Min. Time, and then it is doubled (2X, 4X, 8X, etc.) until reaching the interval defined by the "Max. Time" parameter. If the Dataset does not present changes within this time, the message is retransmitted continuously with the Max. Time interval (also known as heartbeat).

The dataset comparison and message sending activities performed by each Control Block are associated with an RTU task. If the *Fast Goose* option is enabled, the Control Block performs these activities in the ProtTask, which is a high-priority cyclic task with a very short interval (default is 4ms), thus allowing extremely fast GOOSE messages to be configured to meet performance requirements demanded by critical parts of the system. If this option is not enabled, the Control Block performs these activities in the main RTU task, the MainTask.

> **ATTENTION**
>
> If a Dataset is associated with a GOOSE Control Block whose *Fast Goose* option is enabled, it is not possible to use this same Dataset in a Report Control Block. If this kind of setting is performed, the Report will not work correctly.

The table below shows all the configuration fields present on this screen, informing their default values and the configuration possibilities:

Configuration	Description	Default	Options
Name	Name that will identify this GOOSE Control Block for configuration and communication with other devices.	GCB_n	Set of up to 32 characters (letters, numbers and underline).
Description	Description of the GOOSE Control Block, to assist in project documentation.	-	Set of up to 80 characters (letters, numbers and underline).
GOOSE ID	Also known as AppID, it is a user-defined identification string.	GoCBRef	Set of up to 65 characters (letters, numbers and underline).

altus

Configuration	Description	Default	Options
Dataset	Dataset that will be monitored and transmitted by this GOOSE Control Block.	-	-
MAC	Destination MAC address (multicast).	01-0C-CD-01-00-00	01-0C-CD-01-00-00 to 01-0C-CD-01-01-FF
APP ID	Application identification number. This number can be used to distinguish GOOSE messages sent by different parts of the automation system.	0x0000	0x0000 to 0x3FFF
Needs Comissioning	Should be enabled by the user to indicate that the configuration of this GOOSE Control Block has not yet been completed.	Desabilitado	Enabled/Disabled
Dataset Config Revision	Number that must be used to indicate the revision of the Dataset associated with this GOOSE Control Block.	1	0 to 4294967295
Max. Time (ms)	Maximum time interval for retransmission of a GOOSE message.	4000	0 to 4294967295
Min. Time (ms)	Minimum time interval for retransmission of a GOOSE message.	500	0 to 4294967295
Fast GOOSE	Defines in which task of the RTU this Control Block will be executed. If selected, the comparison of values and sending of the GOOSE message will be performed in the ProtTask. Otherwise, this activity will be performed in the MainTask.	Desabilitado	Enabled/Disabled
VLAN	If selected, enables the use of VLAN in the GOOSE message.	Desabilitado	Enabled/Disabled
VLAN ID	VLAN identification number.	0x0000	0x0000 to 0x0FFF
VLAN Priority	VLAN priority number.	4	0 to 7

Table 178: GOOSE Publisher Configuration Parameters

The configuration of the GOOSE messages to be received by the Xtorm RTU is performed in the GOOSE Subscriber tab by importing SCL (*Substation Configuration Language*) files. These files are generated by the IED configuration tool responsible for sending the message, and internally store the information of the related GOOSE Control Blocks. The import is performed using the button located in the top left corner of the screen:
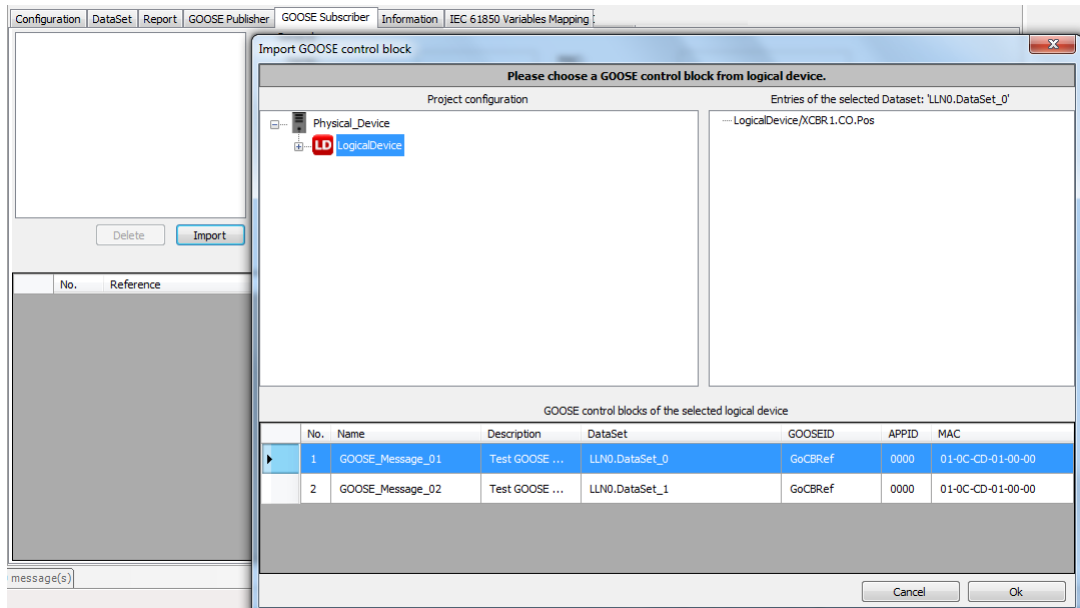
Figure 110: Importing a SCL File

After clicking the import button and selecting the desired SCL file, a screen is opened where a list of Physical Devices and Logical Devices contained in this file is presented. When clicking on a Logical Device, the bottom of the screen shows the list of GOOSE messages (Control Blocks) available to be received. After selecting the desired message to receive, just click *OK*.

Returning to the GOOSE Subscriber tab, the top left corner shows the list of messages set to be received.
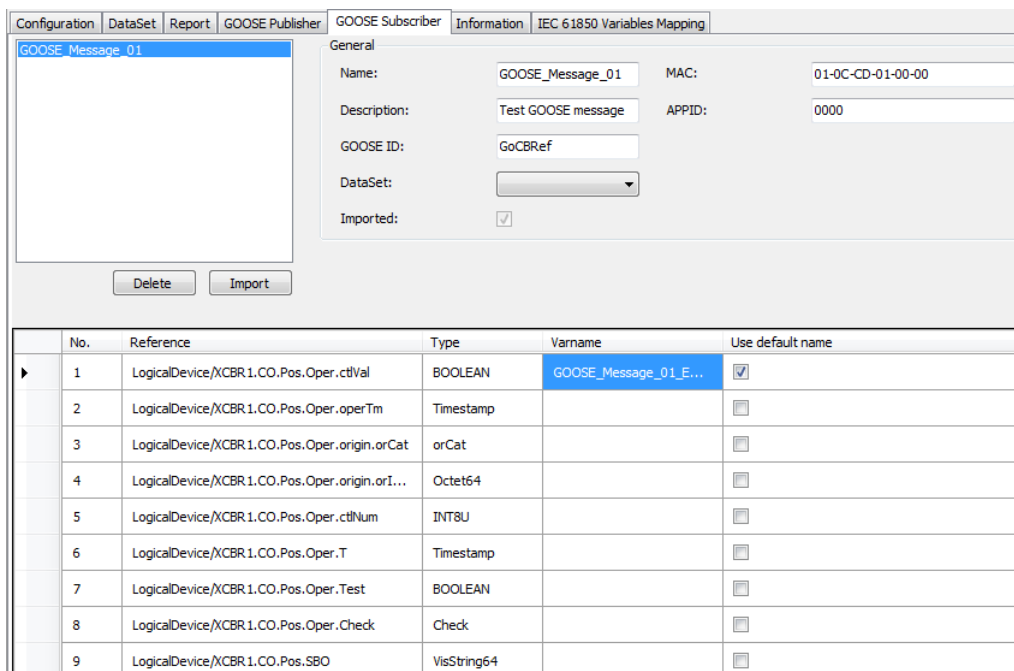
Figure 111: Setting the GOOSE Subscriber Communication

When selecting a message, the information related to the message to be received is displayed on the right side. These fields are automatically filled in by importing the data contained in the GOOSE Control Block associated with the message, so they should not be modified by the user.

In the bottom part of this screen the data contained in the selected message are presented, where it is possible to configure which elements of the received dataset will be effectively stored in the RTU's memory. This mapping is done through the *Varname* field, where the name of the variable to which the data will be copied when the message is received must be informed.

This field can be filled in manually by the user with the name of a variable that already exists in the application, or it can also be filled in automatically using the *Use default name* option. By checking this option, the MasterTool Xtorm tool automatically creates a variable in the IEC61850_Generated_GVL to store the corresponding element received by the GOOSE message. The name of this variable is formed by the name of the GOOSE Control Block adding the suffix "_EntryX", where X is a number indicating the position of the element in the message.

4.5.16.6. Configuration of the IEC 61850 Variable Mapping

The *IEC 61850 Variables Mapping* tab enables the association between internal RTU variables and Logical Node Attributes. This feature is essential to allow the implementation of Logical Nodes from data coming from other communication protocols, since the data types between protocols may not be fully compatible.
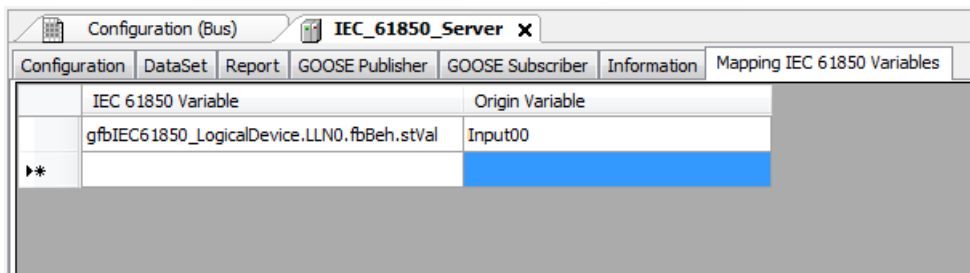
Figure 112: IEC 61850 Variable Mapping

The configuration of the IEC 61850 Variable mapping basically consists of a table with two columns, one containing the variable representing the Attribute of the Logical Node (IEC 61850 Variable) and the other containing the variable of the RTU with which the Attribute will be associated (*Origin Variable*). Each line consists of one mapping, the maximum limit of which is defined at the beginning of this section.

In the example in the figure above, the *stVal* Attribute of the *Beh* object of Logical Node *LLN0* has been mapped to a variable named *Input00*. Since this association may imply a data type conversion, a Origin Variable can only be associated with an Attribute whose type has the same sign (signed/unsigned) and equal or greater size (in bits). The only type that does not follow this rule is DBP, because it consists of a structure of two BOOL variables but even then it can be mapped to Attributes that represent the value of double points and have 8 bits. The table below shows the mapping possibilities between a *Origin Variable* and a *IEC 61850 Variable*:

Origin Variable Type	IEC 61850 Variable Type	
	IEC 61131-3	**IEC 61850**
BOOL	BOOL	BOOLEAN
SINT	SINT, DBP	INT8
INT	INT, DINT, REAL	INT16, INT32, REAL
DINT	DINT, REAL	INT32, REAL
USINT	USINT	INT8U
UINT	DINT, UDINT, REAL	INT32, INT32U, REAL
UDINT	UDINT,REAL	INT32U, REAL
REAL	REAL	FLOAT32
DBP	SINT	INT8

Table 179: Possible Data Types for the IEC 61850 Variable Mapping

If the Origin Variable is associated to a source capable of generating events with its own time stamp (DNP3 IED or digital input module for example), the IEC 61850 Server driver will receive these events and update the attribute "t" (*Timestamp*) of the Object referring to the attribute that was mapped, behaving as a "consumer" as described in section CPU Event Queue.

4.5.16.7. IEC 61850 Server Diagnostics

The configured IEC 61850 Server protocol diagnostics are stored in variables of type *T_DIAG_IEC61850 _SERVER*. The information related to the transmission/reception of GOOSE messages is stored in arrays with dimension corresponding to the

maximum amount that can be configured. Each position of the array has a structure that has information corresponding to a *GOOSE Control Block* or a *Subscription*. The diagnostics are described in the table below.

Diagnostic Variable T_DIAG_IEC61850_SERVER.*	Size	Description
Diagnostics:		
bDriverConfigured	BOOL	Indicates that the driver has been configured correctly.
tGooseGenDiagnostic. uiGooseControlBlocks	UINT	Number of GOOSE Control Blocks (GCBs) configured.
tGooseGenDiagnostic. uiGooseSubscriptions	UINT	Number of GOOSE messages configured to be received.
tGooseControlBlock. tGooseControlBlock[n]. bActive	BOOL	Indicates that this diagnostic structure is active.
tGooseControlBlock. tGooseControlBlock[n]. uiAPPID	UINT	Identification number of the GOOSE Control Block.
tGooseControlBlock. tGooseControlBlock[n]. udiDataChange	UDINT	Number of times any change in the Dataset elements has been detected.
tGooseControlBlock. tGooseControlBlock[n]. udiRetransmission	UDINT	Number of retransmissions of the GOOSE message (it is reset every time there is a Data Change).
tGooseControlBlock. tGooseControlBlock[n]. udiTotal	UDINT	Total number of messages transmitted (includes the initial transmission and all retransmissions).
tGooseSubscription. tGooseSubscription[n]. bActive	BOOL	Indicates that this diagnostic structure is active.
tGooseSubscription. tGooseSubscription[n]. uiAPPID	UINT	ID number of the GOOSE Control Block corresponding to the message configured for reception.
tGooseSubscription. tGooseSubscription[n]. udiReceived	UDINT	Number of receptions of the GOOSE message configured for reception.
tGooseSubscription. tGooseSubscription[n]. udiDataChange	UDINT	Number of times that some change in the Dataset elements was detected when receiving the message.
tGooseSubscription. tGooseSubscription[n]. udiTimeout	UDINT	Indicates that the GOOSE message was not received within the time defined by Time Allowed to Live (TAL).
tGooseSubscription. tGooseSubscription[n]. tTimestamp	T_TIMESTAMP	Time stamp of the moment when some change was detected in the Dataset elements when receiving the message.

Diagnostic Variable T_DIAG_IEC61850_SERVER.*	Size	Description
tReportControlBlock. tReportControlBlock[n]. sRCB_NAME	STRING(65)	Report Control Block name.
tReportControlBlock. tReportControlBlock[n]. sIP_ADDR	STRING(15)	IP address of the IEC 61850 Client that enabled and/or reserved this Report.
tReportControlBlock. tReportControlBlock[n]. bEnabled	BOOL	Indicates that this Report has been enabled by an IEC 61850 Client.
tReportControlBlock. tReportControlBlock[n]. bResv	BOOL	Indicates that this Report was reserved by an IEC 61850 Client (applies only to Buffered Reports).
tReportControlBlock. tReportControlBlock[n]. bBuffered	BOOL	Indicates that this Report is of type Buffered.
tReportControlBlock. tReportControlBlock[n]. byOccupation	BYTE	Reports the internal buffer occupancy (from 0 to 100, in %, applicable only for Buffered Reports).
tConnectionStatus. tConnectionStatus[n]. bActive	BOOL	Indicates that a connection is established with an IEC 61850 Client.
tConnectionStatus. tConnectionStatus[n]. sIP_ADDR	STRING(15)	IP address of the IEC 61850 Client that is connected.
tConnectionStatus. tConnectionStatus[n]. udiFramesTx	UDINT	Reports the number of frames transmitted in the communication with the IEC 61850 Client that is connected.
tConnectionStatus. tConnectionStatus[n]. udiFramesRx	UDINT	Reports the number of frames received in communication with the IEC 61850 Client that is connected.

Table 180: IEC 61850 Server Diagnostics

4.5.16.8. Commands Interception

The interception of commands received by the IEC 61850 Server driver is accomplished through the *CommandReceive* function block as described in the Interception of Commands from Control Center section. This feature allows the interception of selection and operation commands for the following CDCs (Common Data Classes): SPC, DPC, INC, BSC, ISC.

Below is an example of an application using the ST language showing the interception of a command to the *BlkOpn* object of an XCBR Logical Node, responding *"SUCCESS"* to the IEC 61850 Client.

```
PROGRAM UserPrg
VAR
  CRReceive: LibRtuStandard.CommandReceiver;
  CCommand:  LibRtuStandard.COMMAND_T;
  CRResult:  LibRtuStandard.COMMAND_RESULT:= LibRtuStandard.COMMAND_RESULT.
   SUCCESS;
END_VAR
```

altus

```
CRReceive.dwVariableAddr:= ADR(gfbIEC61850_LogicalDevice.XCBR1.fbBlkOpn.Oper.
    ctlVal);
CRReceive.bExec:= TRUE;
CRReceive.dwTimeout:= 500;
IEC61850_Generated_GVL.gfbIEC61850_LogicalDevice.XCBR1.fbBlkOpn.ctlModel :=
    IEC61850ServerXtorm.tyIEC61850_AT_ENUM_CtlModels.
    IEC61850_AT_CTLModels_DIRECT_WITH_NORMAL_SEC;

// When a command is captured:
IF CRReceive.bCommandAvailable THEN
  // Save its data
  CCommand:= CRReceive.sCommand;
  //  Answers 'SUCCESS'
  CRReceive.eCommandResult:= CRResult;
  CRReceive.bDone:= TRUE;
END_IF

CRReceive();

IF CRReceive.bDone THEN
  CRReceive.bDone:= FALSE;
END_IF
```

4.5.17. PROFINET Controller

For correct use of the PROFINET Controller protocol, it is necessary to consult the manual MU214621 - Nexto Series PROFINET Manual .

4.6. Communication Performance

4.6.1. MODBUS Communication

The configurable MODBUS devices in Hadron Xtorm CPU run in the background, with a priority below the user application and in a cyclical way. Therefore, their performance will vary according to the remaining time, taking into account the difference between the interval and the time the application takes to run. For example, a MODBUS device in an application that runs every 100 ms, with a 50 ms runtime, will have a lower performance than an application of 25 ms running every 100 ms interval. This is because, in the second case, the CPU will have a longer time between each MainTask cycle to perform the lower priority tasks.

The number of cycles it takes for the device, slave or server, to respond to a request must also be taken into account.

To process and transmit a response, a MODBUS RTU Slave will take two cycles (MODBUS task cycle time), while a MODBUS Ethernet Server will only take one cycle. However, this is the minimum time between receiving a request and sending the response. If the request is sent just after the execution of a MODBUS task cycle, the time may be equivalent to 2 or 3 times the cycle time for Slave MODBUS, and 1 to 2 times the cycle time for Server MODBUS.

In this case: Maximum Response Time = 3 * (cycle time) + (time of execution of the tasks) + (time interframe chars) + (send delay).

For example, for a MODBUS Ethernet Server task with a 50 ms cycle, in an application running for 60 ms every 100 ms, the server will only be able to execute one cycle between each cycle of the application. On the other hand, with the same application, being executed for 10 ms, but with a 100 ms interval, the MODBUS will perform better, because while the application is not running, it will be executing every 50 ms and only with each cycle of the MainTask it will take longer to execute. For these cases, the worst performance will be the sum of the user application execution time and the cycle time of the MODBUS task.

For MODBUS RTU Master and MODBUS Ethernet Client devices the operating principle is exactly the same, but it takes into account the configured scan time (ms) in the MODBUS relation and not the cycle time of the MODBUS task. For these cases, the cycle time of the MODBUS task is specified as follows:

altus

MODBUS RTU Master

- Cycle time of the task = ((Minimum Scan Time (ms) configured in the relations / 5).

If the cycle time of the task, calculated according to the expression above, is greater than the value set in the *Communication Time-out* field available in the *Advanced Settings* tab of the Device, the Cycle Time of the task will be equal:

- Cycle time of the task = Communication time-out (ms) configured in the device.

MODBUS Ethernet Client

- Cycle time of the task = (Minimum Scan Time (ms) configured in relations / 3).

If the cycle time of the task, calculated according to the expression above, is greater than the value set in the *Communication Time-out* field available in the *Advanced Settings* tab of the Device, the Cycle Time of the task will be equal:

- Cycle time of the task = Communication time-out (ms) configured on the device.

For these cases, the worst performance of a relation will be to be executed after its scan time, added to the runtime of the user application.

Notice that the number of running MODBUS devices also affects its performance. In an user application with 60 ms runtime and 100 ms interval, will leave only 40 ms for the CPU to perform all other lower priority tasks. Therefore, a CPU with a single MODBUS Ethernet Server will have a better performance than the one using four of these devices.

4.6.1.1. MODBUS Server

A MODBUS Ethernet Server device is able to respond to a given number of requests per second, that is, it is able to transfer n bytes per second, depending on the size of each request. The smaller the task cycle of the MODBUS Server, the greater the impact of the number of connections in its response rate. However, for cycle times lower than 20 ms this impact is not linear.

> **ATTENTION**
>
> The communication performances mentioned in this chapter are examples using a CPU with only one MODBUS TCP Server device, without any logic in the application that could delay the communication. Therefore, these performances should be taken as maximums.

For cycle times equal or larger than 20 ms, the increase of the response rate is linear, and can be calculated using the formula:

$N = - C \times (Z - (Z / (T \times 1000)))$

$Z = 1 / T$

"N" is the average number of responses per second, "C" is the number of connections and T is the cycle time of the MODBUS task (in seconds)

Taking the example of a MODBUS Server with connection and cycle time of 50 ms:

$C = 1; T = 0.05$ s; $Z = 1 / 0.05 = 20;$

$N = 1 \times (20 - (20 / (0.05 \times 1000))) = 1 \times (20 - (20 / 50)) = 1 \times (20 - 0.4) = 1 \times 19.6$

$N = 19.6$

Thus, in this configuration, the MODBUS Server will is able to respond 19 requests per second (average)

If this obtained value is multiplied by the number of bytes in each request, a transfer rate of n bytes per second will be obtained.

4.6.2. DNP3 Communication

4.6.2.1. DNP3 Server

The CPU runs the DNP3 Server driver in the same way as other communication Server drivers, i.e., in background, cyclically, and with a priority below the user application. The task of this driver specifically runs at every 50ms, and just 1 run cycle of the driver is enough to process and respond requests. However, due to the fact it is a low priority task, it may not be able to run with the same frequency, as it depends on the percentage of free CPU and also on the competition with tasks of other protocols configured on the CPU. The percentage aforementioned is given by the difference between the MainTask interval and the time that the user application takes to run.

To help in the comprehension of the DNP3 Server driver performance, check below the result of tests carried out with a DNP3 Client simulator, connected to a HX3040 with a DNP3 Server driver. The configured database consisted of 4000 digital points and 500 analog points (all with quality and time stamp). The MainTask execution time was 70ms (interval of 100 ms).

- Time to complete an integrity polling: less than one second.
- Time to transfer 4.000 digital events + 500 analog events: 4 seconds.

altus

4.6.2.2. DNP3 Client

As the DNP3 Server, the CPU also runs the DNP3 Client driver in background and cyclically, at each 50ms. Its priority is below the user application and it needs 3 run cycles to accomplish the remaining requests. Unlike the MODBUS Client driver, this task period is fixed in 50 ms, regardless of the polling period set in the requests. Thus, the polling period effectively executed by the driver may have an inaccuracy of up to 150ms.

4.6.3. IEC 60870-5-104 Communication

4.6.3.1. IEC 60870-5-104 Server

The CPU runs the IEC 60870-5-104 Server driver in the same way as other communication Server drivers, i.e., in background, cyclically, and with a priority below the user application. The task of this driver specifically runs at every 50ms, and just 1 run cycle of the driver is enough to process and respond requests. However, due to the fact it is a low priority task, it may not be able to run with the same frequency, as it depends on the percentage of free CPU and also on the competition with tasks of other protocols configured on the CPU. The percentage aforementioned is given by the difference between the MainTask interval and the time that the user application takes to run.

To help in the comprehension of the IEC 60870-5-104 Server driver performance, check below the result of tests carried out with a IEC 60870-5-104 Client simulator, connected to a HX3040 with a IEC 60870-5-104 Server driver. The configured database consisted of 4000 digital points and 500 analog points (all with quality and time stamp). The MainTask execution time was 70ms (interval of 100 ms).

- Time to complete an General Interrogation command: less than one second.
- Time to transfer 4.000 digital events + 500 analog events: 25 seconds.

4.6.3.2. IEC 60870-5-104 Client

The IEC 60870-5-104 Client driver is also executed by the CPU in the background, with a priority below the user application, cyclically every 50ms.

4.7. System Performance

In cases where the application has only one user task MainTask, responsible for the execution of a single programming unit of type Program called MainPrg, the CPU consumes a certain amount of time for the task to be processed. This time is called *Execution Time*.

In an application, we can find out the average Runtime of the application using MasterTool Xtorm, in the Device Tree, under Device, in the following path:

Device –> Application –> Task Configuration (Monitor tab), Average Cycle Time column.

Attention should be paid to the Execution Time so that it does not exceed 80 % of the interval set in the MainTask. For example, in an application where the interval is 100 ms, an adequate Execution Time is up to 80 ms. This is because the CPU needs time for the execution of other tasks such as communication processing, display and memory card handling, and these tasks also take place within the range (the remaining 20 % of the Execution Time).

MasterTool Xtorm also has the option to limit the Execution Time of the MainTask, so that the Execution Time mentioned above is respected. This is done by consisting the Watchdog of the user task in 80% of the interval.

If the user is aware of the above information and wants to use this Watchdog consistency, this will be possible by checking the *Generate error on task watchdog consistency* checkbox in the CPU's configuration tab. However, it is recommended to keep the watchdog time configured with the default of 1000 ms and the checkbox to generate error in the consistency unchecked.

> **ATTENTION**
>
> If the percentage of 80% is not respected and the user task execution time approaches or exceeds the configured MainTask interval, the display and diagnostic button may not respond, since their priority in system execution is lower than user tasks. If an application with errors is loaded on the CPU, it may be necessary to reboot the CPU without loading this application, as described in section Not Loading the Application at Startup.

4.7.1. I/O Scanning

The updating of the inputs and outputs associated to the bus modules is performed primary on the main task of the RTU, called MainTask, according to the sequence described below:

1. Inputs Reading;
2. Outputs Writing;
3. User application execution;
4. Wait to complete the interval set for the MainTask.

As described above, the total cycle time is the result of those parts: Inputs Reading, Outputs writing and User application execution. Thus, the application runtime is affected by the amount of bus input/output modules. The length of the I/O update part is approximately 1 ms for each 20 modules integrated into the project. The last MainTask step (waiting to complete the set range) is the stage where the communication drivers and all other low-priority services run

Additionally, it is possible to configure the RTU to perform the update of modules inputs and outputs by using the option called *"Enable I/O update per task"*, located at the CPU configurations. When this option is enabled, the CPU performs an access to the local bus on the context of each task for updating the inputs and outputs that are being used on that task. This option can be very useful in cases that ProtTask is used and it is necessary to have a fast response of inputs and outputs (protection logics for example). The sequence of the actions is slightly different than the MainTask, as it is described below:

1. User application execution
2. Outputs Writing
3. Inputs Reading
4. Wait to complete the interval set for the MainTask

> **ATTENTION**
>
> Even if an I/O point is used and updated in other tasks, with the option *Enable I/O update per task*, it will continue to be updated in MainTask too; except when all the module's I//O points are used and updated in another task, in this case will no longer be updated by MainTask

> **ATTENTION**
>
> The selection list *"Always update variables"* defines the updating of I/O variables:
>
> - *Use parent device configuration*: update according to the configuration of the parent device.
> - *Activate 1 (use the bus cycle task if it is not used in another task)*: I/O variables are updated in the bus cycle task if they are not used in any other task.
> - *Activate 2 (always in the bus cycle task)*: all variables are updated at every bus task cycle, regardless of whether they are used and whether they are mapped to an input or output channel.

4.7.2. Memory Card

Data transfers involving the memory card is performed by the CPU in the background, as this gives priority to the execution of user application and communication processing. Thus, the transfer of files to the card may suffer an additional significant time, depending on the Cycle Time of the user application.

The time required to read/write files on the card will be directly affected by the Cycle Time of the user application since this application has priority in execution.

Further information about the use of the memory card see Memory Card section.

4.8.   RTC Clock

The CPUs have an internal clock that can be used through the *NextoStandard.lib* library. This library is automatically loaded during the creation of a new project (to perform the library insertion procedure, see Libraries section). The figure below shows how to include the blocks in the project:
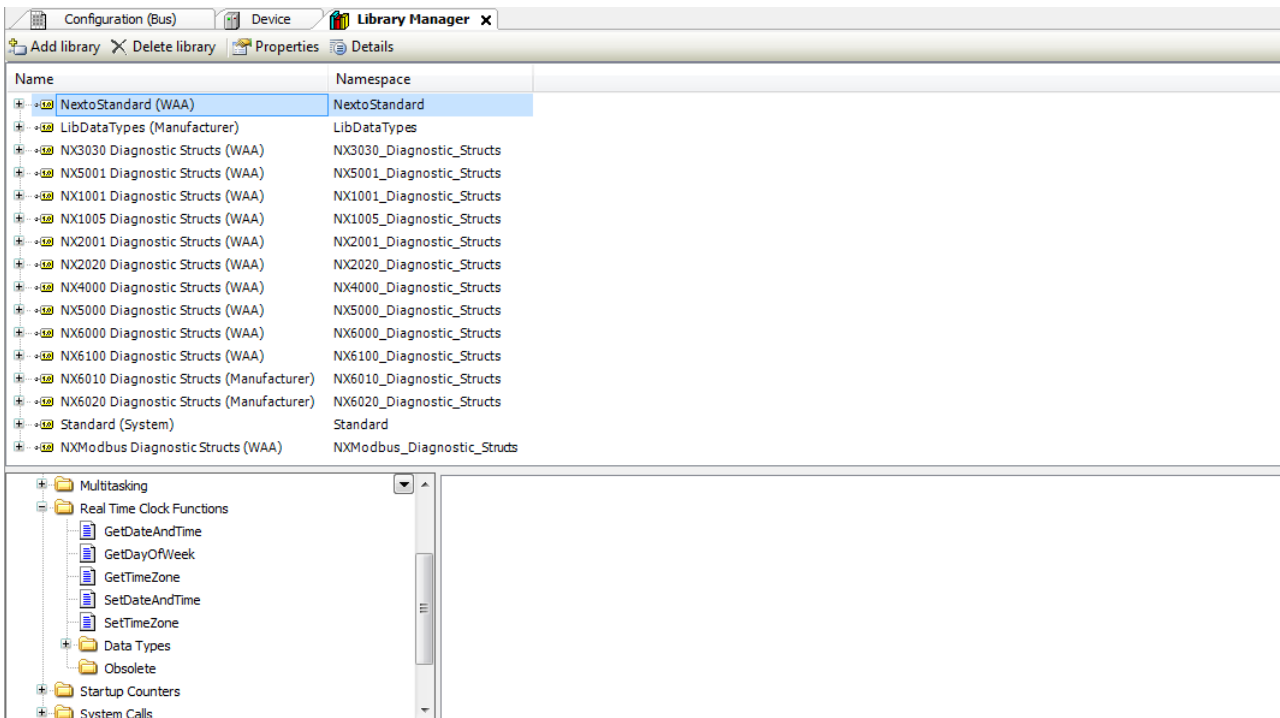
Figure 113: Clock Reading and Writing Blocks

> **ATTENTION**
>
> Function blocks of RTC Reading and Writing, previously available in 2.00 MasterTool IEC XE or older become obsolete from 2.00 or newer, the following blocks are no longer used: *NextoGetDateAndTime*, *NextoGetDateAndTimeMs*, *NextoGetTimeZone*, *NextoSetDateAndTime*, *NextoSetDateAndTimeMs* and *NextoSetTimeZone*.

4.8.1. Function Blocks for RTC Reading and Writing

Among other function blocks, there are some very important used for clock reading (*GetDateAndTime*, *GetDayOfWeek* and *GetTimeZone*) and for date and time new data configuring (*SetDateAndTime* and *SetTimeZone*). These functions always use the local time, that is, take into account the value defined by the *Time Zone*.

The proceedings to configure these two blocks are described below.

> **ATTENTION**
>
> The obsolete function blocks for reading and writing the RTC (*NextoGetDateAndTime*, *NextoGetDateAndTimeMs*, *NextoSetDateAndTime* and *NextoSetDateAndTimeMs*) cannot be used in the redundant data area in redundant projects . They must be used in non-redundant POUs, such as the *NonSkippedPrg* POU. More details on how the POU *NonSkippedPrg* works can be found in NonSkippedPrg Program.

4.8.1.1. Function Blocks for RTC Reading

The clock reading can be made through the following functions:

4.8.1.1.1. GetDateAndTime

GetDateAndTime function is used to read the date and time in miliseconds.

altus
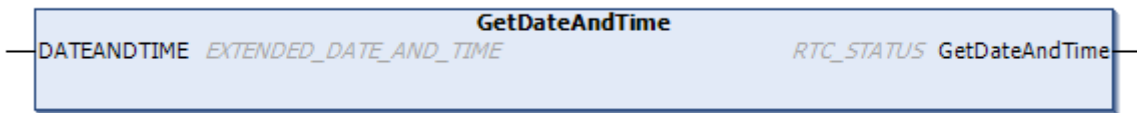
Figure 114: Date and Hour Reading

Input Parameters	Type	Description
DATEANDTIME	EXTENDED_DATE _AND_TIME	This variable returns the value of date and hour of RTC in the format shown at Table 190.

Table 181: Input Parameters of GetDateAndTime

Output Parameters	Type	Description
GETDATEANDTIME	RTC_STATUS	Returns the function error state, see Table 192.

Table 182: Output Parameters of GetDateAndTime

Utilization example in ST language:

```
PROGRAM UserPrg
VAR
Result : RTC_STATUS;
DATEANDTIME : EXTENDED_DATE_AND_TIME;
xEnable : BOOL;
END_VAR
--------------------------------------------------------------------------
IF xEnable = TRUE THEN
Result := GetDateAndTime(DATEANDTIME);
xEnable := FALSE;
END_IF
```

4.8.1.1.2. GetTimeZone

The following function reads the Time Zone configuration, this function is directly related with time in Time Zone at SNTP synchronism service:
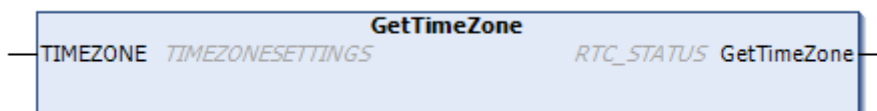
Figure 115: Configuration Reading of Time Zone

Input Parameters	Type	Description
TIMEZONE	TIMEZONESETTINGS	This variable presents the reading of Time Zone configuration.

Table 183: Input Parameters of GetTimeZone

Output Parameters	Type	Description
GetTimeZone	RTC_STATUS	Returns the function error state, see Table 192.

Table 184: Output Parameters of GetTimeZone

Utilization example in ST language:

```
PROGRAM UserPrg
VAR
GetTimeZone_Status  : RTC_STATUS;
TimeZone      : TIMEZONESETTINGS;
xEnable : BOOL;
END_VAR
----------------------------------------------------------------------
IF xEnable = TRUE THEN
GetTimeZone_Status := GetTimeZone(TimeZone);
xEnable := FALSE;
END_IF
```

4.8.1.1.3. GetDayOfWeek

GetDayOfWeek function is used to read the day of the week.
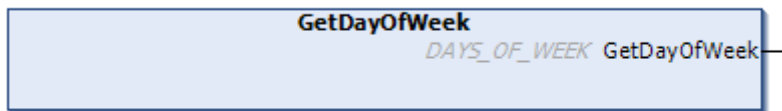
Figure 116: Day of Week Reading

Output Parameters	Type	Description
GetDayOfWeek	DAYS_OF_WEEK	Returns the day of the week. See Section 191.

Table 185: Output Parameters of GetDayOfWeek

When called, the function will read the day of the week and fill the structure *DAYS_OF_WEEK*.
Utilization example in ST language:

```
PROGRAM UserPrg
VAR
DayOfWeek : DAYS_OF_WEEK;
END_VAR
-----------------------------------------------------------------------
DayOfWeek := GetDayOfWeek();
```

4.8.1.2. RTC Writing Functions

The clock settings are made through function and function blocks as follows:

4.8.1.2.1. SetDateAndTime

SetDateAndTime function is used to write the settings on the clock. Typically the precision is on the order of hundreds of milliseconds.
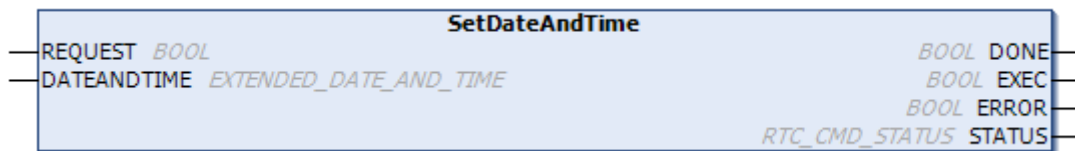
Figure 117: Set Date And Time

Input parameters	Type	Description
REQUEST	BOOL	This variable, when receives a rising edge, enables the clock writing.
DATEANDTIME	EXTENDED_DATE _AND_TIME	Receives the values of date and hour with milliseconds. See section 190.

Table 186: Input Parameters of SetDateAndTime

Output parameters	Type	Description
DONE	BOOL	This variable, when true, indicates that the action was successfully completed.
EXEC	BOOL	This variable, when true, indicates that the function is processing the values.
ERROR	BOOL	This variable, when true, indicates an error during the Writing.
STATUS	RTC_STATUS	Returns the error occurred during the configuration. See Table 192.

Table 187: Output Parameters of SetDateAndTime

When a rising edge occurs at the *REQUEST* input, the function block will write the new *DATEANDTIME* values on the clock. If the writing is successfully done, the *DONE* output will be equal to *TRUE*. Otherwise, the *ERROR* output will be equal to *TRUE* and the error will appear in the *STATUS* variable.

Utilization example in ST language:

```
PROGRAM UserPrg
VAR
SetDateAndTime : SetDateAndTime;
xRequest : BOOL;
DateAndTime : EXTENDED_DATE_AND_TIME;
xDone : BOOL;
xExec : BOOL;
xError : BOOL;
xStatus : RTC_STATUS;
END_VAR
--------------------------------------------------------------------------
IF xRequest THEN
  SetDateAndTime.REQUEST:=TRUE;
  SetDateAndTime.DATEANDTIME:=DateAndTime;
  xRequest:= FALSE;
END_IF
SetDateAndTime();
SetDateAndTime.REQUEST:=FALSE;
IF SetDateAndTime.DONE THEN
  xExec:=SetDateAndTime.EXEC;
  xError:=SetDateAndTime.ERROR;
  xStatus:=SetDateAndTime.STATUS;
END_IF
```

ATTENTION

If you try to write time values outside the range of the RTC, the values are converted to valid values, provided they do not exceed the valid range of 01/01/2000 to 12/31/2035. For example, if the user attempts to write the value 2000 ms, it will be converted to 2 seconds, write the value 100 seconds, it will be converted to 1 min and 40 seconds. If the type value of 30 hours, it is converted to 1 day and 6 hours, and so on.

4.8.1.2.2. SetTimeZone

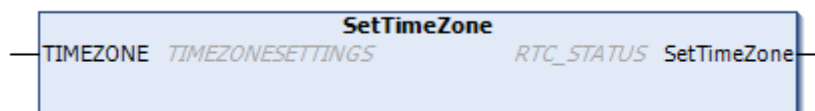The following function block makes the writing of the time zone settings:

Figure 118: Writing of the Time zone Settings

Input parameters	Type	Description
TIMEZONE	TIMEZONESETTINGS	Structure with time zone to be configured. See Table 193.

Table 188: SetTimeZone Input Parameters

Output parameters	Type	Description
SetTimeZone	RTC_STATUS	Returns the error occurred during the reading/setting. See Table 192.

Table 189: SetTimeZone Output Parameters

When called, the function will configure the *TIMEZONE* with the new system time zone configuration. The configuration results is returned by the function.

Utilization example in ST language:

```
PROGRAM UserPrg
VAR
Status : RTC_STATUS;
TimeZone : TIMEZONESETTINGS;
xWrite : BOOL;
END_VAR
--------------------------------------------------------------------------
//FB SetTimeZone
IF (xWrite = TRUE) THEN
Status := SetTimeZone(TimeZone);
  IF Status = RTC_STATUS.NO_ERROR THEN
    xWrite := FALSE;
  END_IF
END_IF
```

> **ATTENTION**
>
> To perform the clock should be used time and date values within the following valid range: 00:00:00 hours of 01/01/2000 to 12/31/2035 23:59:59 hours, otherwise , is reported an error through the *STATUS* output parameter. For details of the *STATUS* output parameter, see the section RTC_STATUS.

4.8.2. RTC Data Structures

The reading and setting function blocks of the Nexto Series CPUs RTC use the following data structures in its configuration:

4.8.2.1. EXTENDED_DATE_AND_TIME

This structure is used to store the RTC date when used the function blocks for date reading/setting within milliseconds of accuracy. It is described in the table below:

altus

Structure	Type	Variable	Description
EXTENDED_DATE_ AND_TIME	BYTE	byDayOfMonth	Stores the day of the set date.
	BYTE	ByMonth	Stores the month of the set date.
	WORD	wYear	Stores the year of the set date.
	BYTE	byHours	Stores the hour of the set date.
	BYTE	byMinutes	Stores the minutes of the set date.
	BYTE	bySeconds	Stores the seconds of the set date.
	WORD	wMilliseconds	Stores the milliseconds of the set date.

Table 190: EXTENDED_DATE_AND_TIME

4.8.2.2. DAYS_OF_WEEK

This structure is used to store the day of week:

Enumerable	Value	Description
DAYS_OF_WEEK	0	INVALID_DAY
	1	SUNDAY
	2	MONDAY
	3	TUESDAY
	4	WEDNESDAY
	5	THURSDAY
	6	FRIDAY
	7	SATURDAY

Table 191: DAYS_OF_WEEK Structure

4.8.2.3. RTC_STATUS

This enumerator is used to return the type of error in the RTC setting or reading and it is described in the table below:

Enumerator	Value	Description
RTC_STATUS	NO_ERROR (0)	There is no error.
	UNKNOWN_COMMAND (1)	Unknown command.
	DEVICE_BUSY (2)	Device is busy.
	DEVICE_ERROR (3)	Device with error.
	ERROR_READING_OSF (4)	Error in the reading of the valid date and hour flag.
	ERROR_READING_RTC (5)	Error in the date and hour reading.
	ERROR_WRITING_RTC (6)	Error in the date and hour writing.
	ERROR_UPDATING_SYSTEM _TIME (7)	Error in the update of the system's date and hour.
	INTERNAL_ERROR (8)	Internal error.
	INVALID_TIME (9)	Invalid date and hour.
	INPUT_OUT_OF_RANGE (10)	Out of the limit of valid date and hour for the system.

altus

Enumerator	Value	Description
	SNTP_NOT_ENABLE (11)	Error generated when the SNTP service is not enabled and it is done an attempt for modifying the time zone.

Table 192: RTC_STATUS

4.8.2.4. TIMEZONESETTINGS

This structure is used to store the time zone value in the reading/setting requests of the RTC's function blocks and it is described in table below:

Structure	Type	Variable	Description
TIMEZONESETTINGS	INT	iHour	Set time zone hour.
	INT	iMinutes	Set time zone minute.

Table 193: TIMEZONESETTINGS

Note:

Function Blocks of Writing and Reading of Date and Hour: different libraries of *NextoStandard*, which have function blocks or functions that may perform access of reading and writing of date and hour in the system, are not indicated. The *NextoStandard* library has the appropriate interfaces for writing and reading the system's date and hour accordingly and for informing the correct diagnostics.

4.9. User Files Memory

Hadron Xtorm Series CPUs have a memory area destined to general data storage, in other words, the user can store several project files of any format in the CPU memory. This memory area varies according to the CPU model. See section Memory.

In order to use this area, access the project in the MasterTool IEC Xtorm software and click on the Devices Tree (left bar). Double click *Device*, select the CPU (Communication Settings), and click *Files*. Click *Refresh*, both in the computer files column (left) and in the CPU files column (right) as shown in the figure below.
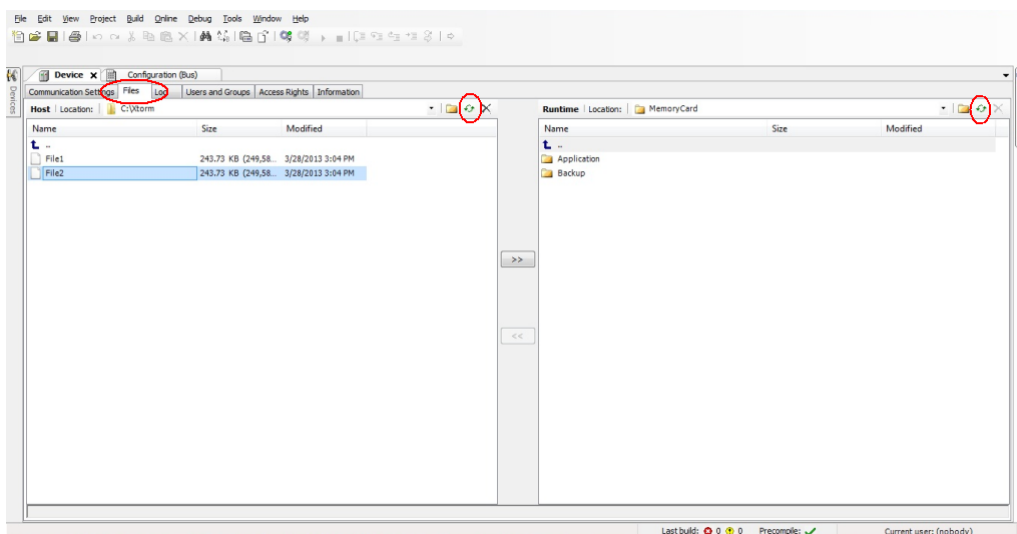

Figure 119: User Files Access

After updating the CPU files column, the root directory of files stored in the CPU is displayed and the folder to which files are transferred can be selected. The folder *InternalMemory* is a default folder to be used to store files in the internal memory of

altus

the CPU (32 Mbytes), since it is not possible to transfer files to the root directory. If necessary, other folders can be created in the root directory or subfolders within the *InternalMemory* folder. The *MemoryCard* folder is the directory where the memory card is mounted, if it is inserted in the CPU. Files transferred to the *MemoryCard* folder will be transferred directly into the memory card.

> **ATTENTION**
>
> In the case where the memory card is inserted after the CPU startup, an username and password will be requested to perform the MasterTool Xtorm access and/or file transfers to the memory card or vice versa. The standard user with privileges to access the CPU is *"Owner"* and the default password for that user is *"Owner"* as well.

In order to perform a file transfer from the microcomputer to the CPU just select the desired file in the left column and press "»" (screen center), as shown in the figure below. The download time varies depending on the file size and cycle time (execution) of the current CPU application, and may take several minutes.

The user does not need to be in Run Mode nor connected to the CPU to perform the transfers, since it has the ability to connect automatically in such processes.
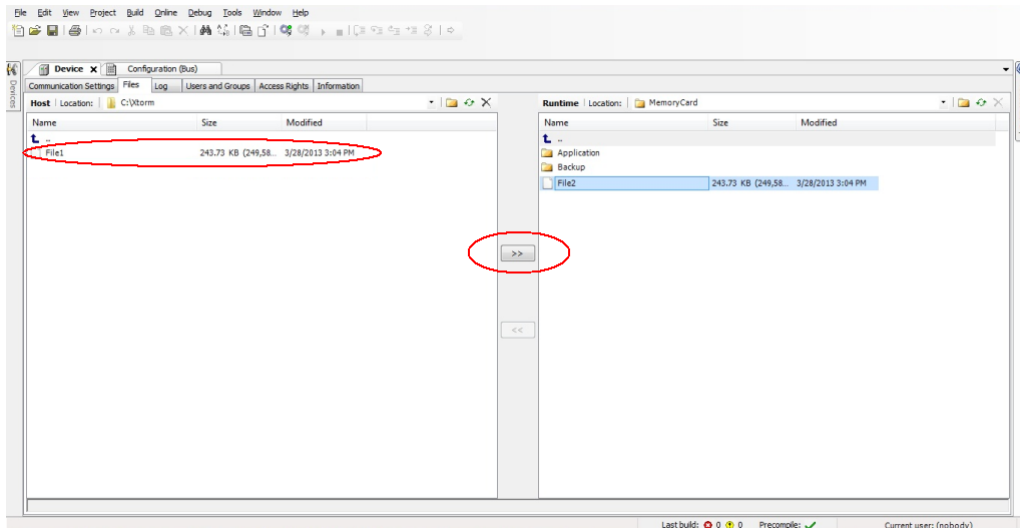

Figure 120: Transferring Files

> **ATTENTION**
>
> The files contained in the folder of a project created by MasterTool Xtorm tool have special names reserved by the system in this way cannot be transferred through the *Files* tab. If the user wishes to transfer a project to the user memory, you must compact the folder and then download the compressed file (* .zip for example).

In case it is necessary to transfer documents from the CPU to the PC in which the MasterTool Xtorm software is installed, the user must follow a very similar procedure to the previously described, as the file must be selected from the right column and the button "«" pressed, placed on the center of the screen.

Furthermore, the user has some operation options in the storing files area:

- **New directory** : Creates a new folder in the user memory area.

- **Delete item** : Removes the files from the folders in the user memory area.

- **Refresh** : Updates the files on the MasterTool Xtorm screen (user memory files and computer files).

altus

Figure 121: Utilization Options

> **ATTENTION**
>
> For a CPU in Stop Mode or no application, the transfer rate to internal memory is approximately 150 Kbytes/s.

4.10. Memory Card

Among other memories, the Hadron Xtorm Series CPU enables the user to use a memory card, defined according to the characteristics in section Memory Card Interface, which stores, among other files, the project and the application that is in the internal memory of the CPU.
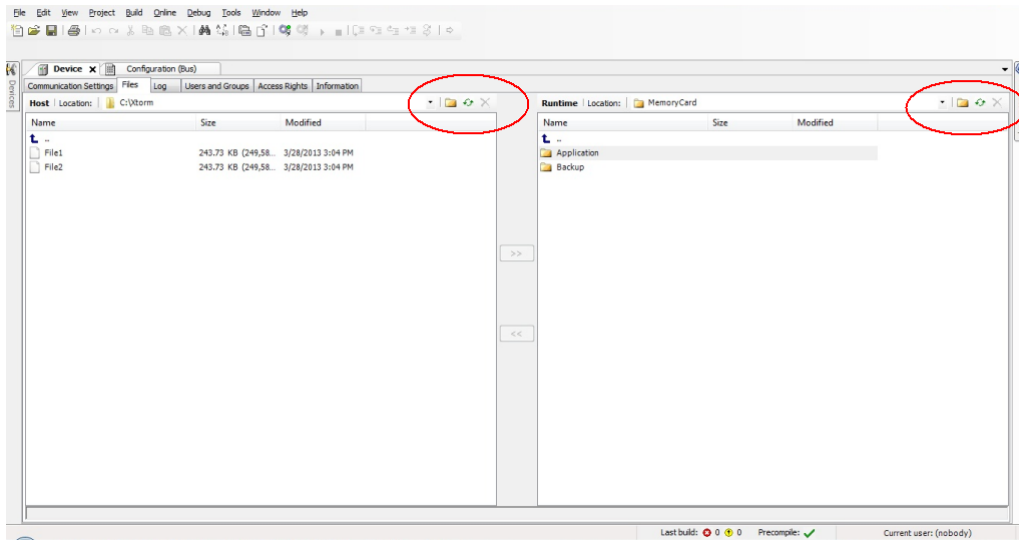
When the card is inserted in the CPU and it features a file system different than FAT32, it automatically identifies those files and prompts the user about formatting the files. In negative case the user cannot use the card, as it will not mount. The CPU displays a message informing a mismatch format, so the display card does not indicate the card. If the user decides to format the files, the CPU takes a few minutes to execute the operation, depending on the cycle time (execution) of the application which is running in the CPU. Once the memory card is mounted, the CPU will read general information from it, leaving the memory card access slower in the first few minutes. This is done only when the memory card is inserted or when the CPU is restarted.

> **ATTENTION**
>
> It is recommended to perform the memory card formatting procedure directly on the Hadron Xtorm CPU to avoid possible usage problems, increased mounting time or even incorrect operation.
> It is not recommended to remove the memory card or de-energize the CPU during formatting or during file transfer, as this can cause loss of data as well as irreversible damage to the card.

During project configuration, within the MasterTool Xtorm software, the user enables the option to copy the project from the CPU to the memory card or to copy the project on the card to the CPU. In this same screen the user can set access passwords, controlling the use of this information. For more information about the table, see the section Memory Card.
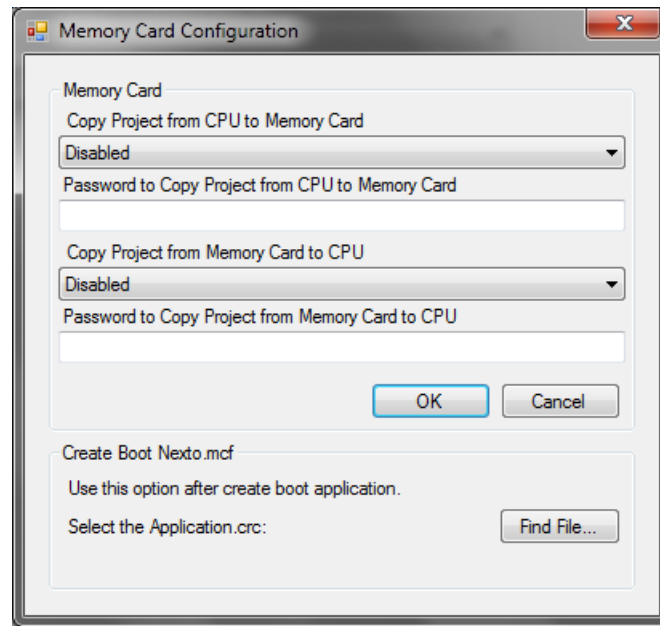
altus

Figure 122: Memory Card Settings

When a password is configured for the memory card in MasterTool, it is necessary to perform the following steps so that when the project is sent, the encrypted file which is created by MasterTool has the password included in its content and it is also sent.

First set the password(s) and click on the *"OK"* button. At this point the passwords have been registered and the next step is to execute in the *Communication* menu the *"Create Boot Application"* command, remembering that you cannot be logged into the CPU to perform this procedure. After executing this command, three files are created. One with extension *"app"* and another with extension *"crc"*, the folder where the project is saved will also create a file with extension *".ncf"* necessary to transfer the project from the card to the CPU. To complete the password setting operation, you must click again on the button *"Memory Card"*, which is in the configuration of the General Parameters of the CPU and then locate the file with extension *"crc"* generated in the previous step, using the button *" Find File..."*. Once these steps are done, MasterTool Xtorm will send all the files needed to perform the operations of sending and receiving projects via memory card.

In case the card is mounted, the password will be recorded on it, otherwise the set password in MasterTool will be requested if the user tries to transfer the project from the CPU to the card.

To perform the sending from the CPU to the memory card or vice versa, the user, in addition to enabling it in the MasterTool Xtorm software and entering the password, will have to access the Menu *"Memory Card"* on the CPU, using the diagnostics button, and select the desired transfer option. Then the password will be prompted, if the user has set it during application setup. Then, with a short press on the diagnostic button the digits are incremented and with a long press they are confirmed. At the sixth confirmed digit, the CPU will consist the password and start the process. When the passwords of both the application in the CPU and the application in the memory card are the same, no password entry is required in the CPU menu to transfer applications. For more information on using the diagnostics button, see the section One Touch Diag.

To remove the memory card, just access the *"Memory Card"* Menu on the CPU, using the diagnostics button, and select the memory card removal option, and wait until the card icon disappears from the status screen of the graphical display.

ATTENTION

If the memory card is removed without being disassembled via the CPU menu, while transferring or formatting files, this process may result in the loss of data on the card, as well as corrupting the files it contains. This process may imply the need for reformatting the card when inserted again in the CPU.

ATTENTION

If there is any file in the root of the memory card named *"Application"* or *"Backup"*, it will be deleted for the creation of the folders with the same name, used by the CPU to store the application project and the project archive. Folders with these names will not be overwritten.

4.10.1. Access in MasterTool

The access to the memory card is linked to the same memory screen in the MasterTool Xtorm software, and it is built on the *"MemoryCard"* folder. The *"Application"* and *"Backup"* directories are created inside the memory card every time it is inserted into the CPU. If they already exist, the system will recognize them and will not overwrite the folders.
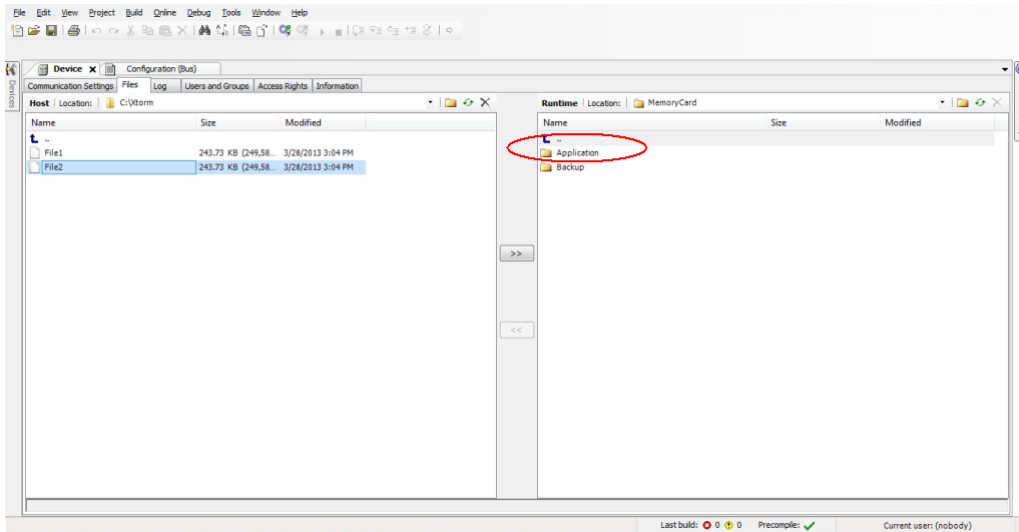
Figure 123: Root Directory with Memory Card

The file transfer is similar to the user memory use (User Files Memory). Access the *"MemoryCard"* directory and send the files, as shown in the figure below.
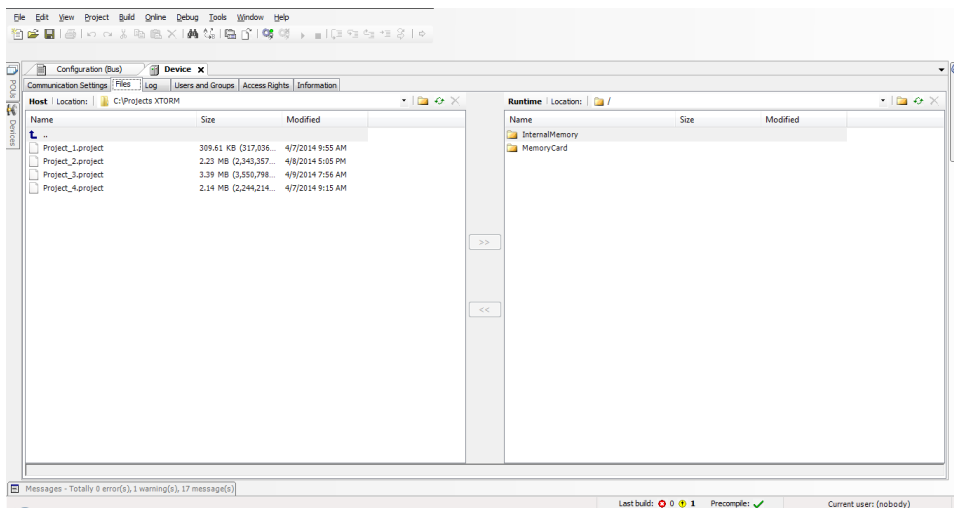
Figure 124: Files Saved in the Memory Card

In the memory card directory, in addition to the files that are stored on the card, will be the folders *"Application"* and *"Backup"*. In these folders the application and the current project will be saved if the user chooses to transfer or backup them via the PCU menu.

> **ATTENTION**
>
> The file transfer time depends on the difference of the interval time minus the average execution time of the running task(s) (time available until the next task cycle), i.e. the greater this difference is for each task in an application, the faster the transfer of a data from the memory card to the CPU/MasterTool Xtorm or vice versa.
>
> Transferring files to the memory card will be slower than transferring them to the internal memory of the CPU. For a CPU in Stop mode or with no application, the transfer rate approaches 100 Kbytes/s.

> **ATTENTION**
>
> MasterTool Xtorm has some reserved filenames that can not be transferred to the memory card: Application.app, Application.crc, Archive.prj, Stdlogger.csv

4.11. Informative Menu and of CPU's Configuration

Access to the Informative and Configuration Menu of the Hadron Xtorm CPU, as well as detailed access to diagnostics, are available through levels, where to access the menu information, change levels and modify some setting, a long press on the diagnostics button is enough, and to navigate through the items of the same level, a short press on the diagnostics button is enough. See the section One Touch Diag to verify the operation and the difference between the types of diagnostic button presses.

The table below shows the menu levels and the type of each screen available on the CPU, that is, whether it is informational, configurable or returns a level.

Level 1	Level 2	Level 3	Type
HARDWARE	TEMPERATURE	-	Informative
	CONTRAST	CONTRAST LEVEL	Configurable
	DATE AND TIME	-	Informative
	BACK	-	Back Level
LANGUAGE	ENGLISH	>ENGLISH	Configurable
	PORTUGUES	>PORTUGUES	Configurable
	ESPANOL	>ESPANOL	Configurable
	BACK	-	Back Level
NETWORK	NET 1 IP ADDRESS	-	Informative
	NET 1 MASK	-	Informative
	NET 2 IP ADDRESS	-	Informative
	NET 2 MASK	-	Informative
	NET 3 IP ADDRESS	-	Informative
	NET 3 MASK	-	Informative
	NET 4 IP ADDRESS	-	Informative
	NET 4 MASK	-	Informative
	NET 5 IP ADDRESS	-	Informative
	NET 5 MASK	-	Informative
	NET 6 IP ADDRESS	-	Informative
	NET 6 MASK	-	Informative
	BACK	-	Back Level
SOFTWARE	FIRMWARE	-	Informative
	BOOTLOADER	-	Informative
	PROC. AUX.	-	Informative
	BACK	-	Back Level
REDUNDANCY	SWITCH TO ACTIVE?	-	Operacional
	SWITCH TO STANDBY?	-	Operacional
	BACK	-	Back Level

altus

Level 1	Level 2	Level 3	Type
MEMORY CARD	UNMOUNT	-	Operacional
	CPU > MEMORY CARD	CPU PASSWORD	Operacional
	BACKUP	-	Operacional
	FORMAT	CONFIRM?	Operacional
	MEMORY CARD > UCP	CARD PASSWORD	Operacional
	BACK	-	Back Level
BACK	-	-	Back Level

Table 194: CPU's Menu Levels

Note:

Memory Card: The memory card will only be available in the menu if it is inserted in Hadron Xtorm CPU.

Password: The access password to memory card's data will be only needed in case it was previously configured in MasterTool Xtorm software. It's not possible to edit the password through the menu. The password needs to have at least 6 characters.

Redundancy: "Redundancy" menu will only be available in the menu if the CPU is identified as redundant.

As shown by the table above, among the options available for visualization and modification are the main data needed by the user, such as:

- Hardware information resources:
  - TEMPERATURE – CPU Internal Temperature(E.g..: 36 C 97 F)
  - CONTRAST – Contrast CPU visor adjust
  - DATE AND TIME – CPU Date and Time configured (E.g..: 2001.01.31 00:00)

- CPU's Menu language switch:
  - PORTUGUES – Switch to Portuguese idiom
  - ENGLISH – Switch to English idiom
  - ESPANOL – Switch to Spanish idiom

- Device's Network information resources:
  - NET 1 IP ADDRESS – IP Address (E.g..: 192.168.0.1)
  - NET 1 MASK – Subnetwork Mask (E.g..: 255.255.255.0)
  - NET 2 IP ADDRESS – IP Address (E.g..: 192.168.1.1)
  - NET 2 MASK – Subnetwork Mask (E.g..: 255.255.255.0)
  - NET 3: DISABLED. (If it is not in use in user's application)
  - NET 4: DISABLED. (If it is not in use in user's application)
  - NET 5: DISABLED. (If it is not in use in user's application)
  - NET 6: DISABLED. (If it is not in use in user's application)

- Software information resources:
  - FIRMWARE – CPU's software version (E.g..: 1.0.0.0)
  - BOOTLOADER – CPU's Bootloader version (E.g..: 1.0.0.0)
  - PROC. AUX. – CPU's auxiliary processor version (E.g..: 1.0.0.0)

- CPU's redundancy menu access:
  - SWITCH TO ACTIVE ?
  - SWITCH TO STANDBY ?

- Memory Card's data access:
  - MEMORY CARD>CPU – Project transference from memory card to CPU
  - CPU>MEMORY CARD – Project transference from CPU to memory card
  - FORMAT – Formats the memory card to FAT32 system

- UNMOUNT – Enables the secure removal of memory card
- BACKUP – Does a CPU's data backup on memory card

The figure below describes an example of how to operate the Hadron Xtorm CPU menu, through the contrast adjustment procedure in the Status screen. Besides facilitating the configuration, it is possible to identify all screen levels and the type of pressing to navigate between them, and to change the other parameters, such as Language and insert the password(s) in the Memory Card, it is only necessary to follow the same access logic. The short press shows that the contrast is being increased (lighter), and on the next press after its maximum value, it returns to the minimum value (darker). The long press shows confirmation of the desired contrast and a return to the previous level.
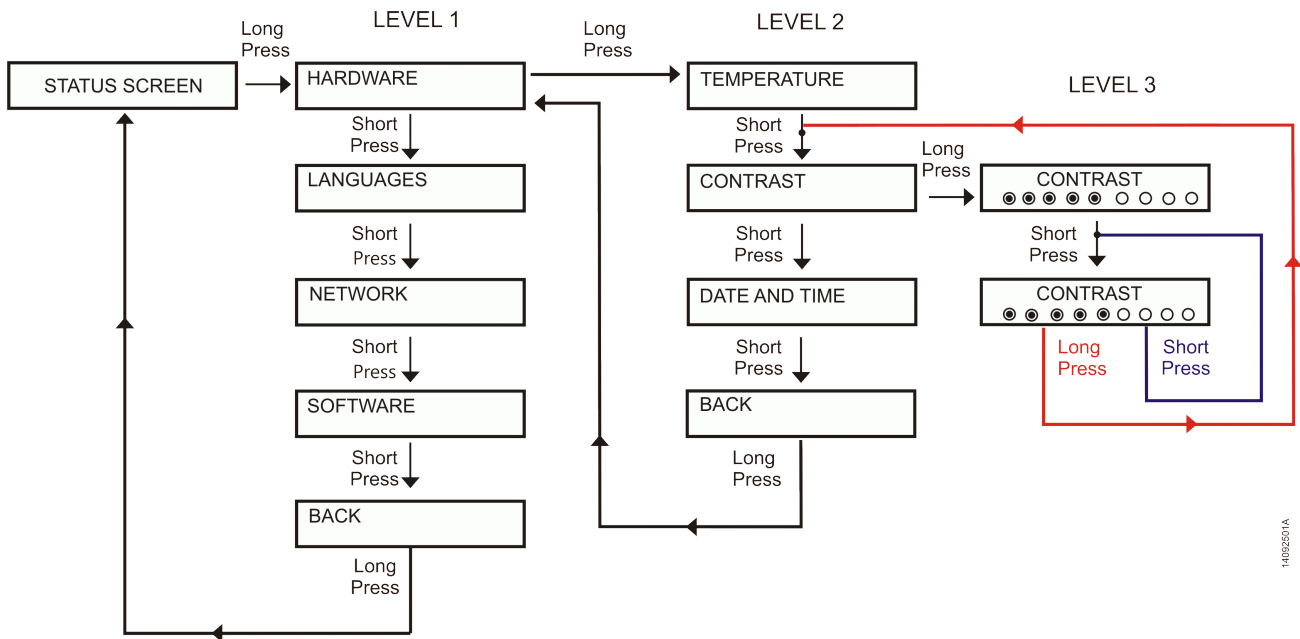

Figure 125: Contrast Adjustment

In addition to the Hadron Xtorm CPU menu being exited by a long press on the diagnostics button on the *BACK* screen of level 1, there are also other exit conditions, which are described below:

- Short pressing at any time on the other modules present in the bus causes the CPU to exit the menu and show the diagnostics of the desired module.
- Inactive timeout, any level, over 5 s.

4.12. Function Blocks and Functions

4.12.1. Inputs and Outputs Update

By default, the local bus and CPU integrated I/O are updated on every execution cycle of MainTask. The Refresh functions allows to update these I/O points asynchronously at any point of user application code.

When the function blocks to update the inputs and outputs are not used, the update is performed every cycle of the Main-Task.

> **ATTENTION**
>
> At the startup of a CPU of this series, the inputs and outputs are only updated for reading and prepared for writing when the MainTask is performed.
> All other system tasks that run before MainTask will be with the inputs and outputs invalid.

4.12.1.1. REFRESH_INPUT

This function block is used to update the specified module inputs without the necessity to wait for the cycle to be completed. It is important to notice that the filters configured in the MasterTool IEC XE and the update time of the module inputs will have to be considered in effective time of the inputs update in the application developed by the user.

> **ATTENTION**
>
> The *REFRESH_INPUT* function must only be used in MainTask.
> To update inputs in other tasks, the option *Enable I/O update per task* must be selected, for further information about this option, consult Table 34.

> **ATTENTION**
>
> *REFRESH_INPUT* function does not support inputs that have been mapped to symbolic variables. For proper operation it is necessary that the input is mapped to a variable within the memory direct representation of input variables (%I).

> **ATTENTION**
>
> The *REFRESH_INPUT* function updates only the direct variables %I that are declared in the "Bus: I/O Mapping" tab of the module addressed in the respective rack/slot of the function. In the case of communication modules/interfaces (MODBUS, Profibus, etc.), the update does not include the direct variables of the device mappings.
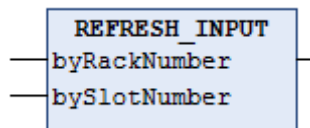
```
REFRESH_INPUT
byRackNumber
bySlotNumber
```

Figure 126: Block for Input Updating

Input parameters	Type	Description
byRackNumber	BYTE	Rack number.
bySlotNumber	BYTE	Position number where the module is connected.

Table 195: REFRESH_INPUT Input Parameters

Possible *TYPE_RESULT*:

- **OK_SUCCESS:** Execution success.
- **ERROR_FAILED:** This error is returned if the function is called for a module that has only outputs, or also if the option *Always update variables* (located in the module's configuration screen, tab *I/O Mapping*) is not checked.
- **ERROR_NOTSUPPORTED:** The called routine is not supported by the product.
- **ERROR_PARAMETER:** Invalid / unsupported parameter.
- **ERROR_MODULE_ABSENT:** The module is absent in the bus.
- **ERROR_MODULE_NOTCONFIGURED:** The module is not configured in the application.
- **ERROR_MODULE_NOTRUNNING:** The module is not running (is not in operational state).
- **ERROR_MODULE_COMMFAIL:** Failure in the communication with the module.
- **ERROR_MODULE_NOTFOUND:** The module was not found in the application or is not supported.

Utilization example in ST language:

altus

```
PROGRAM UserPrg
VAR
Info: TYPE_RESULT;
byRackNumber: BYTE;
bySlotNumber: BYTE;
END_VAR
//INPUTS:
byRackNumber := 0;
bySlotNumber := 10;
//FUNCTION:
Info := REFRESH_INPUT (byRackNumber, bySlotNumber); //Function call.
//Variable "Info" receives possible function errors.
```

4.12.1.2.   REFRESH_OUTPUT

This function block is used to update the specified module outputs. It is not necessary to wait until the cycle is finished. It is important to notice that the update time of the module outputs will have to be considered in the effective time of the outputs update in the application developed by the user.

> **ATTENTION**
>
> The *REFRESH_OUTPUT* function must only be used in MainTask.
> To update outputs in other tasks, the option *Enable I/O update per task* must be selected, for further information about this option, consult Table 34.

> **ATTENTION**
>
> *REFRESH_OUTPUT* function does not support inputs that have been mapped to symbolic variables. For proper operation it is necessary that the input is mapped to a variable within the memory direct representation of input variables (%Q).

> **ATTENTION**
>
> The *REFRESH_OUTPUT* function updates only the direct variables %Q that are declared in the "Bus: I/O Mapping" tab of the module addressed in the respective rack/slot of the function. In the case of communication modules/interfaces (MODBUS, Profibus, etc.), the update does not include the direct variables of the device mappings.
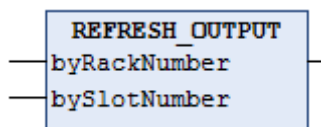
```
REFRESH_OUTPUT
byRackNumber
bySlotNumber
```

Figure 127: Block for Output Updating

altus

Input parameters	Type	Description
byRackNumber	BYTE	Rack number.
bySlotNumber	BYTE	Position number where the module is connected.

Table 196: REFRESH_OUTPUT Input Parameters

Possible *TYPE_RESULT*:

- **OK_SUCCESS:** Execution success.
- **ERROR_FAILED:** This error is returned if the function is called for a module that has only inputs, or also if the option *Always update variables* (located in the module's configuration screen, tab *I/O Mapping*) is not checked.
- **ERROR_NOTSUPPORTED:** The called routine is not supported by the product.
- **ERROR_PARAMETER:** Invalid / unsupported parameter.
- **ERROR_MODULE_ABSENT:** The module is absent in the bus.
- **ERROR_MODULE_NOTCONFIGURED:** The module is not configured in the application.
- **ERROR_MODULE_NOTRUNNING:** The module is not running (is not in operational state).
- **ERROR_MODULE_COMMFAIL:** Failure in the communication with the module.
- **ERROR_MODULE_NOTFOUND:** The module was not found in the application or is not supported.

Utilization example in ST language:

```
PROGRAM UserPrg
VAR
Info: TYPE_RESULT;
byRackNumber: BYTE;
bySlotNumber: BYTE;
END_VAR
//INPUTS:
byRackNumber := 0;
bySlotNumber := 10;
//FUNCTION:
//Function call.
Info := REFRESH_OUTPUT (byRackNumber, bySlotNumber);
//Variable "Info" receives possible function errors.
```

4.12.2. Retain Timer

The timer retain is a function block developed for applications as production line clocks, that need to store its value and restart the counting from the same point in case of power supply failure. The values stored by the function block, are only zero in case of a *Reset Cold*, *Reset Origin* or a new application *Download* (see the MasterTool IEC XE User Manual - MU299609), when the counters keep working, even when the application is stopped (Stop Mode).

> **ATTENTION**
>
> It is important to stress that, for the correct functioning of the Timer Retain blocks, the variables must be declared as Retain (*VAR RETAIN*). It's also important to notice that in simulation mode, the Timer Retain function blocks do not run properly due to need the Nexto CPU for correct behavior.

The three blocks already available in the MasterTool IEC XE software *NextoStandard* library are described below (for the library insertion proceeding, see MasterTool IEC XE Programming Manual – MP399609, section Library).

altus

4.12.2.1. TOF_RET

The function block *TOF_RET* implements a time delay to disable an output. When the input *IN* has its state changed from (TRUE) to (FALSE), or a falling edge, the specified time *PT* will be counted and the *Q* output will be driven to (FALSE) at the end of it. When the input *IN* is in logic level 1 (TRUE), the output *Q* remain in the same state (TRUE), even if this happened in the middle of the counting process. The *PT* time can be changed during the counting as the block assumes the new value if the counting hasn't finished. Figure 128 depicts the *TOF_RET* block and Figure 129 shows its graphic behavior.

Figure 128: TOF_RET Block

Input parameters	Type	Description
IN	BOOL	This variable, when receives a falling edge, enables the block counting.
PT	TIME	This variable specifies the block counting limit (time delay).

Table 197: TOF_RET Input Parameters

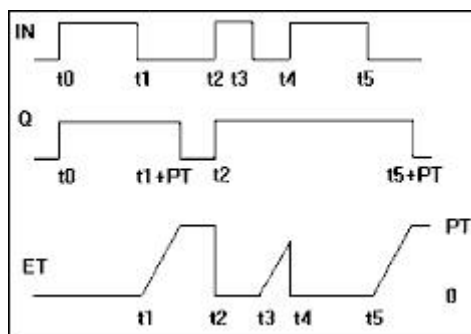| Output parameters	Type	Description
Q	BOOL	This variable executes a falling edge as the PT variable (time delay) reaches its maximum value.
ET	TIME	This variable shows the current time delay.

Table 198: TOF_RET Output Parameters

Figure 129: TOF_RET Block Graphic Behavior

altus

Utilization example in ST language:

```
PROGRAM UserPrg
VAR RETAIN
bStart : BOOL := TRUE;
TOF_RET : TOF_RET;
END_VAR

// When bStart=FALSE starts counting
TOF_RET( IN := bStart,
PT := T#20S);

// Actions executed at the end of the counting
IF (TOF_RET.Q = FALSE) THEN
bStart := TRUE;
END_IF
```

4.12.2.2. TON_RET

The *TON_RET* implements a time delay to enable an output. When the input *IN* has its state changed from (FALSE) to (TRUE), or a rising edge, the specified time *PT* will be counted and the *Q* output will be driven to (TRUE) at the end of it. When the input *IN* is in logic level 0 (FALSE), the output *Q* remain in the same state (FALSE), even if it happens in the middle of the counting process. The *PT* time can be changed during the counting as the block assumes the new value if the counting hasn't finished. Figure 130 depicts the *TON_RET* block and Figure 131 shows its graphic behavior.

Figure 130: TON_RET Function Block

Input parameters	Type	Description
IN	BOOL	This variable, when receives a rising edge, enables the function block counting.
PT	TIME	This variable specifies the block counting limit (time delay).

Table 199: TON_RET Input Parameters

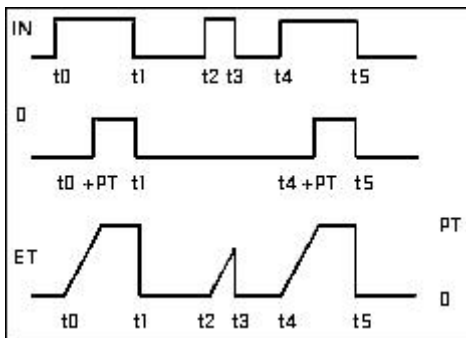| Output parameters	Type	Description
Q	BOOL	This variable executes a rising edge as the PT variable (time delay) reaches its maximum value.
ET	TIME	This variable shows the current time delay.

Table 200: TON_RET Output Parameters

altus

Figure 131: TON_RET Block Graphic Behavior

Utilization example in ST language:

```
PROGRAM UserPrg
VAR RETAIN
bStart : BOOL;
TON_RET : TON_RET;
END_VAR

// Quando bStart=TRUE starts counting
TON_RET( IN := bStart,
PT := T#20S);

// Actions executed at the end of the counting
IF (TON_RET.Q = TRUE) THEN
bStart := FALSE;
END_IF
```

4.12.2.3. TP_RET

The *TP_RET* function block works as a trigger. The timer which starts when the *IN* input has its state changed from (FALSE) to (TRUE), that is, a rising edge, it is increased until the *PT* time limit is reached. During the counting, the *Q* output is (TRUE), otherwise it is (FALSE). The *PT* time can be changed during the counting as the block assumes the new value if the counting has not finished. Figure 132 depicts the *TP_RET* and Figure 133 shows its graphic behavior.

Figure 132: TP_RET Function Block

Input parameters	Type	Description
IN	BOOL	This variable, when receives a rising edge, enables the function block counting.
PT	TIME	This variable specifies the function block counting limit (time delay).

Table 201: TP_RET Input Parameters

altus

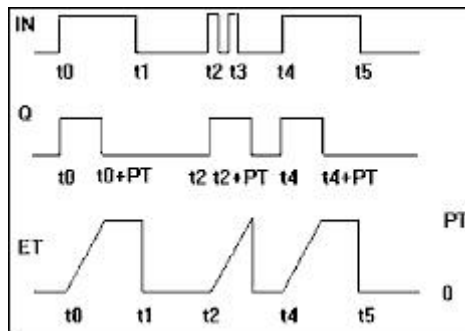| Output parameters	Type	Description
Q	BOOL	This variable is true during the counting, otherwise is false.
ET	TIME	This variable shows the current time delay.

Table 202: TP_RET Output Parameters

Figure 133: TP_RET Block Graphic Behavior

Utilization example in ST language:

```
PROGRAM UserPrg
VAR RETAIN
bStart : BOOL;
TP_RET : TP_RET;
END_VAR

// Configure TP_RET
TP_RET( IN := bStart,
PT := T#20S);

bStart := FALSE;

// Actions executed during the counting
IF (TP_RET.Q = TRUE) THEN
// Executes while the counter is activated
ELSE
// Executes when the counter is deactivated
END_IF
```

4.12.3. Non-Redundant Timer

The non-redundant timer is used in applications for the redundant NX3030 CPU which need a timer in the non-redundant program of a half-cluster. This timer does not use the IEC timer, therefore, it will not be synchronized in case the reserve half-cluster assumes the active status and the active one goes for reserve.

The three types of blocks already available in the *NextoStandard* library of the MasterTool IEC XE software are describe as follows (for doing the procedure of library's inclusion, check MasterTool IEC XE Programming Manual – MP399608, section Library).

altus

4.12.3.1. TOF_NR

The *TOF_NR* function block implements a delay time for disabling an output and has its functioning and configuration similar to the TOF_RET function block, differentiating itself only for not being redundant nor retentive.

Figure 134: TOF_NR Function Block

Utilization example in ST language:

```
PROGRAM NonSkippedPrg
VAR
bStart : BOOL := TRUE;
TOF_NR : TOF_NR;
END_VAR

// When bStart=FALSE starts the counting
TOF_NR( IN := bStart,
PT := T#20S);

// Actions executed at the end of the counting
IF (TOF_NR.Q = FALSE) THEN
bStart := TRUE;
END_IF
```

4.12.3.2. TON_NR

The *TON_NR* function block implements a delay time to enable an output and has its functioning and configuration similar to the TON_RET function block, differentiating only for not being redundant nor retentive.

Figure 135: TON_NR Function Block

Utilization example in ST language:

```
PROGRAM NonSkippedPrg
VAR
bStart : BOOL;
TON_NR : TON_NR;
END_VAR

// When bStart=TRUE starts the counting
TON_NR( IN := bStart,
PT := T#20S);
```

altus

```
// Actions executed at the end of the counting
IF (TON_NR.Q = TRUE) THEN
bStart := FALSE;
END_IF
```

4.12.3.3. TP_NR

The *TP_NR* function block works as a trigger and has its functioning and configuration similar to the TP_RET function block, differentiating only for not being redundant nor retentive.
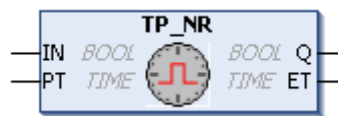

Figure 136: TP_NR Function Block

Utilization example in ST language:

```
PROGRAM NonSkippedPrg
VAR
bStart : BOOL;
TP_NR : TP_NR;
END_VAR

// Configure TP_NR
TP_NR( IN := bStart,
PT := T#20S);

bStart := FALSE;

// Actions executed during the counting
IF (TP_NR.Q = TRUE) THEN
// Executes while the counter is activated
ELSE
// Executes when the counter is deactivated
END_IF
```

4.12.4. User Log

Feature that allows the user to create own records and write to log files on the memory card present in the CPU. The files are generated in a specific directory of the memory card in the CSV format, allowing viewing in text editors and spreadsheets. The separator was the semicolon character. For more information about the use of the memory card, see section Memory Card.

There are two functions available, one for log information and another to remove all records. The following is a description of the types of data used by the functions:

Data type	Option	Description
USER_LOG_EVENT_TYPES	USER_LOG_EVENT_ERROR	The user is free to use the best indication according to log message severity.
	USER_LOG_EVENT_DEBUG	
	USER_LOG_EVENT_INFO	
	USER_LOG_EVENT_WARN	
USER_LOG_MESSAGE		Log message with 150-character max.
USER_LOG_ERROR_CODES	USER_LOG_OK	The operation was performed successfully.
	USER_LOG_FAILED	The operation was not performed successfully. The reason for the failure can be checked in the system logs – see section System Log.
	USER_LOG_BUFFER_FULL	Messages are being added beyond the processing capacity.
	USER_LOG_NO_MEMORY	At the time, there were no resources to perform the operation.
	USER_LOG_FILE_SYSTEM_ERROR	There was an error while accessing the memory card or there is no available space. Error information can be verified in the logs of system – see section System Log.
	USER_LOG_NO_MEMORY_CARD	There is a memory card present in the CPU.
	USER_LOG_MEMORY_CARD_FULL	There is no free space available on the memory card.
	USER_LOG_PROCESSING	The resource is busy executing the last operation, for example, deleting all log files.

Table 203: Data Type for User Log

The following two functions are available in the LibLogs library in MasterTool Xtorm 1.10. To perform the procedure of inserting a library, see the section Libraries.

> **ATTENTION**
>
> User Logs are available only from version 1.1.0.12 of the Hadron Xtorm Series CPU. Also, version 1.10 or higher of MasterTool Xtorm is required to use this feature.

4.12.4.1. UserLogAdd

This function is used to add a new user log message, adding in a new line to the log file on the memory card. The message must have a maximum length of 150 characters, and the event type of the message. Application variables can be registered using conversion to string and concatenation with the main message. The date and time information in UTC (timestamp) is automatically added in the message with a resolution of milliseconds where the event was registered. The date and time information is also used in the formation of the names of the log files.

The *UserLogAdd* function can be used to enter multiple messages within a single task and also in different application tasks. However independent of each execution of the function in the application, being on the same task or on different tasks, all use the same feature to record the desired messages. For this reason it is recommended that the addition of messages using the *UserLogAdd* function in the application be held every 50 ms to prevent the return of buffer overload. If the function is performed in periods shorter than the indicated, but respect the average time of 50 ms between each message addition at the end of the interval for the task, also prevents the return of buffer overload. So that the logs are added correctly, it is important to respect time limits when the card is inserted and at startup of the CPU, mentioned in section Memory Card. After the operation the function returns the options for the given type *USER_LOG_ERROR_CODES* as Table 203. When the function return is not *USER_LOG_OK*, the message was not registered and the function *UserLogAdd* should be re-executed with the

desired message. In case of return of consecutive writing failures, the memory card can be damaged. The replacement by a healthy memory card ensures that the latest logged messages will be recorded on the card that is not damaged, since the CPU is not restarted.

The figure below represents the function *UserLogAdd* and table below the input parameters:
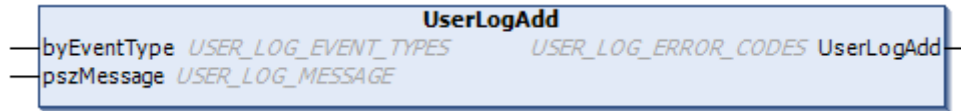

Figure 137: UserLogAdd Function

Input Parameters	Type	Description
byEventType	BYTE	This variable specifies the event type of the log being added as options for the USER_LOG_EVENT_TYPES data type.
pszMessage	USER_LOG_MESSAGE	This variable should contain the set of characters that compose the message to be added to the log file. The message must contain a maximum of 150 characters.

Table 204: UserLogAdd Input Parameters

The log files are generated and organized on the memory card in a specific directory path depending on the CPU serial number and the firmware version installed. For example, for a CPU with serial number 445627 and firmware version 1.4.0.4, the location where the log files should be written to the memory card is *MemoryCard/UserLog/445627/1.4.0.4/*.

The names of the log files are formed by the date and time (timestamp) of the first message. Except when there's a problem to use this name, for example, another existing file with the same name, in this situation it is used the instant date and time. The file name follows the following pattern: *year/month/day/hour/minute/second/millisecond.CSV*. In case of file access problem due to defective sector not enabling to continue writing, will be added to the name of this file the extension *"corrupted"* and a new file will be created. The amount of logs per file is not fixed, varying depending on the size of messages. The amount of created files is limited to 1024 with maximum size of 1 MB each, so the memory card requires 1 GB of free space. When it reaches the limit of 1024 files created on the memory card, during CPU operation, the oldest files are removed so that files with latest logs are preserved, even in cases of partial manual removal of the files in the directory where the files are being written.

The viewing of the log files can be performed through worksheets or conventional text editors. The concatenated information, for improved visualization, may use semicolons between the strings of the message to separate them. One must be careful in formatting cells with floating point values.

Utilization example in ST language:

```
PROGRAM UserPrg
VAR
  eLogError : USER_LOG_ERROR_CODES;
  sMessage :USER_LOG_MESSAGE;
END_VAR

IF (m_rTemperature > MAX_TEMPERATURE_ACCEPT) THEN
  sMessage := 'Temperature higher than expected: ';
  sMessage := concat(sMessage, REAL_TO_STRING(m_rTemperature));
  sMessage := concat(sMessage, '°');
  eLogError := UserLogAdd(USER_LOG_EVENT_WARN, sMessage);
//eLogError variable gets possible function errors.
END_IF
```

Log file content example: (*UserLog-201308271506245738.csv*)

```
Model; NX3008
Serial number; 445627
Firmware version; 1.4.0.4

27/08/2013 15:06:24.5738; WARN; Overtemperature: 25°
27/08/2013 16:37:45.3476; WARN; Overtemperature: 25°
28/08/2013 09:10:55.4201; WARN; Overtemperature: 26°
```

4.12.4.2.   UserLogDeleteAll

The *UserLogDeleteAll* function performs the deletion of log files present in the directory created specifically for the CPU in which is inserted the memory card, i.e. are only deleted the logs contained in the directory named with the CPU firmware version that exists within the directory with the CPU serial version. The log files deleted are only files that exist at the time of memory card mounting and the generated by the *UserLogAdd* function. Logs of other CPUs and files added manually by the user during execution are not deleted. The figure below represents the function *UserLogDeleteAll*.
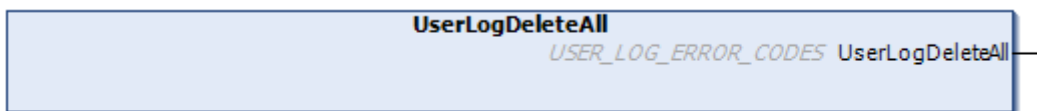

Figure 138: UserLogDeleteAll Function

Utilization example in ST language:

```
PROGRAM UserPrg
VAR
eLogError : USER_LOG_ERROR_CODES;
END_VAR

IF (m_DeleteLogs = TRUE) THEN
eLogError := UserLogDeleteAll();
m_DeleteLogs := FALSE;
//eLogError variable gets possibles function errors.
END_IF
```

> **ATTENTION**
>
> The *UserLogDeleteAll* function's return does not indicate operation completed, just confirmation of execution that can take a large amount of time if there are hundreds of log files in the directory. The function to record the new user log is unavailable right now, returning the *USER_LOG_PROCESSING* option for any operation. The result of the operation can also be checked in the system log.

4.12.5.   ClearRtuDiagnostic

The *ClearRtuDiagnostic* functional block provided by the *LibRtuStandard* library can be used when it is necessary to clear the diagnostics that indicate Loss of Retentivity, Event Queue Overflow and can also reset the flag of clean event queue. As described in the following table, the action that the function will perform is defined by the ENUM code assigned in the *eDiagnostic* input.

altus

Name	ENUM (BYTE)	Description
DIAG_RETENTIVITY_LOST	0	Clears Retentivity Loss diagnostics
DIAG_QUEUE_OVERFLOW	1	Clears Event Queue Overflow diagnostics.
DIAG_QUEUE_CLEARED	2	Clears Clean Event Queue diagnostics.

Table 205: Description of actions of ClearRtuDiagnostic

The following table describes the execution status of the function that will be shown in its output variable.

Name	ENUM (UDINT)	Description
OK_SUCCESS	0	Sucess.
ERROR_PARAMETER	3	Parameter error (the value assigned on eDiagnostic input is out of the valid range).

Table 206: Results of ClearRtuDiagnostic

Example in ST language, where the function call will clear the *DG_HX3040.tSummarized. bRetentivityLost* bit of the CPU diagnostics structure:

```
PROGRAM UserPrg
VAR
Result : TYPE_RESULT;
END_VAR

Result := ClearRtuDiagnostic(eDiagnostic := 0);
```

4.12.6. ClearEventQueue

The *ClearEventQueue* function available by the *LibRtuStandard* library can be used when it's needed to clear the CPU's event queue and of all instanced drivers.

According to table below the function's execution result is going to be showed in its output variable.

Name	ENUM (UDINT)	Result Description
OK_SUCCESS	0	Success
ERROR_FAILED	1	General error
ERROR_NOTSUPPORTED	2	The called routine is not supported by the product
ERROR_PARAMETER	3	Invalid/unsupported parameter
ERROR_MODULE_ABSENT	16	The module is absent in the bus
ERROR_MODULE_NOTCONFIGURED	17	The module is not configured in the application
ERROR_MODULE_NOTRUNNING	18	The module is not running (isn't in operational state)
ERROR_MODULE_COMMFAIL	19	Failure in the communication with the module
ERROR_MODULE_NOTFOUND	20	The module wasn't found in application or is not supported

Table 207: ClearEventQueue Function Results

altus

Using example in ST language, where the function call is going to clear the events queue, and consequently, reset the communication drivers events queue usage diagnostics *T_DIAG_DNP_SERVER_1.tClient_*.tQueueDiags.wUsage:*

```
PROGRAM UserPrg
VAR
   ClearEventQueueStatus : TYPE_RESULT;
END_VAR

ClearEventQueueStatus := ClearEventQueue();
```

4.13. User Management and Access Rights

Provide functions to define user accounts and configure access rights to the project. Note that the specific user management device must be supported to control the access rights in the CP file system and objects at runtime.

The rights to access the design objects via specified actions are assigned only to groups and, therefore, each user must belong to a group.

4.13.1.   User Management and Project's Access Rights

4.13.1.1.   User Management

The configuration of users and groups is made in the *Project* dialog in the *Project Settings* window.

The projects include, automatically, with a default group, called *Everyone*. All users of other groups are members of this particular group. Thus, each user account is automatically provided with at least the default settings. The Everyone group can not be deleted (only renamed) and its members can not be removed.

The project also features called Owner group that contains the user *Owner*. Users can be added or removed from this group, but at least one user must remain. This group can not be deleted and always has all access rights. Both the Owner group, as the Owner user can be renamed.

When starting the programmer and a project, there is no user logged on. However, the user logon can be accomplished through a set account with name and password, so he can get specific access rights.

Note that each project has its own user management, for example, to have specific access rights in a library, the user must logon in this library separately. Users and groups defined in different projects are not identical, even if they have the same name.

ATTENTION

- User passwords are stored irreversibly. If a password is lost, the corresponding user account can no longer be used. If the Owner password is lost, the entire project may be unusable.
- By default, in new projects, the password Owner user is empty.
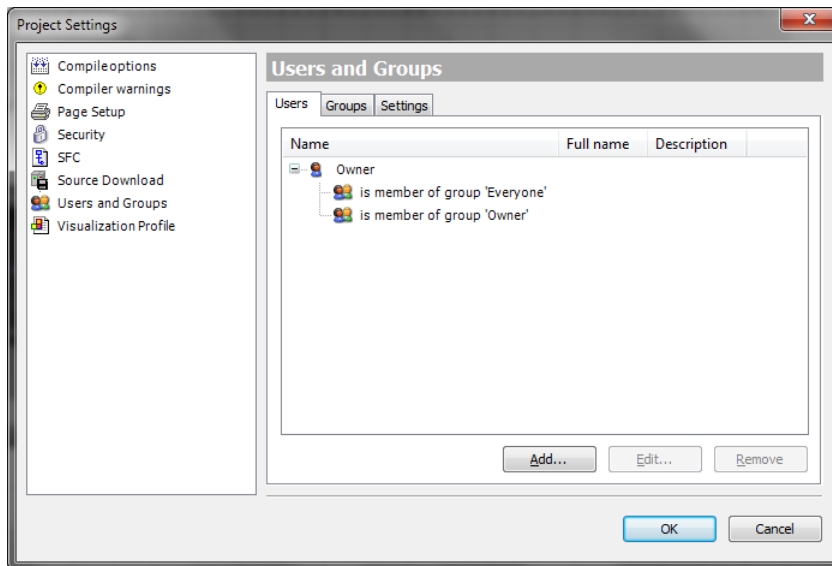

4.13.1.1.1. Users

Figure 139: Project Settings - Users

Registered current users are listed in a tree structure. Through the *Add* or *Edit* commands you can display beside Name (logon), the full name and the user description. Each user's properties can be viewed or not (are hidden) through the plus and minus sign respectively. Each user, by default, is member of the *"Everyone"*.

To define a new user account, use the *Add* button to open the *Add User* dialog.
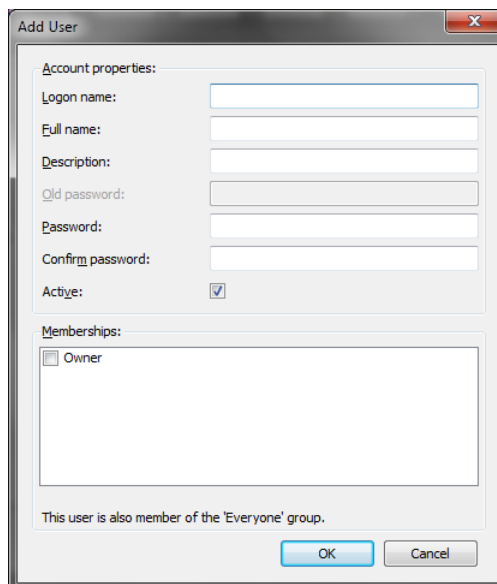

Figure 140: Adding an User

Account properties owns the following fields:

- *Logon name:* New users logon name..
- *Full name:* New user's full name. Just useful as additional information.
- *Description:* New user's description. Just useful as additional information.

- *Old password:* This field is only editable when the dialog is used to modify an existent user account. Before change the password, however it's necessary type the updated password.
- *Password:* New user's password. Digits will appear as (*).
- *Confirm password:* the password previously entered must be repeated and if the two passwords do not match, an error message will be displayed. This typing is also represented by asterisks (*).
- *Active:* This option active the user account, making it valid. When the account is not valid, the user can not perform the logon. The account is automatically disabled when they made repeated logon attempts with the wrong password.
- *Memberships:* This list lists all existing groups, and the group "Everyone", it belongs to the new user automatically. Selecting the respective items is defined to which the new user groups must belong.

To set the new user, close the dialog with OK. In the event of any inconsistency (incorrect password, login name of absence, existing user), an error message will appear.

To modify an existing user account, use the *Edit* button and open the corresponding dialog. These fields are the same as the *Add User* dialog. The password fields - for security reasons - will have 32 asterisks (*). After changing the desired items, close the dialog with OK to apply the new settings.

To delete one or more user accounts, select their users in the appropriate list and press *Remove*. Note that this action does not require confirmation. You can not delete all group members (at least one must remain). If you try this, an error message will be shown.
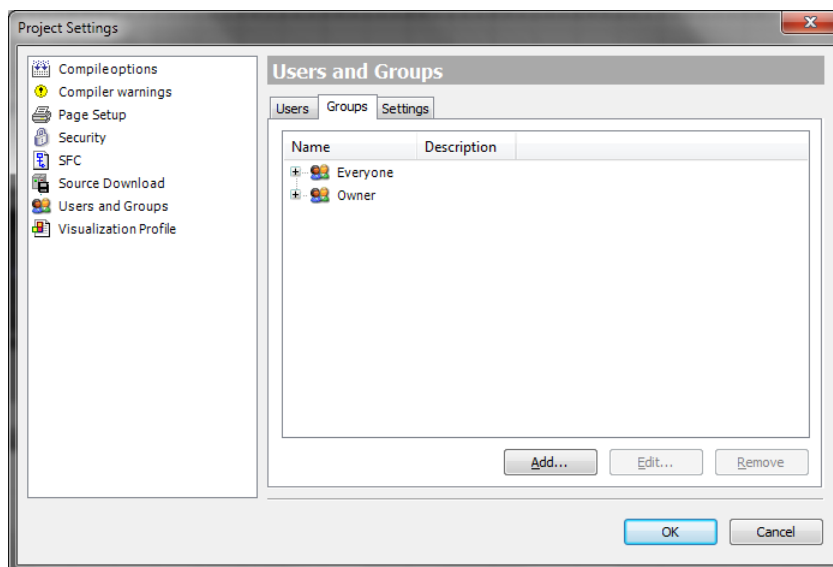
4.13.1.1.2. *Groups*

Figure 141: Project Settings - Groups

The available current groups are listed in a tree structure. The members of each group can be viewed or not by the plus and minus sign, respectively. Remember that the member must be part of a group.

To add a new group, use the *Add* button and open the corresponding dialog.
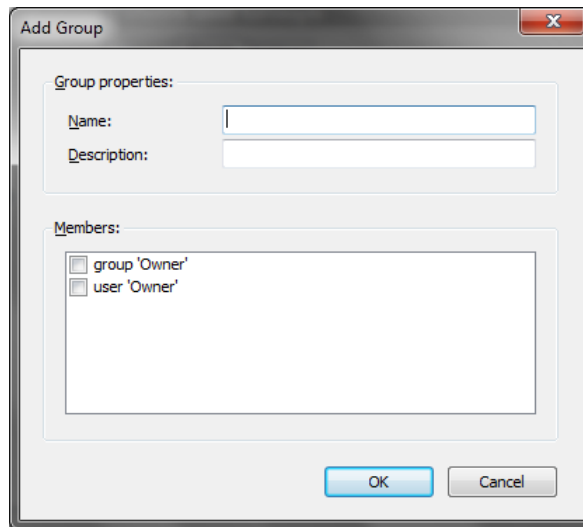
Figure 142: Add Group

The following fields must be filled:

- *Name:* New group's name.
- *Description:* New group's description. Serves only as additional information.
- *Members:* In this list are presented all users and groups. Select those that should be part of the current group.

To configure the new group, close the dialog with *OK*. In case of any inconsistencies (no name, group already existing, selection of a group that would cause a "group cycle"), an error message will be displayed.

To modify an existing group, use the *Edit* button to open the *Edit Group* dialog. The fields are the same as in the *Add Group* dialog (Figure above). The password fields - for security reasons - will have 32 asterisks (*). After changing the desired items, close the dialog with *OK* to apply the new settings.

To remove one or more groups, select the respective groups in the tree and press *Remove*. Note that this action does not require confirmation. The members of the deleted groups will remain unchanged. You cannot delete the *Everyone* and/or *Owner* groups. If the user tries to do this, an error message will appear.
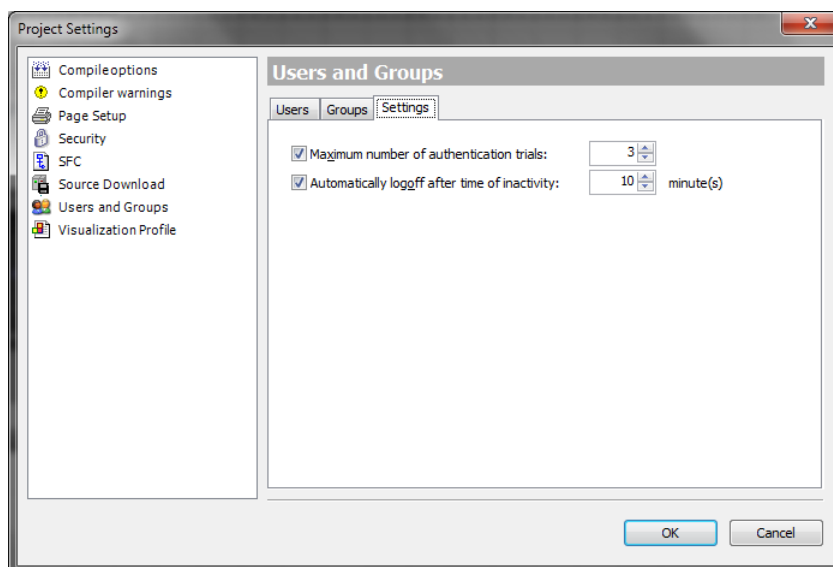
4.13.1.1.3. Settings

Figure 143: Project Settings - User and Groups Settings

The following basic configurations of user accounts can be done:

- *Maximum number of authentication trials:* If this option is enabled, the user account will become invalid after the specified number of attempts to carry out the login with the wrong password. If the option is not enabled, the user can perform as many attempts as you want. Default: enabled option (3 attempts). Allowed values: 1-10.
- *Automatically logoff after time of inactivity:* If this option is enabled, the user account of the connection will be automatically lost after a certain period of inactivity (lack of actions via mouse or keyboard). Default: enabled option (10 minutes). Allowed time values: 1-180 minutes.

4.13.1.2. Access Right's Management

User management in a project is only useful if combined with the management of access rights.

In a new project, basically all access rights are not set automatically, but set to a default value, that is, usually the rights are "guaranteed".

During project execution, each right can be explicitly granted or denied and set back to default. The management of access rights is done in the *Permissions* dialog or - for object access rights - in the *Access Control* dialog (part of the *Object Properties* dialog).

The access rights to the objects are "inherited". If the object has a "main" object, the access rights of this becomes the default settings of the secondary object (example: if an action is assigned to a program, it is inserted in its structure. Thus, the program is the "main" object of the action). With regard to access rights, usually relations of the main objects - side correspond to the relations shown in the POUs or Devices tree and are indicated in the *Permissions* dialog using the syntax "<main object> <secondary object>.".

Example: Action ACT is attributed to MainPrg object (POU). Thus, in the window of the POUs, ACT is shown in the object tree in MainPrg. In the Permissions dialog, ACT is represented by "MainPrg.ACT" indicating that MainPrg is the "principal" of ACT. The right "change" was explicitly denied to MainPrg and to a certain group of users, the default value of this right to ACT would also be "denied" automatically.

To access *Permissions* screen user must click on this option in *Project* menu, after *User Management*. The following screen will appear.
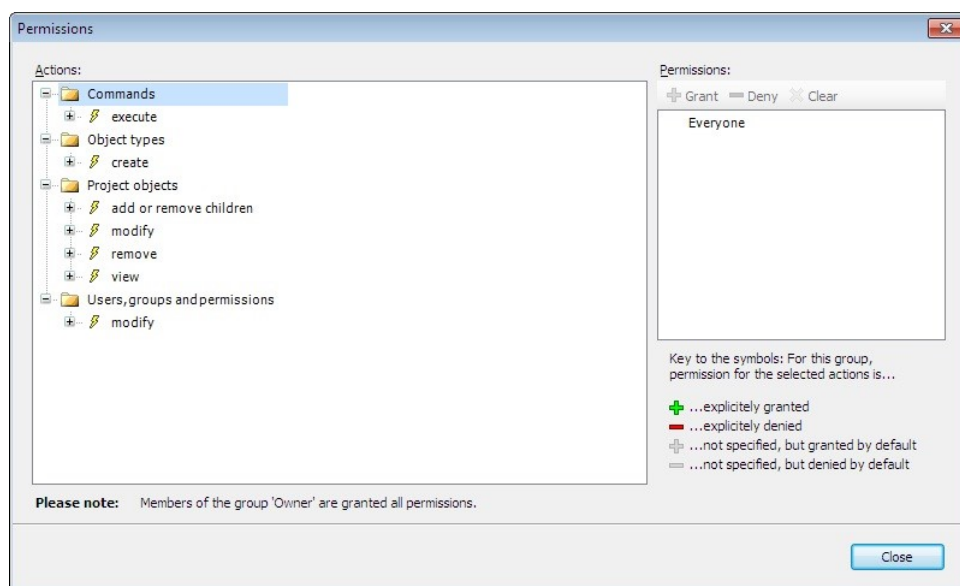

Figure 144: Permissions

The *Actions* window displays all the rights of possibilities, ie all actions that can be performed on objects. The tree is structured as follows:

- : At the top of the tree are the names of some categories configured to visually organize access rights. These categories refer to the Command implementation, user accounts configuration and groups, the creation of types of objects and also to view, edit, delete and treatment of secondary objects in the project Objects.

- : In each category there are the specific actions that can be executed about the command, user account, group, object type or design of the object. These we have also just visual function.

Possible actions:

- *Execute* (command menu execution).

- *Create* (creation of a new object in the actual).

- *Add or remove children* (of an existent object).

- *Modify* (edition of an object in the editor).

- *Remove* (remove or cut an object).

- *View* (view of an object in the editor).

- Each item "action" may contain "devices", that is, objects (of the project).
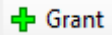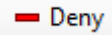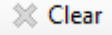
The permissions window provides a list of all available user groups (except the Owner group) and a toolbar to set access rights to the group.

To the left of each group name are icons indicating the current permission for the selected device in the *Actions* window:

- : The currently selected action in the *Actions* window is guaranteed to group.

- : The currently selected action in the *Actions* window is denied the group.

- : The right to perform one or more actions selected in the *Actions* window is not guaranteed explicitly, but by default, due to that right has been guaranteed to the "main" object. Basically this is the default setting for all the rights that have not been explicitly configured.

- : The right to perform one or more actions selected in the *Actions* window is not explicitly denied, but by default, for example, in cases where that right has been assigned to the main object.

The icons are not displayed if they are selected several actions without the unique settings for the selected group at the time.

To configure rights to a group, select an action and the desired group in *Actions* and *Permissions* respectively. After, use the appropriate buttons in the window toolbar *Permissions*:

- : Grant explicitly.

- : Deny explicitly

- : Clear the rights guaranteed to a selected action in the window Actions will be excluded (default Configuration will return)

4.13.2. UCP's User Management and Access Rights

HX3040 CPUs have a user permissions management system, which blocks or allows certain actions for each group of users in the CPU. To edit these rights in the CPU, the user needs to access a project in the MasterTool Xtorm software, and does not need to be logged in to the CPU. The user must then click on the Device Tree, located on the left side of the program, double-click on the item *Device* and then select the CPU in the *Communication Settings* tab that will open. Only the *Users and Groups* and *Access Rights* tabs relate to this topic. The figure below exemplifies the steps to access these tabs on the CPU.
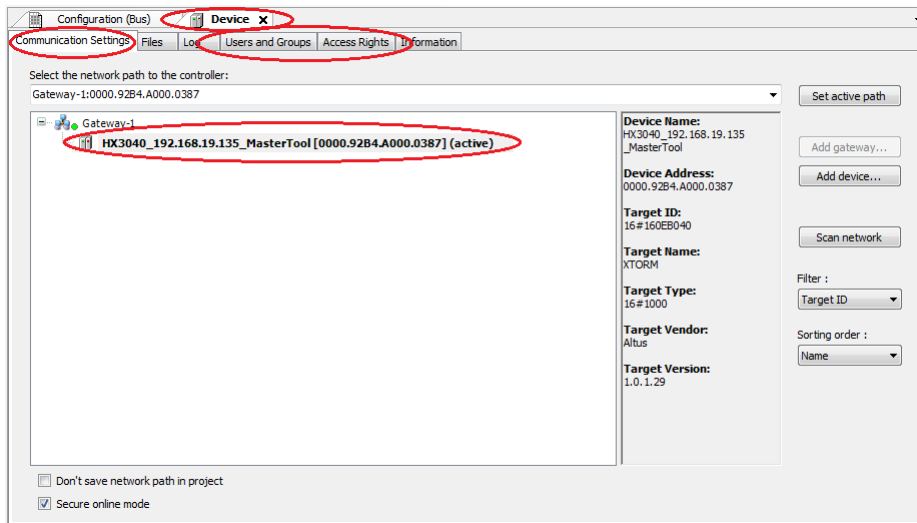
Figure 145: User and Groups and Access Rights tabs access

> **ATTENTION**
>
> If the user forgets the password(s) of the account(s) with access to the CPU, the only way to recover this access is to update the CPU firmware.

4.13.2.1. Users and Groups

The dialog *Users and Groups* is provided in a dialogue guide *Device*. It allows you to configure user and group accounts that, together with the management of access rights control access to objects in the PLC in online mode.

4.13.2.1.1. Common

For some functions of a controller that can be performed only by authorized users, uses the *Online User Management*. This option provides the ability to define user accounts, assign access rights for groups and force the user authentication at login.

The specific user management device can be pre-defined by the description of this. Also depends on the device description which settings can be edited in the configuration dialogs in programmer.

Just as the project user management, users must be members of groups and user groups can only get certain access rights.

4.13.2.1.2. Using the Configuration Dialog

Basically, the treatment of online user management dialog is similar to the design of user management. There is the possibility of "importing" user accounts settings from the project user management.
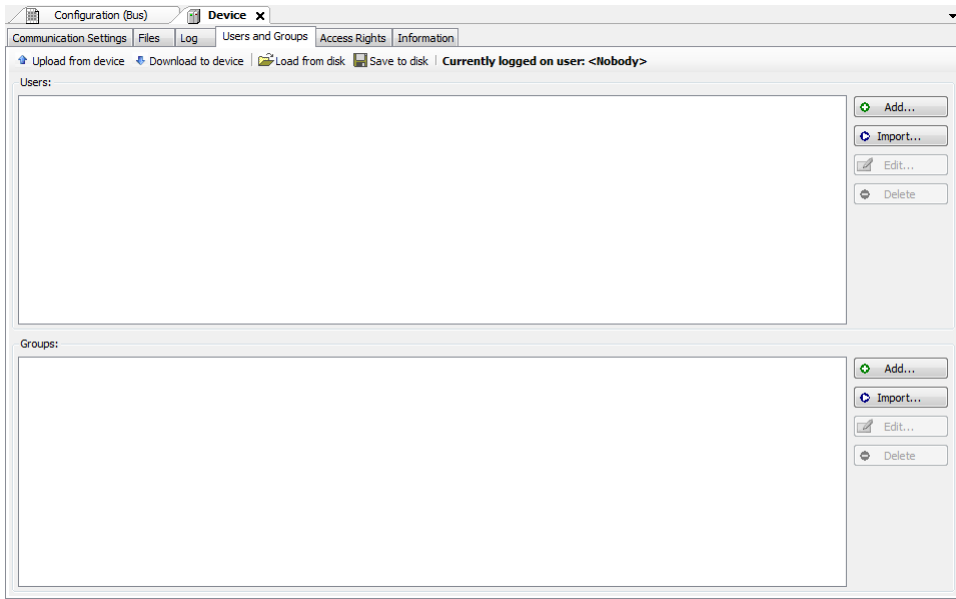
Figure 146: Device Dialogue, User and Groups

4.13.2.1.3. Using the Configuration Dialog - Users

The following buttons are available to configure user accounts:
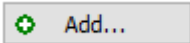
Add... : The dialogue *Add User* defines a user name and password. The password must be retyped in the field *Confirm password*.

> **ATTENTION**
>
> By opening this dialogue the fields *Password* and *Confirm password* will be filled with ficti-tious characters, the user must replace these characters for a valid password.
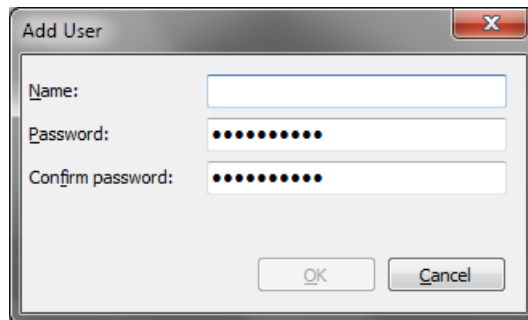
Figure 147: Add User
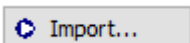
Import... : The *Import Users* dialog shows all the names of users currently defined in the project user management. Select one or more items and confirm with OK. In the *Enter Password* dialog, type the password (as defined in management) so that the user account is imported into the specific user management device.
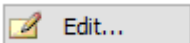
Edit... : The name and password of the current user account selected can be modified. The dialogue *Edit User <user name>* matches the dialogue *Add User*.

Delete : The current selected account will be excluded

altus

4.13.2.1.4. Using the Configuration Dialog - Groups
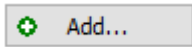
○ Add... : The *Add Group* dialog sets a new name for this and selects, from the currently defined users, those who are part of the group.
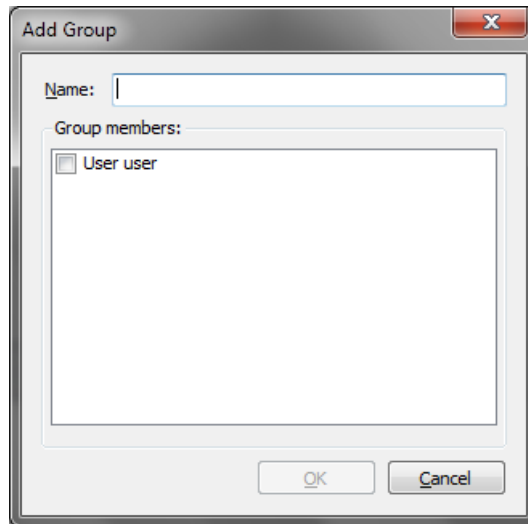
Figure 148: Add Group - CPU
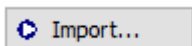
C Import... : The dialogue *Import Groups* presents a list of groups currently defined in user management of the project. Select one or more items and confirm with *OK* to integrate them into the list of the specific user management device groups.
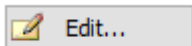
✎ Edit... : The current group selected can be modified which refers to name users related. For this, is used *Edit Group* <groups name>, which corresponds to dialogue *Add Group*.

● Delete : The current selected group will be excluded.

4.13.2.1.5. Applying and Storing the Current Configuration

See the respective buttons on the top bar of the dialog.
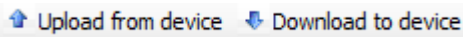
⬆ Upload from device ⬇ Download to device : The current configuration of user management should be sent to the device to make it effective. The settings currently applied to the device can be charged in the configuration dialog.
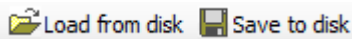
📂 Load from disk 💾 Save to disk : the current configuration can be stored in an XML file (* .dum) and reloaded from this file, which is useful for configuring the same user configuration on multiple systems. The standard dialog to browse files on the system will be provided for this purpose. The file filter is automatically set to *"* .dum"* (specific file *"user management"*).

The current settings can be printed or documented using the commands *Print...*(Archive menu) and *Document...* (Project menu), respectively.

4.13.2.1.6. Considerations of User and Groups Default

In the current firmware versions there are the users: *Administrator* and *Everyone*. And also groups: *Administrator, Developer, Everyone, Service* and *Watch*. As shown in the table below:

Users	Groups
Administrator	Administrator
Everyone	Developer
	Everyone
	Service
	Watch

Table 208: Users and Groups

4.13.2.1.7.   Considerations of User and Groups Default - Users and Groups Default

The following groups and users are defined by default in the Hadron Xtorm Series CPUs. This division into a larger number of groups is to present an initial proposal for different levels of users who can access the CPU.

4.13.2.1.8.   Considerations of User and Groups Default - Users and Groups Default - Administrator Group

This group has all the privileges and you cannot remove it. The *Developer* group is part of this group.

4.13.2.1.9.   Considerations of User and Groups Default - Users and Groups Default - Developer Group

Group created to define access rights to users who are application developers. The *service* group is part of this group. If not used this group can be deleted.

4.13.2.1.10.   Considerations of User and Groups Default - Users and Groups Default - Everyone Group

This is the default group to perform the hits on a CPU while there are no groups and users defined.

4.13.2.1.11.   Considerations of User and Groups Default - Users and Groups Default - Service Group

Group created to define access rights to users to provide some kind of service in CPUs, such as maintenance crews. The *Watch* group is part of this group. If not used this group can be deleted.

4.13.2.1.12.   Considerations of User and Groups Default - Users and Groups Default - Watch Group

Group created to define access rights to users who can only view without making any modification in the application, if not used this group can be deleted.

4.13.2.1.13.   Considerations of User and Groups Default - Users and Groups Default - Administrator User

The Administrator user is defined in the *Everyone* and *Administrator* groups. The default password for the Administrator user is *"Administrator"* and can be modified.

4.13.2.1.14.   Considerations of User and Groups Default - Users and Groups Default - Everyone User

The user Everyone is set to *Everyone* and *Administrator* groups. This user has no password set.

4.13.2.1.15.   Users And Groups Of Old Projects

To keep the users and groups of an outdated project after a firmware update in a CPU or in new Xtorm UCPs its necessary in the old project with the original firmware execute the command *Upload from Device*, thus seeking the CPU settings, and after, the command *Save to Disk*, thus saving the current configuration to a file.

In the new XTorm UCP or the updated UCP, execute the commando *Load from Disk*, and select the previously generated file, finally, execute the command *Download to Device*, thus sending the settings to the CPU.

4.13.2.2. Access Rights

This dialogue is provided on a *Device* dialog tab (Device Editor). It is part of the *Online Users Management* and serves to grant and deny permissions to certain user group defined at the time, thus determining the access rights to files and objects (an application, for example) when the CPU is running.

Note that these permissions can only be assigned to groups and not to single users. Therefore, a user must be defined as a member of a group. The configuration of users and groups is done in the device editor *Users and Groups* tab.

The figure below shows the permission to add and remove child nodes to / CPU Logic object granted to the user group *Everyone e Owner*.
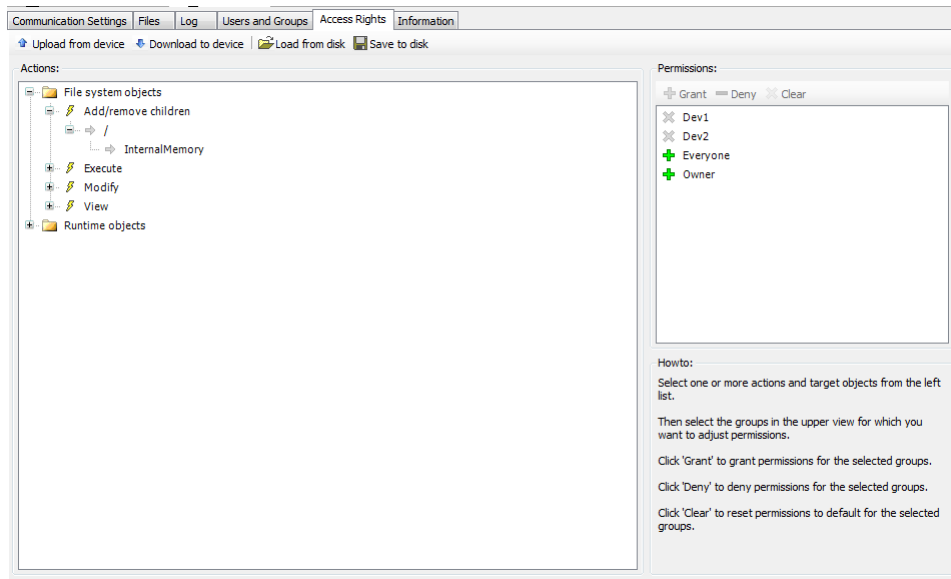
Figure 149: Device Acess Rights

Access rights are configured for each device and enable for each user the actions according to the table below. The access rights can be defined according to the user's needs. The same can define the type of access to the project, being allowed or not to Add / Remove, Modify, View and Execute.

Execution Objects			Access Rights			
			Add / Remove	Modify	View	Execute
Device	Logger	-			X	
	PlcLogic	Application	X	X	X	X
		_Backup&Restore		X	X	
	PlcShell	-		X		
	RemoteConnections	-	X	X	X	X
	Settings	-		X	X	
	UserMenagement	-		X	X	
	X509	-		X	X	

Table 209: Actions and Rights

Note:

Access Rights: These are the permissions that can be configured according to the needs. The table above indicates, according to each object type, which rights / permissions can be set. In addition, it is possible to set the permissions for each device group type, according to the availability of the execution object.

See below how to set access permissions and how to make them to be uploaded to the device or stored in a rechargeable file.

4.13.2.2.1.   Setting Access Permissions

To set permissions to execute an action in one or more objects, select them above the desired action in *Action* window. Then, select the desired group in the window *Permissions* and click on *Grant* or *Deny* button (also in *Permissions* window).

See the following description of specific dialogs.

4.13.2.2.2.   Setting Access Permissions - Actions

This part of the dialog lists the actions that can be performed during the run in files on the CPU file system and execution objects, such as applications. The tree is structured as follows:

   Categories of objects

At the top level for structuring purposes, there are the "folders" relating to the objects of the file system and execution of objects.

   Actions type

In this "folder", there us to the four types of shares that may be running on specific objects. These nodes are used only for structural purposes:

-   *Add or remove children* (addition or removal of "children" objects to an existing object).

-   *Execute* (for example, start/stop application, configure breakpoints, etc).

-   *Modify* (for example, send applications, etc).

-   *View* (monitoring).

-   Objetos (action "devices").

In each action node type, are the *"dispositivos"* (objects) of action (e.g.,   Dispositivo).

These objects mapped in the device tree or structure of the file system, are displayed in a structured way.

> **ATTENTION**
>
> Assign a right of access to a "main node" in the object tree, usually means the "children node" will inherit this setting until you get a very specific definition. However, depending on the device, it may be treated differently. Anyway, the inheritances are not displayed in the dialogues.

4.13.2.2.3.   Setting Access Permissions - Permissions

This field shows the defined user groups. Each group is preceded by icons indicating the permission assigned when referring to the currently selected device in the window *Actions*.

-   : The currently selected actions in the *Actions* window are granted for the group.

-   : The currently selected actions in the *Actions* window are denied for the group.

-   : There is no explicit definition of access rights to the shares currently selected in the window *Actions*.

If several actions without unique settings (for the group) are selected, no icon is displayed

Button bar: after selecting the objects and the desired group (*Actions* and *Permissions* window), one of the following buttons can be used:

-   **Grant** : Explicitly granted access permission.

-   **Deny** : Explicitly denied access permission.

-   **Clear** : The right of access granted to the shares currently selected in the *Actions* window will be deleted, i.e. it returns to the default.

4.13.2.2.4. Applying and Storing the Current Configuration

See the respective buttons on the top bar of the dialog.

⬆ Upload from device ⬇ Download to device : The definitions of the configured access rights should be sent to the device to take effect. The currently applied on the device settings are loaded in the configuration dialog.

📂 Load from disk 💾 Save to disk : The current configuration can be stored in a file-xml (* .drm) and reloaded from this file, which is useful to define the same user configuration on multiple systems. Thus, the standard dialog to search the file system will be provided. The file filter is automatically set to *"* .drm"* (file *"device access rights"*).

The current settings can also be documented in printed versions via command *"Print..."* (menu Archive) or *"Document..."* (menu Project).

4.13.2.2.5. Old Projects Access Rights

To keep the old project access rights on new projects after the firmware update of the CPU or of a new Xtorm CPU, is needed in the old project with the original firmware run the command *Upload From Device*, thus seeking the CPU settings, and after, the command *Save to Disk*, thus saving the current configuration to a file.

In the new UCP Xtorm or updated CPU, run the command *Load from Disk*, and selecting the file generated before eventually execute command *Download to Device*, thus sending the settings for the CPU.

4.14. Management Web Page Access

Developed to perform configuration and diagnostics access to some features. The *Management* tab of the System Web Page has its access protected by user and password, with *admin* as the default value for both fields.

On the Management tab, there are other resources such as *System*, *Network*, *SNMP*, *USB Device*, *Firewall*, *OpenVPN*, and *FTP Server*. The resources available on this tab vary according to the features available for the controller used and can only be accessed after the user has logged in, as shown in the figure below.
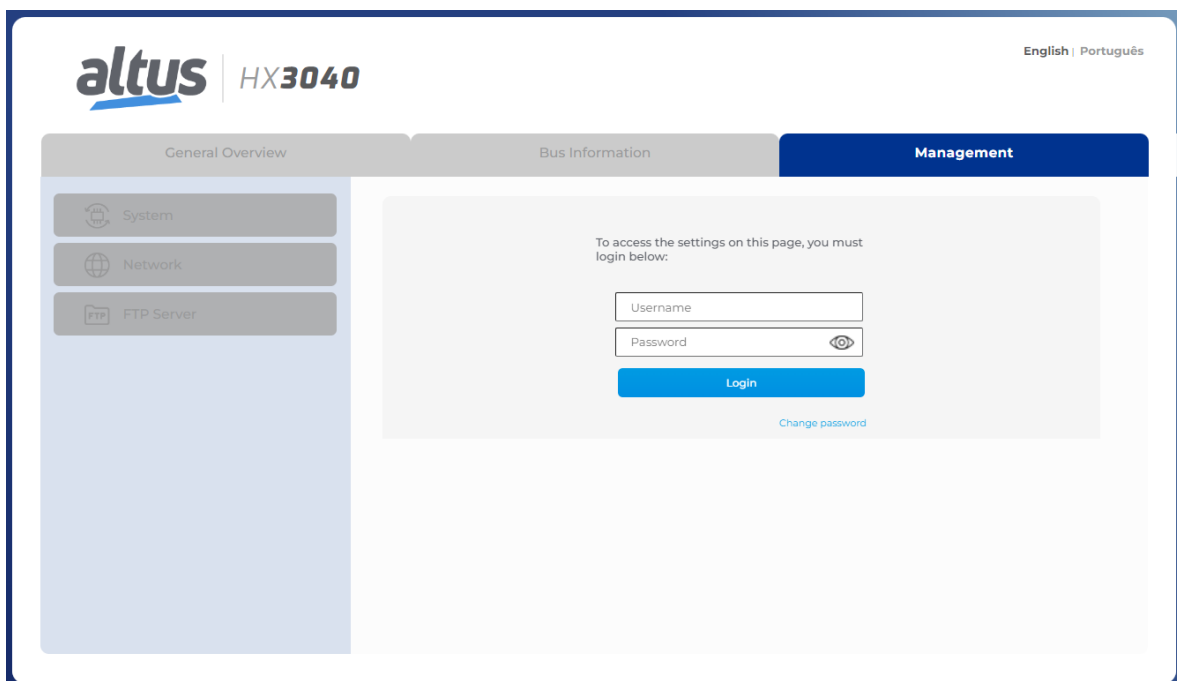
Figure 150: Management Tab Access

4.14.1. System Page

In the *System* section, you can perform a CPU firmware update. For cases in which the update is done remotely (through a radio or satellite connection, for example), the minimum speed of this link must be 128 kbps.

4.14.1.1. Clock Setting

On the System Web Page, it is possible to adjust the controller's clock, which is found in the *System* section of the Management tab. The date and time format follows the ISO 8601 standard for date and time sampling (YYYY/MM/DD hh:mm:ss), as shown in the image below:
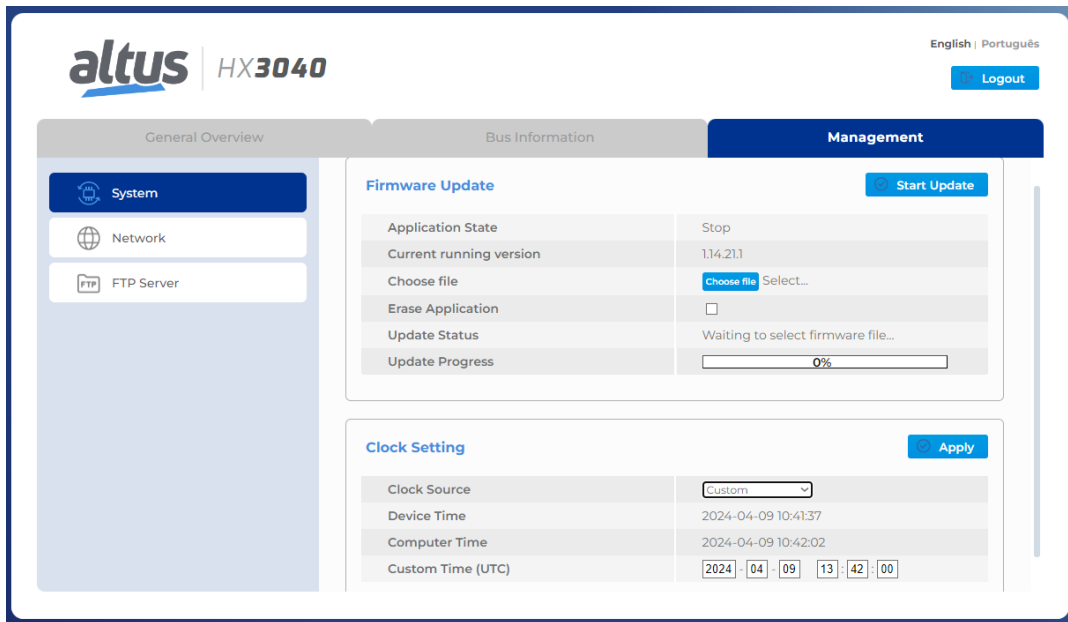
Figure 151: Clock Setting

This feature has two modes for adjusting the device's time, which can be selected in the item "Clock Source", providing the user with two options for synchronizing the clock.

4.14.1.1.1. Computer Time (UTC)

In computer time mode, user can apply the time configured on his computer in UTC for his device. To do so, select the option "Computer" in the "Clock Source" item. After clicking on the "Apply" button, it is necessary to validate the device's credentials, then the CPU will receive the date and UTC time that are configured on the computer.

4.14.1.1.2. Custom Time (UTC)

In the custom time mode, the user can prepare a custom time in UTC standard to be applied to the device's internal date and time. To do so, select the "Custom" option in the "Clock Source" item. With the mode selected, the user must configure the desired date and time in the "Custom Time (UTC)" item, which will be initialized with the browser's local time. So, after the user clicks on the "Apply" button and validates the device's credentials, it will have its internal time configured with the time configured in the item "Custom Time (UTC)".

> **ATTENTION**
>
> The lowest configurable date and time value is 2000/01/01 00:00:00. The highest date and time value is 2035/12/31 23:59:59.

4.14.2.   Network Page

4.14.2.1.   Introduction

Designed to assist in the usability of the controller, the *Network* section (figure below) allows you to change network addresses and run the Network Sniffer.

4.14.2.2.   Page Access

The implementations of the Network page are available in a dedicated section located in the Management tab on the controller's web page, as shown below:
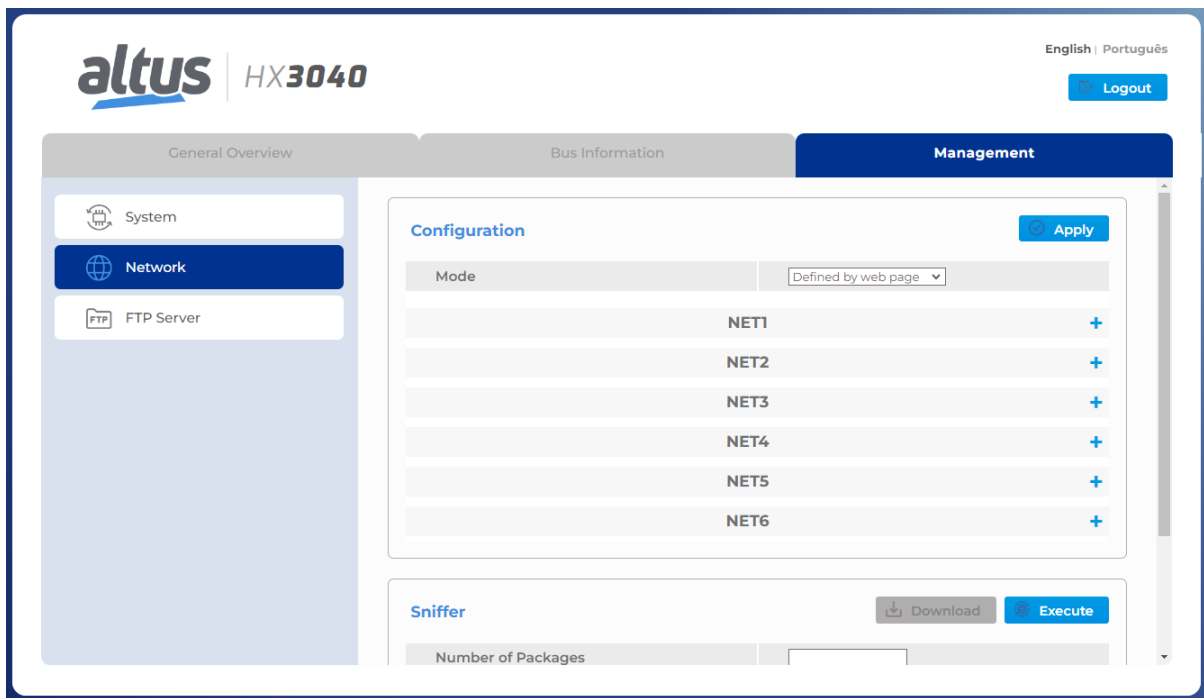
Figure 152: Network Web Page

4.14.2.3.   Network Configuration

4.14.2.3.1.   Defined by Application

The Mode field defines which configuration the controller should load for its interfaces. This field can be configured as *Defined By Web Page* or *Defined By Application*.

When set to *Defined by Application*, the interface table is disabled, not allowing changes, as shown in the figure below. In this mode, the settings applied to the controller are those defined by the application.

> **ATTENTION**
>
> The table for network configuration is displayed only when there is no application on the controller or the controller is not running. It is not possible to change the network settings while an application is running on the controller.

Below is an image with *Defined by Application* mode selected, showing the interface table disabled.
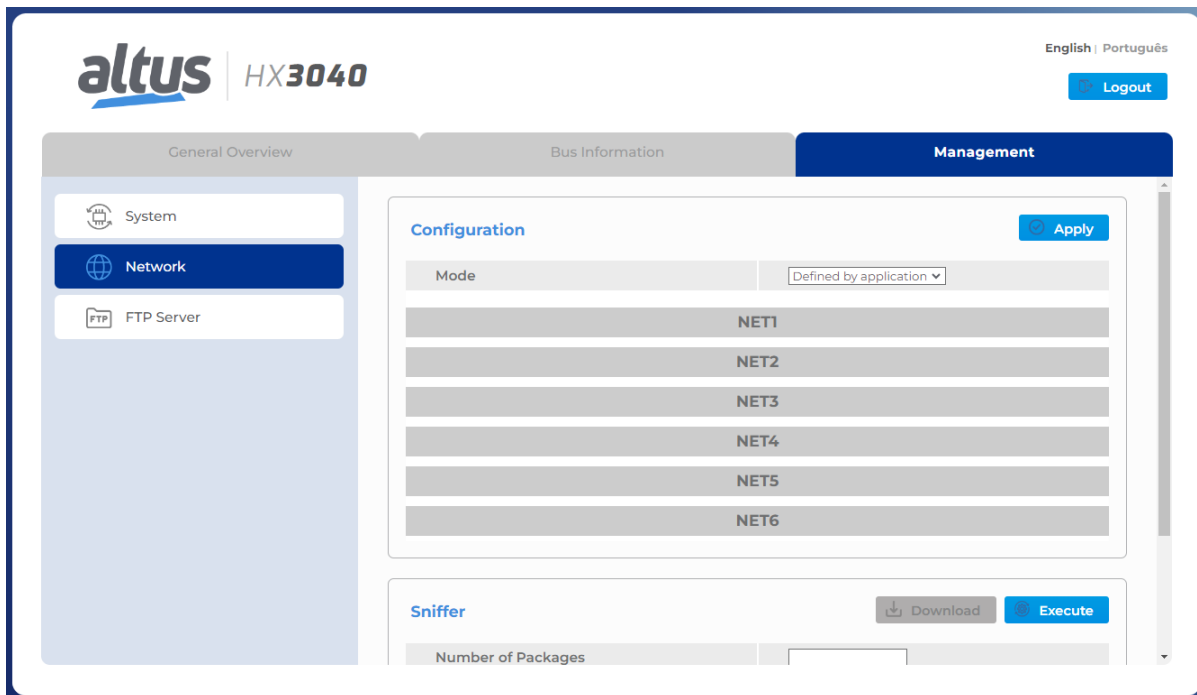
Figure 153: Interfaces Table - Application Mode

4.14.2.3.2. *Defined by web page*

For *Defined by Web Page* mode, the interface table remains enabled as shown in the figure below.

In this mode, the user can set the IP Address, Network Mask, and Gateway for each of the available Ethernet interfaces, as well as enabling and disabling NETs 2 and 3.
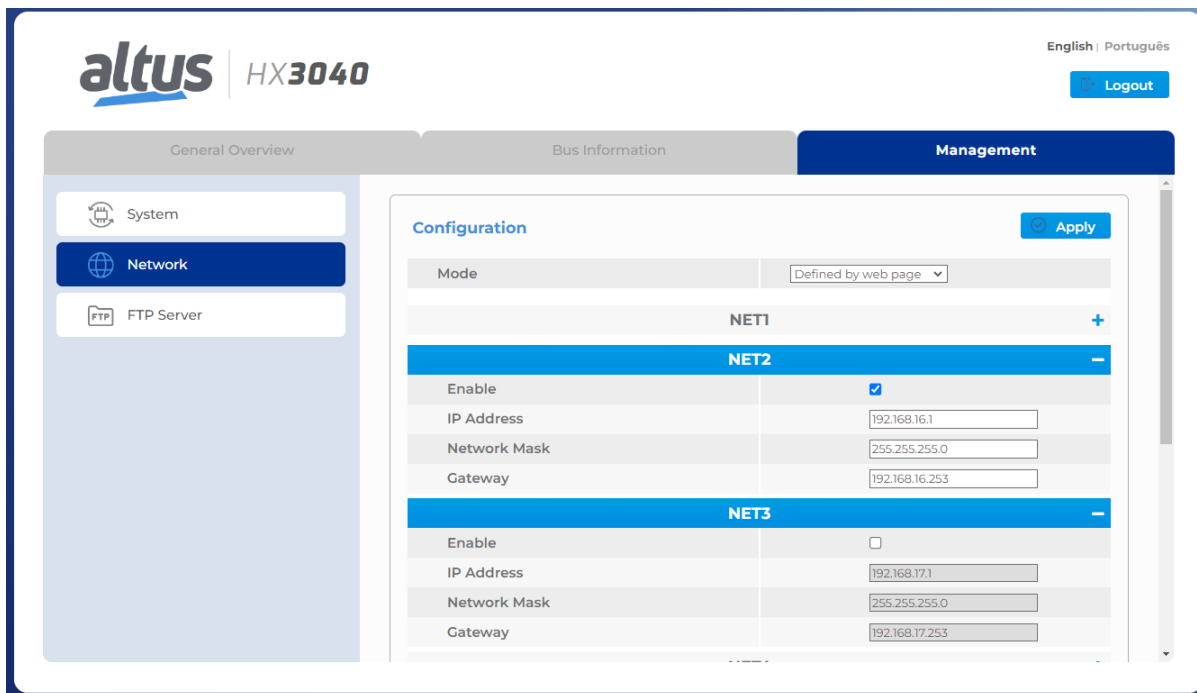
Figure 154: Interfaces Table - Web Mode

The *Enable* checkbox allows the user to enable and disable the Ethernet interfaces. This is only available to be checked or

unchecked when the configuration is *Defined by web page*.

When this checkbox is unchecked, it indicates that the NET interface is disabled, i.e., it will not receive configuration and will be deactivated, as the NET 3 of the figure above. When the interface is enabled (with the checkbox checked), according to NET 2 in the figure above, the settings are available for editing.

To have the settings applied to the controller, simply click the *Apply* button. This process checks if there were any errors in the configuration made and, if so, displays a message on the browser screen indicating the error. If the settings are correct, after clicking *Apply*, a confirmation window appears in the browser to apply the new settings. By clicking *OK*, the settings are sent to the controller and applied.

> **ATTENTION**
>
> When making network changes in the controller, the interfaces will be restarted, which may cause a communication loss. Especially when changing the IP address value.

When applying settings using the *Defined by Application* mode, the controller will assume the configuration that was defined by the loaded application. If there is no application, the current configuration will be maintained, with only the configuration mode being changed.

Using the *Defined by Web Page* mode, the addresses indicated on the web page will be loaded.

> **ATTENTION**
>
> The *Defined by Web Page* mode configures the interfaces to operate in Simple Mode.

It is possible to monitor through MasterTool whether the IP address is configured from the Web Page or from the application by the *bNetDefinedByWeb* diagnostic BIT in the *Application* group, which will change to *TRUE* if the IP is configured from the Web Page and to *FALSE* if it is configured from the application.

4.14.2.4. Network Sniffer

The network sniffer, shown in the figure below, can be used to observe traffic on physical interfaces, except for USB devices such as modems and wifi adapters. It has two basic settings:

Number of Packets: This is the number of packets you want to capture. The configured value of this parameter must be within the range of 100 to 25000 packets;

Idle Timeout (seconds): If there is no packet traffic on the interface after this configured timeout, the Sniffer execution is terminated. It can be configured with values between 1 and 3600 seconds.

Using the *Interfaces* table, you can select which interfaces you want the Sniffer to run on, i.e., perform network analysis. You can select all available interfaces and run them on all simultaneously. For disabled interfaces, it is not possible to run Sniffer. If the selected option is disabled, an error will be displayed in the browser.

Only a few moments after the screen opens will the *Execute* button, which starts Sniffer's execution, become available. The *Download* button will only be unlocked if there is a Sniffer related file available for download. If the Sniffer has never been run or the file is deleted, the button will not be available.

When running the Network Sniffer, the page will disable the edit fields, the *Download* button will be locked, and the *Execute* button will become the *Stop* button, as shown in the figure below.
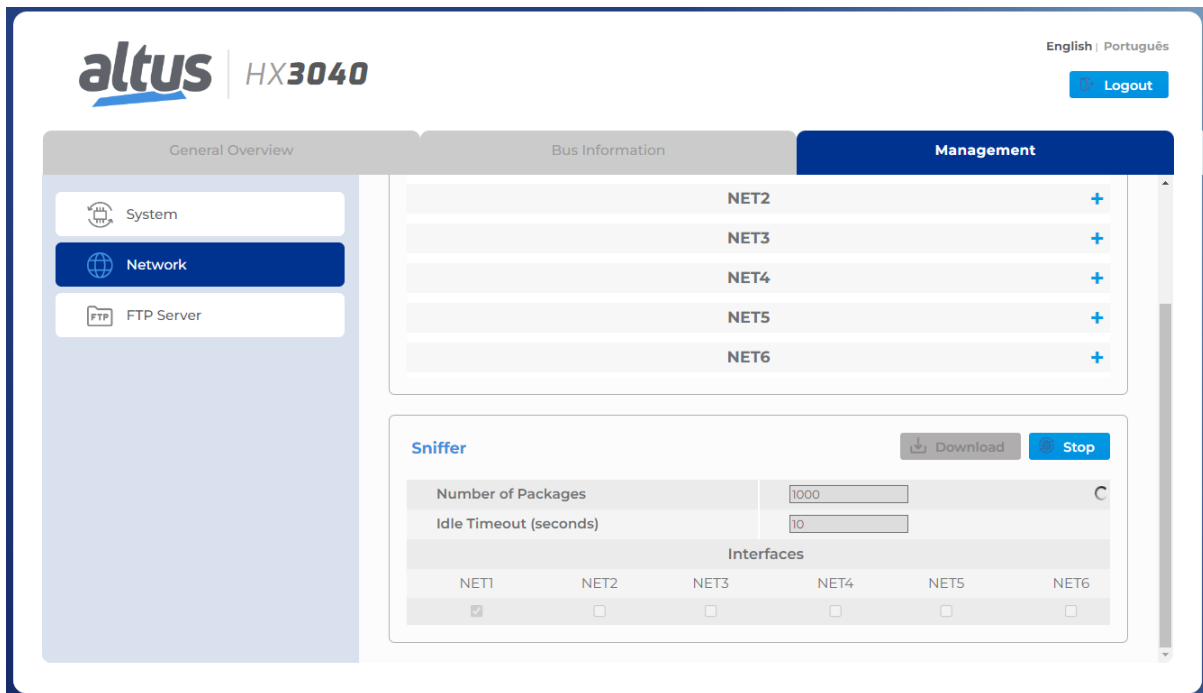
Figure 155: Network Sniffer Running

The *Stop* button can be used to end the sniffer execution at any time after it has been started.

For each of the interfaces on which Sniffer runs, it generates a **.pcap** file. These files are named according to the name of the controller and the interface that was analyzed, for example, **HX3040_NET1.pcap**. These files are found inside a **.zip** file, also named according to the name of the controller, for example, **HX3040_capture.zip**.

At the end of the sniffer execution, a message is displayed asking whether or not to automatically download the generated files. These files are stored in the *InternalMemory* folder of the User Files Memory and can be accessed through the controller's programming software. The downloaded file is always in the **.zip** extension, which groups the other files.

If any problems occur related to insufficient memory due to the generation of sniffer files, it will be indicated to the user. It is recommended to try running the analyzer again with a smaller *Number of Packets* configuration.

The network sniffer can terminate its execution for three reasons: insufficient memory, idle time limit of interfaces exceeded, and manual cancellation.

4.15. FTP Server

FTP (File Transfer Protocol) is a protocol that allows files to be transferred between devices. It operates in the client-server mode, where the CPU becomes a FTP Server, storing files that FTP Clients can access for transfer, download, and upload.

The controlling FTP connection is a TCP connection established through port 21 of the FTP Server. Through port 21, the Client and Server exchange commands and responses to manage the file transfer session.

Through the FTP protocol, the FTP Client can read and write files that are stored either in the internal memory of the CPU or in external memory (memory card, for example) if present in the architecture. The maximum file size that can be transferred varies according to the amount of memory available between internal and external memory. When the memory limit is reached, the transfer will stop, and if the file has not been transferred completely, it will become a corrupted file.

> **ATTENTION**
>
> Downloading and uploading large files through FTP, both from internal and external memory, can affect the performance of the CPU considerably.

> **ATTENTION**
>
> The FTP server does not support communication through Windows File Explorer, so an FTP Client software is required to access it.

4.15.1. Configuration

The FTP configuration is done through a dedicated section located in the *Management* tab of the controller's System Web Page, as shown below:
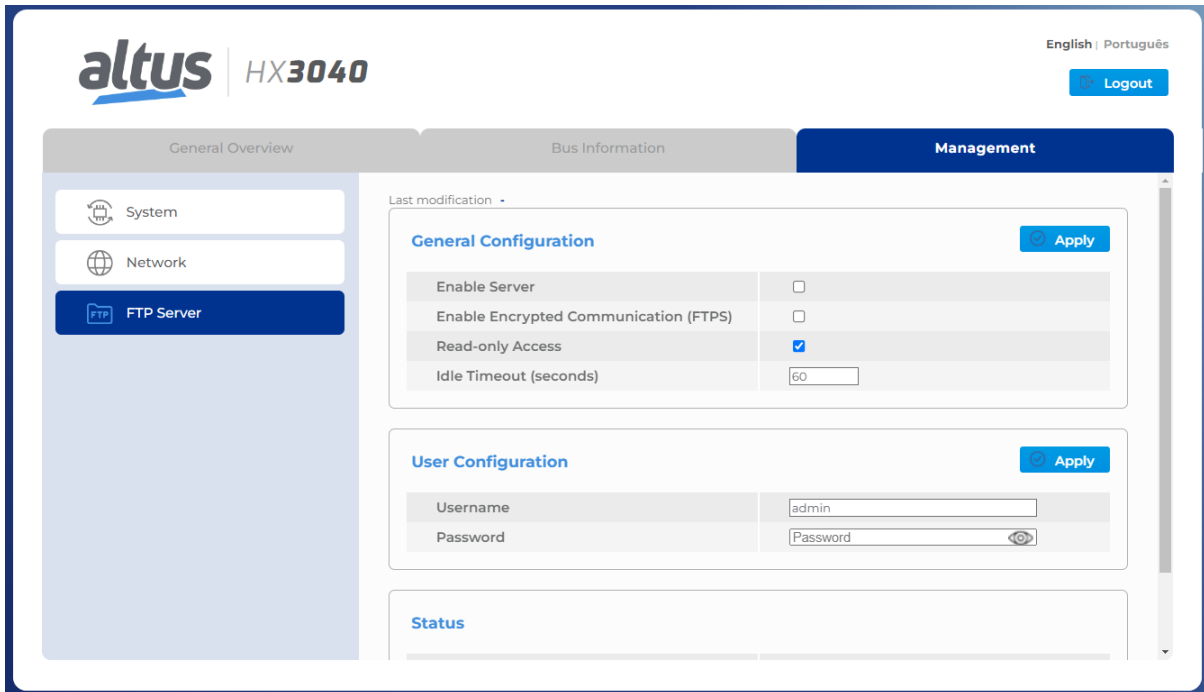
Figure 156: Tela de configuração do Servidor FTP

FTP is a separate feature from MasterTool IEC XE, meaning it does not require any interaction with the programming tool. The settings applied on the *FTP Server* section take effect when confirmed through the *Apply* button and are automatically saved in the controller. As long as the functionality is enabled, it will resume operation even after the device is restarted.

4.15.1.1. General Configuration

4.15.1.1.1. Enable Server

Allows to activate or deactivate the functionality. When the FTP server is enabled, the settings made through the System Web Page are applied to the configuration files. That means the server is available according to the configuration made. If FTP is disabled, the configuration is still stored, but the service cannot be used.

4.15.1.1.2. Enable Encrypted Communication (FTPS)

Enables and disables encrypted communication. This communication is done through Explicit FTPS, also known as FT-PES. This is a secure extension of the FTP protocol, which adds a layer of encryption to the transfer.

In this type of communication, when a client-server connection is established, instead of immediately starting the data transfer, the FTP Client sends an AUTH SSL command to request a secure connection. Then, when enabling encrypted communication, the UCP generates a self-signed certificate to guarantee communication using the SSL (Secure Sockets Layer) protocol.

When the FTP Client performs the authenticated connection, the certificate and its respective information will be displayed, allowing Explicit FTPS communication (FTP over SSL) to be established through port 21.

> **ATTENTION**
>
> The FTPES certificate is valid for one year from the date it was created. To generate a new certificate, simply reapply the configuration.

4.15.1.1.3. Read-only Access

Enables and disables limiting write and read access to the Server. By default, the parameter is enabled.

When the parameter is enabled, the FTP Client has read-only access without the possibility to add or remove any files from the FTP Server. In this case, the only operation allowed is uploading files, that is, transferring them from the Server to the Client. When the parameter is disabled, all operations can be performed.

4.15.1.1.4. Idle Timeout (Seconds)

Sets the maximum time the connection will be maintained before the FTP Server closes it due to inactivity. In other words, once the connection to a FTP Client is opened, if there is no activity after the Idle Timeout, the connection to the Client is closed by the Server.

The parameter can be configured with values between 10 to 60 seconds, the default being 60 seconds.

4.15.1.2.   User Configuration

4.15.1.2.1. Username

User or *Username* required for the Client to connect to the Server.

If a new configuration is made, the previous one is removed. In other words, there is only one FTP user, but this one can be used for multiple connections.

4.15.1.2.2. Password

Manages the authentication key so the FTP Client can connect to the FTP Server.

There is no password recovery so, if the password has been lost, it is necessary to add a new user configuration.

Configurable Item	Minimum Size	Maximum Size	Allowed Characters	Default
Username	4	30	[a-z][A-Z][0-9]@$*_.	admin
Password	4	30	[a-z][A-Z][0-9]@$*_.	admin

Table 210: FTP User Settings

4.15.1.3.   Status

4.15.1.3.1. Current State

Displays the current status of the FTP Server. The possible states are *"Running"*, *"Not Running"*, and *"Restarting Service"*. Each configuration applied will restart the service.

4.15.1.3.2. Connected Clients

Shows the user the current number of active connections.

The FTP Server accepts a maximum of two active connections simultaneously. Some FTP Clients, such as *Filezilla*, use a feature called Multithread File Transfer to improve the efficiency and speed of the transfer. This feature allows the FTP Client to open more than one connection for the transfer of a file. Consequently, when using an FTP Client of this type, just one Client already counts as two active connections on the Server.

Other FTP Clients, such as the command terminal, only use one active connection, allowing two FTP Clients to connect to the Server simultaneously.

altus

5. Initial Programming

The main goal of this chapter is to help in the programming and configuration of Hadron Xtorm Series CPUs so that the user will be able to take the first steps before starting a controller programming.

The Hadron Xtorm Series CPUs use the standard IEC 61131-3 languages IL, ST, LD, SFC, and FBD. IL and ST are textual languages similar to Assembly and C language respectively. LD, SFC and FBD are graphical languages. LD uses the representation of relays and blocks and is similar to relay diagrams. SFC uses the representation of a sequential diagram and allows a clear visualization of the functions performed in each action. This Series CPUs also offer a sixth language called CFC.

The programming is made through the MasterTool Xtorm interface. MasterTool Xtorm allows the use of the six languages in the same project, so the user can apply the best features offered by each language, resulting in more efficient applications development, for easy documentation and future maintenance.

> **ATTENTION**
>
> The physical configuration as well as the order of arrangement of the modules on the bus(es) must be the same as the one configured via MasterTool so that no diagnostics are generated and the system is able to operate correctly.

5.1. Memory Organization and Access

Hadron Xtorm Series uses an innovative memory organization and access feature called *big-endian*, where the most significant byte is stored first and will always be the smallest address (e.g. %QB0 will always be more significant than %QB1, as in the table below, where, for HADRON XTORM CPU, string, the letter U is byte 0 and the letter O is the byte 7).

Besides this, the memory access must be done carefully as the variables with higher number of bits (WORD, DWORD, and LONG), use as index the most significant byte, in other words, the %QD4 will always have as most significant byte the %QB4. Therefore, it will not be necessary to make calculus to discover which DWORD correspond to defined bytes. The table below shows little and big endian organization.

MSB ← **Little-endian** → LSB							
BYTE	%QB7	%QB6	%QB5	%QB4	%QB3	%QB2	%QB1	%QB0
	U	C	P	X	T	O	R	M
WORD	%QW3		%QW2		%QW1		%QW0	
	UC		PX		TO		RM	
DWORD	%QD1				%QD0			
	UCPX				TORM			
LWORD	%QL0							
	UCPXTORM							
MSB ← **Big-endian** → LSB							
BYTE	%QB0	%QB1	%QB2	%QB3	%QB4	%QB5	%QB6	%QB7
	U	C	P	X	T	O	R	M
WORD	%QW0		%QW2		%QW4		%QW6	
	UC		PX		TO		RM	
DWORD	%QD0				%QD4			
	UCPX				TORM			
LWORD	%QL0							
	UCPXTORM							

Table 211: Memory Organization and Access Example

SIGNIFICÂNCIA					SOBREPOSIÇÃO							
Bit	Byte	Word	DWord	LWord	Byte		Word		DWord			
%QX0.7												
%QX0.6												
%QX0.5												
%QX0.4	%QB 00				%QB00							
%QX0.3												
%QX0.2												
%QX0.1												
%QX0.0		%QW					%QW					
%QX1.7		00					00					
%QX1.6												
%QX1.5												
%QX1.4	%QB 01				%QB01							
%QX1.3												
%QX1.2												
%QX1.1												
%QX1.0			%QD					%QW	%QD			
%QX2.7			00					01	00			
%QX2.6												
%QX2.5												
%QX2.4	%QB 02				%QB02							
%QX2.3												
%QX2.2												
%QX2.1												
%QX2.0		%QW					%QW		%QD			
%QX3.7		02					02		01			
%QX3.6												
%QX3.5												
%QX3.4	%QB 03				%QB03							
%QX3.3												
%QX3.2												
%QX3.1												
%QX3.0				%QL				%QW		%QD		
%QX4.7			00					03		02		
%QX4.6												
%QX4.5												
%QX4.4	%QB 04				%QB04							
%QX4.3												
%QX4.2												
%QX4.1												
%QX4.0		%QW					%QW			%QD		
%QX5.7		04					04			03		
%QX5.6												
%QX5.5												
%QX5.4	%QB 05				%QB05							
%QX5.3												
%QX5.2												
%QX5.1												
%QX5.0			%QD					%QW			%QD	
%QX6.7			04					05			04	
%QX6.6												
%QX6.5												
%QX6.4	%QB 06				%QB06							
%QX6.3												
%QX6.2												
%QX6.1												
%QX6.0		%QW					%QW					
%QX7.7		06					06					
%QX7.6												
%QX7.5												
%QX7.4	%QB 07				%QB07							
%QX7.3												
%QX7.2												
%QX7.1												
%QX7.0												

MSB ⇑ LSB

MSB ⇑ LSB

Table 212: Memory Organization and Access

The table above shows the organization and access to memory, exemplifying the byte significance and the disposition of the other variable types, including overlapping:

5.2. Project Profiles

A project profile in MasterTool Xtorm is a group of rules, common features and patterns used in the development of an industrial automation solution. A profile implies in the implementation of the application form. Following a profile is a way to simplify the programming complexity. The applications can be created according the following profiles:

- Profile for RTU.
- Custom Profile.

For each profile defined for the RTS, MasterTool Xtorm provides several compatible templates. When the user selects a template as a model in project creation, the new application will be developed as a specific profile, adopting rules, characteristics and standards defined by the profile associated with the template. Each project profile defines standard names for tasks and programs, which are pre created by the project templates. The developer is required to follow the nomenclature strictly for the tasks, but can follow or not the suggested names for the respective programs.

To ensure compatibility between the project and the profile across the application development, two approaches are used:

- MasterTool IEC Xtorm allows the project creation only based on a template, selected at the same time as the profile to be used
- At the code generation, the MasterTool Xtorm executes the verification of all defined rules for the user valid profile.

The next sections detail the features or patterns of each project profile. Based on these definitions, it is recommended that the user always try to use the simplest profile that meets the needs of their application, migrating to the more sophisticated one only when the corresponding rules are more of a hindrance to development than a didactic simplification.

It should be noted that the programming tool allows the change of the profile of an existing project, but it is up to the developer to make any necessary adjustments so that the project becomes compatible with the rules of the new profile selected.

> **ATTENTION**
>
> In the course of the project profiles some types of tasks are named, which are described in the section Task Configuration.

5.2.1. Profile for RTU

The *Profile for RTU* can be created for both redundant and non-redundant projects. In this type of profile, the application can have up to two user tasks, the *"MainTask"* that is always created by default for simple and redundant projects, and the *"ProtTask"* that will only be created if the user defines its use in the project. The *"MainTask"* is responsible for executing a single programming unit of type *"Program"* called *"MainPrg"*. This program cannot be edited and is responsible for calling four other programming units of type *"Program"* called *"StartPrg"*, *"EngineeringPrg"*, *"AlarmPrg"*, and *"UserPrg"*.

The *"StartPrg"* and *"UserPrg"* programs are the only ones from the *"MainTask"* that can contain user code and can call other programming units of type *"Program"*, *"Function"* or *"Function Block"*, but the user code will be executed by the *"MainTask"*. The *"EngineeringPrg"* and *"AlarmPrg"* programs, on the other hand, are created automatically and are responsible, respectively, for performing the engineering conversions and signaling the alarms mapped by the user on the CPU configuration screens. For more information about alarms and engineering conversion, see the CPU Configuration section.

The *"ProtTask"* is also responsible for executing a single programming unit of type *"Program"*, whose name is *"ProtPrg"*. This program cannot be edited and is responsible for calling *"UserProtPrg"* which, in turn, can call other programming units and is responsible for executing the project protection routines.

In this profile, the *"MaintTask"* will be of type *cyclic* with priority fixed as thirteen (13) and executes exclusively the *"MainPrg"* program, while the *"ProtTask"* task will also be of type cyclic, but will have its priority fixed as one (1), being thus of higher priority than the previous one, and executes exclusively the *"ProtPrg"* program.

The tasks *"MainTask"* and *"ProtTask"* are already completely defined and the developer needs to create the program *"StartPrg", "UserPrg"* and *"UserProtPrg"* by choosing any of the languages of the IEC 61131-3 standard. It is not always possible to convert a program to another language, but it is always possible to create a new program with the same name in replacement that is built in a different language.

The default option of Mastertool Xtorm is to use the *"Mastertool Standard Project"* associated with the *"Profile for RTU"* profile, which includes the *"StartPrg", "UserPrg"* and *"UserProtPrg"* programs created in the languages chosen during project

creation, and the *"ProtPrg", "MainPrg", "AlarmPrg"* and *"EngineeringPrg"* programs always in the ST (Structured Text) language.

Task	POU	Priority	Type	Default	Options	Event
MainTask	MainPrg	13	Cyclic	20 ms	5 to 100 ms	-
ProtTask	ProtPrg	1	Cyclic	4 ms	4 to 20 ms	-

Table 213: Task in Profile for RTU

5.2.2. Custom Profile

The Custom project profile allows the developer to explore all the potential of the Runtime System implemented in the CPUs. No functionality is disabled; no priority, task and programs association or nomenclatures are imposed. The only exception is for MainTask, which must always exist with this name in this Profile.

Beyond the real time tasks, with priority between 00 and 15, which are scheduled by priority, in this profile it is also possible to define tasks with lower priorities in the range 16 to 31. In this range, it's used the Completely Fair Scheduler (time sharing), which is necessary to run codes that can be locked (for example, use of sockets).

The developer is free to partially follow or not the organization defined in other project profiles, according to the characteristics of the application. On the other hand, the Custom model associated with this profile needs no pre-defining elements such as task, program or parameter, leaving the developer to create all the elements that make up the application.

Tasks	POU	Priority	Type	Interval	Event
MainTask	MainPrg	13	Cyclic	20 ms	-
CyclicTask00	CyclicPrg00	13	Cyclic	200 ms	-
CyclicTask01	CyclicPrg01	13	Cyclic	500 ms	-
ExternInterruptTask00	ExternInterruptPrg00	02	External	-	IO_EVT_0
TimeInterruptTask00	TimeInterruptPrg00	01	Cyclic	20 ms	-
ExternInterruptTask01	ExternInterruptPrg01	11	External	-	IO_EVT_1
TimeInterruptTask01	TimeInterruptPrg01	09	Cyclic	30 ms	-
FreeTask	FreePrg	31	Continuous	-	-

Table 214: Custom Profile Tasks

> **ATTENTION**
>
> This controller does not support the IO_EVT_0 and IO_EVT_1 events; therefore, the user must choose between the COM1_CTS or COM1_DCD options.

5.3. New Project

As previously described, there are different types of project profiles, depending on the technical knowledge of each user. This chapter describes the creation of a new project using the Single profile, through the "Wizard" tool, which presents all configuration options from the user system.

Initially, the user must create a new project in MasterTool Xtorm from the *File* menu and then *"New Project..."*, as shown in the figure below.

altus
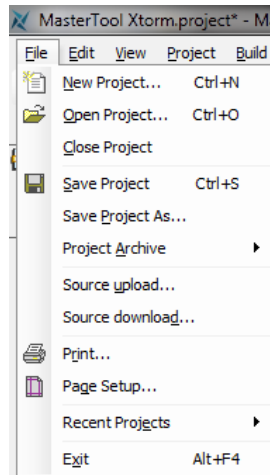
Figure 157: New Project

Afterwards, a window will be presented to the user, asking them to select the type of project they want to create, and then type a name and location to store the project on the computer. Click *OK* to proceed or *Cancel* to abort.
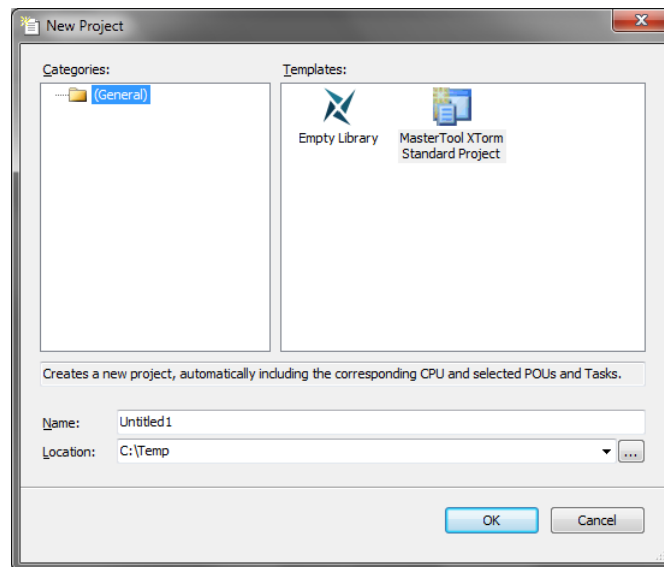
Figure 158: Project Classification

Next the user must select the desired CPU, the basic hardware modules that compose the bus, that is, the rack and power supply model and the redundancy configuration. In this case the UCP HX3040 (no redundancy), a HX9001 rack and a HX8320 power supply (no redundancy) will be used.
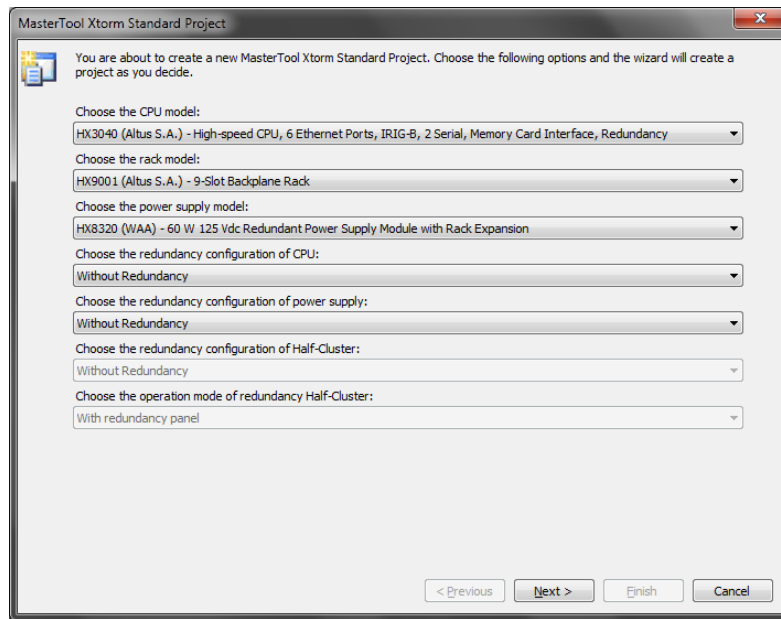
Figure 159: Project Options

If the user selects the option for CPU redundancy, the option for power supply redundancy is automatically checked. This is a project characteristic, and for projects with CPU redundancy, power supply redundancy will be mandatory. In the figure below a screen with the selected option for CPU redundancy can be seen.
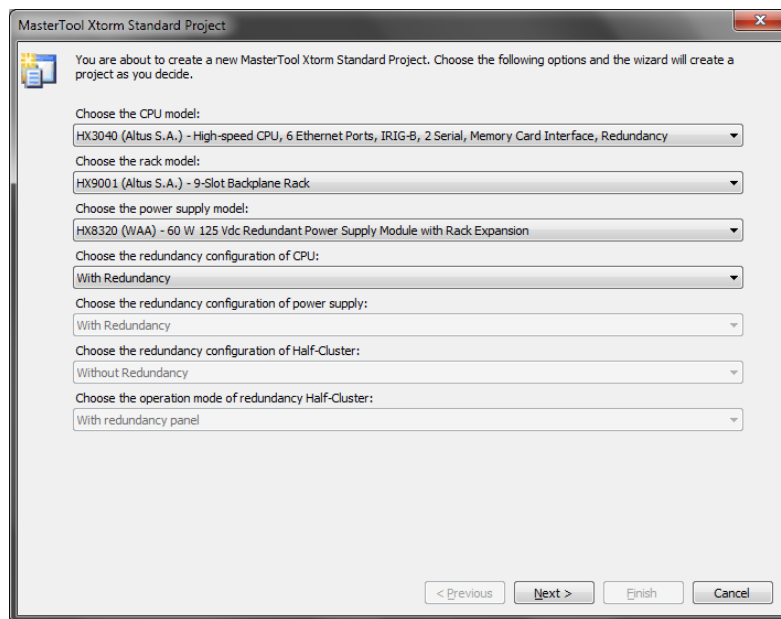
Figure 160: Project Options with CPU Redundancy

On the next screen, the user must configure the number of modules that will be used in the project, and then the Wizard will create the objects of these modules inside the project automatically.
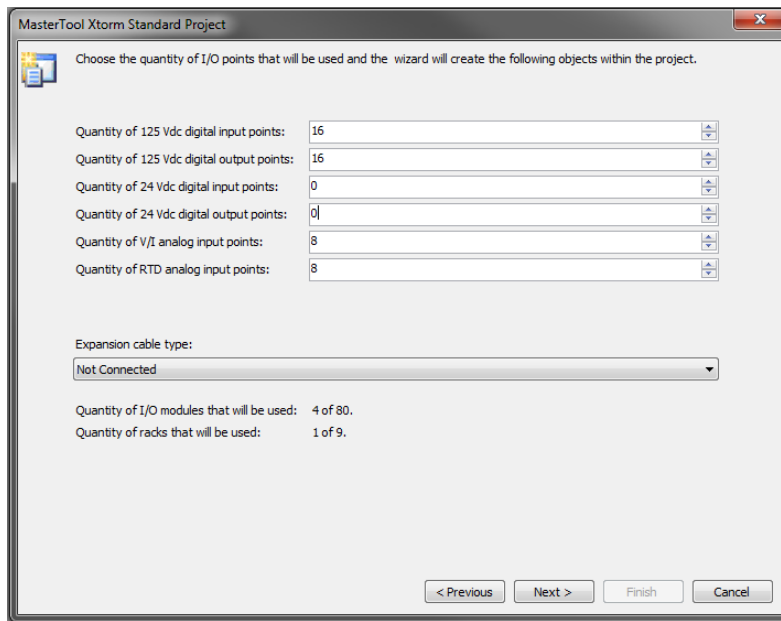
Figure 161: Options of I/O Modules

Then the user must select the profile for the project, and the default language for the POUs (programs). In this case the new project is exemplifying RTU profile without redundancy and ST language. Click *Next* to continue or *Cancel* to abort the project creation.
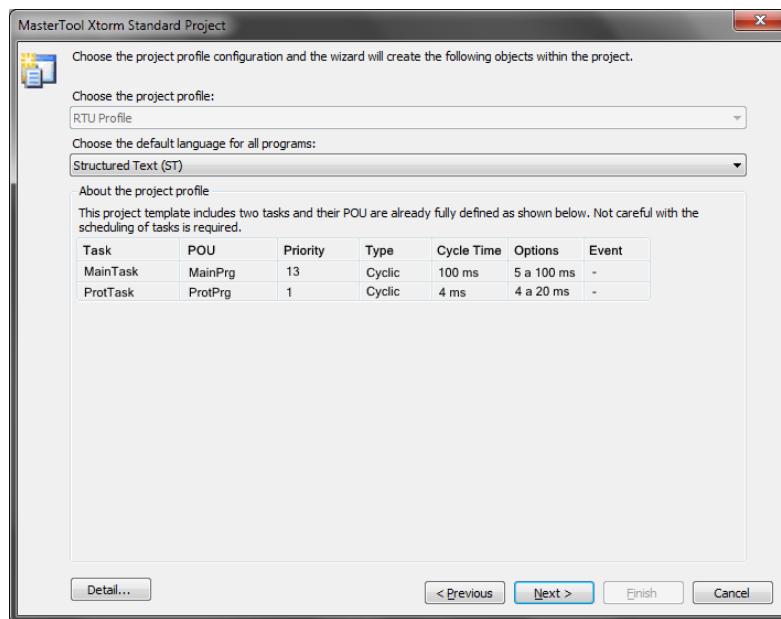
Figure 162: Features Selection of the Project Profile

The next screen defines the language of the POU created by the selected profile. Since the project created is not redundant, there are only two POUs (UserPrg) and (StartPrg) and the ST language remains the same. Click on *Previous* to return to the previous screen, *Finish* to end or *Cancel* to abort.

altus
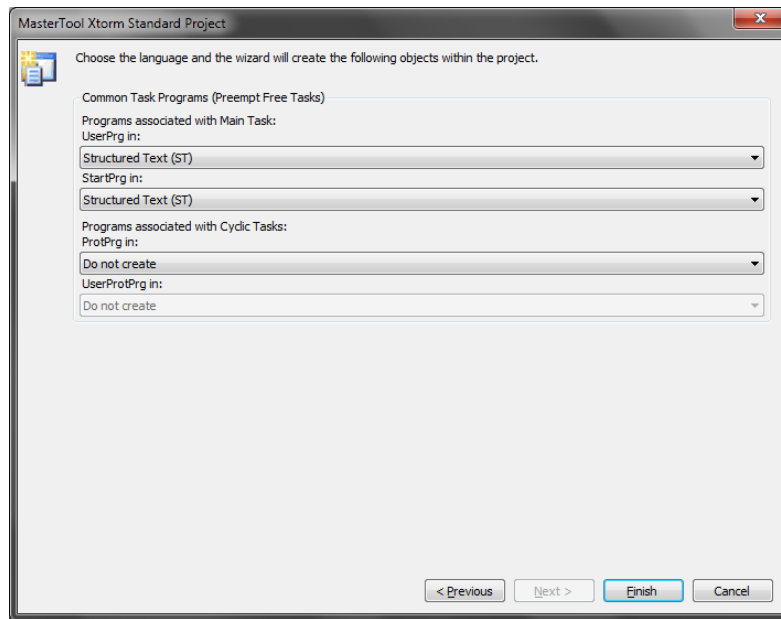
Figure 163: Programming Language

By pressing the *Finish* button, MasterTool Xtorm will start creating the development environment for the project. This procedure may take a few seconds.

5.4. Adding Modules

By default, the CPU and hardware modules selected when creating the project are already inserted into the system configuration. If required, the user can still include other necessary modules. The Xtorm CPU supports up to 100 input/output modules distributed in 16 racks. For further information, see CE123200 of Xtorm Series Power Supplies.

If the *Product Library* tab is not available on the MasterTool Xtorm screen, it must be included, through the *View* menu, by clicking on the "Product Library" item, as shown in the figure below:
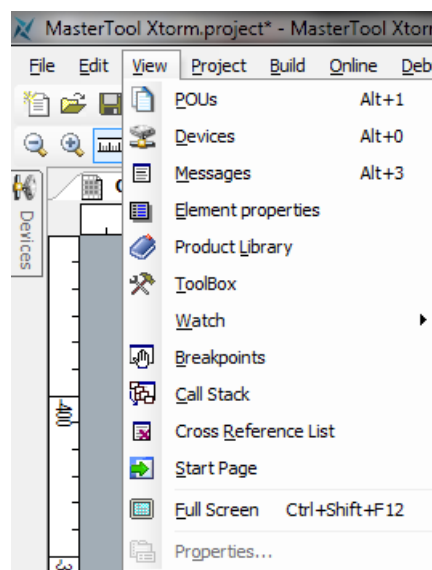
Figure 164: Library Visualization

Then, select the module to be inserted and drag it to the bus configuration area, by pressing the left button of the mouse
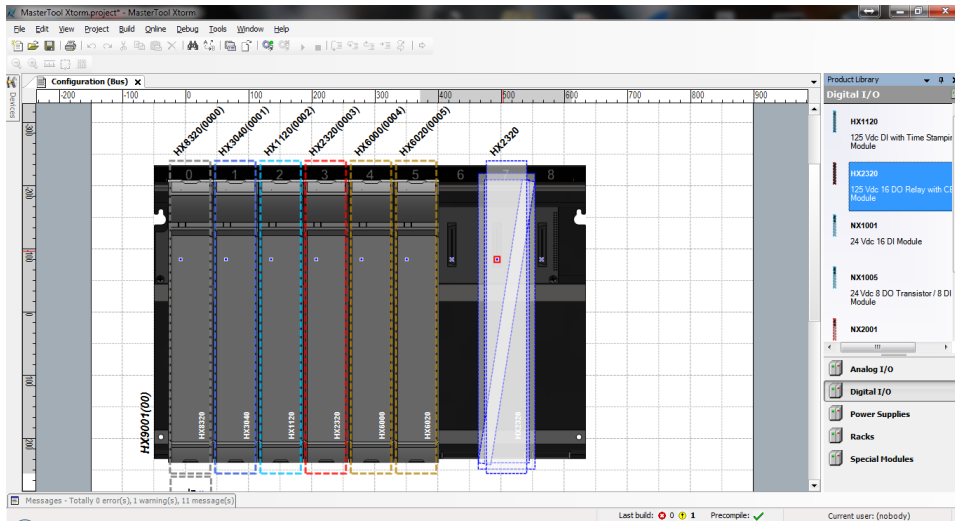
Figure 165: Adding Modules

5.5. Creating POUs

A POU (Program Organization Unit) is a subdivision of the application program that can be written in any of the languages available in MasterTool Xtorm software.

With the creation of the project through a selected profile, some POUs are already created, but the user can create as many as they want, limited by the maximum size of the program memory.

To insert a new POU, just right click on the application name, (default name created for the application is "Application"), select *"Add Object"* and then *"POU..."*, as shown in the figure below:
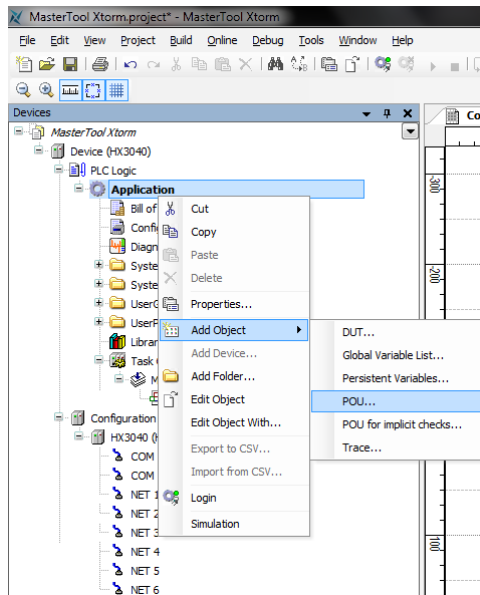
Figure 166: Inserting POUs

A configuration window will appear on the screen, in which the user must enter the name of the POU, select the type and the language to be implemented. Then click on *Add*.

altus
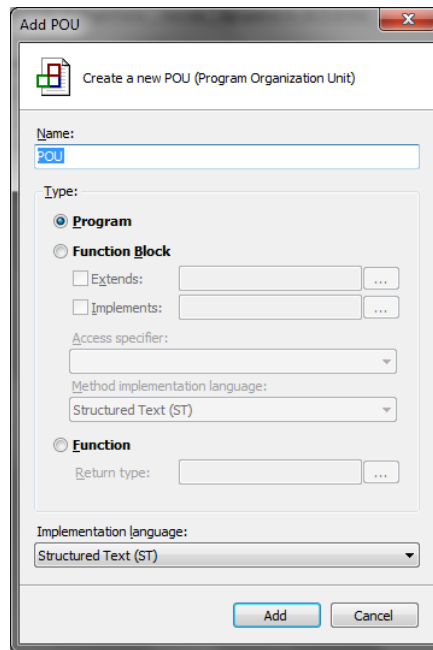
Figure 167: POU Classification

To edit the POU just select the tab with the corresponding name and start developing the application in the language chosen previously. The same procedure is valid for the POUs automatically created by the project profile.
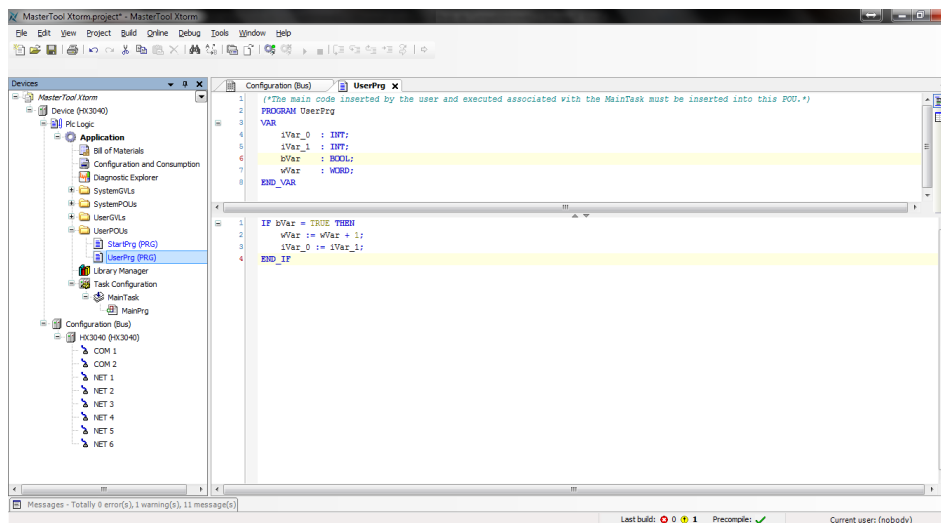
Figure 168: POU Editing

5.6. Creating Tasks

A POU must be related to some task in order to be executed. This scaling mechanism, called *Task*, is very useful for real time systems that define periodic execution or in response to an event (change of state of some Boolean variable). Tasks control the execution of programs in different rates, depending on the characteristics of the application. The need to execute programs at different rates is aimed at the response time requirements of the process under control and to optimize the use of the processing capacity of the CPU. Controllers that use tasks are called multitasking systems.

New tasks are only allowed to be created when the selected project profile is *Customized*, in the other profiles the possible tasks are created and configured automatically.

Thus, to include a new task (if the selected profile allows it), just right-click on the *Task Configuration*, select *Add Object* and then *Task...*, as shown in the figure below.

altus
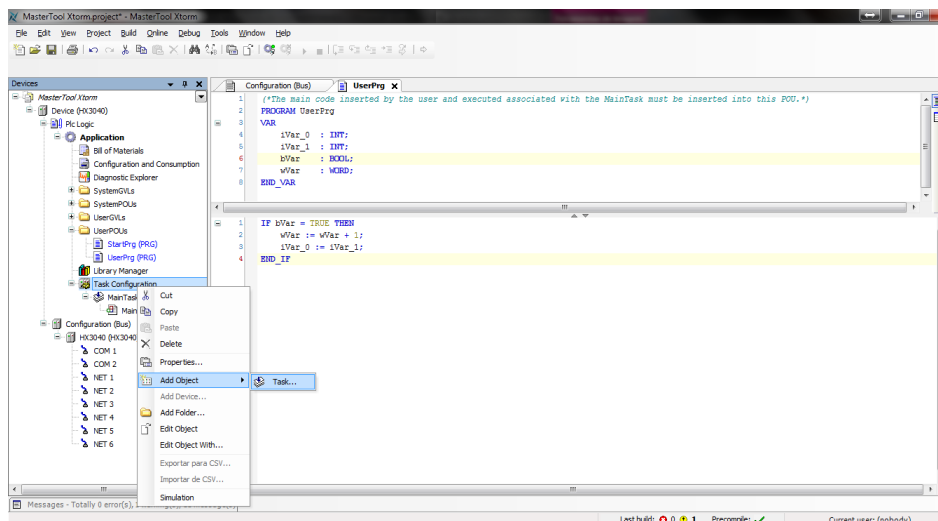
Figure 169: Task Creation

Next, a screen to enter the name of the task will appear. Then click *Add* to end the creation of the task.
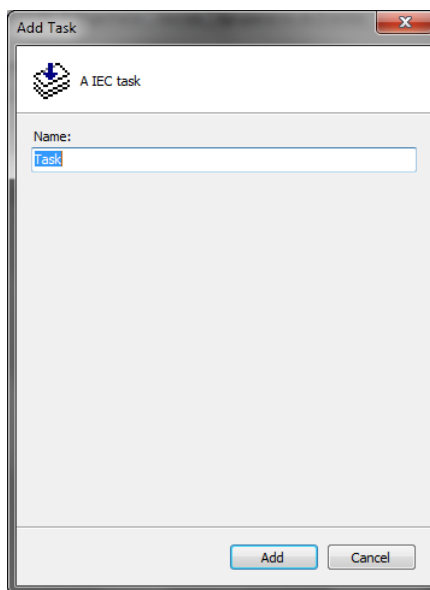
Figure 170: Task Name

5.6.1. Task Configuration

After the task is open, the configuration window will appear for the user to define and classify its functioning.

The field *"Priority (0...31)"* establishes the priority in which the task will be executed in the application, where zero is the highest priority. For instance, the MainTask, created in the majority of the project profiles, has priority 13, so this task is considered with high priority for the system.

The space *"Type"* defines which type and the method used to execute the task, where the following items can be selected:

- *Cyclic*: the task is executed in cycles, or it is called every time interval configured in the field at its side. E.g.: t#100ms.
- *Event*: the task is executed when the variable is from the BOOL type, configured in the field at its side, receives a rising edge, in other words, the variable value goes from FALSE to TRUE.
- *External*: the task is executed when an external event occurs. It is configured in the side field.
- *Freewheeling*: the task is always executed, according to its priority, in other words, tasks with higher priority are executed first.

altus

■ *Status*: the task is executed when the BOOL variable, configured in the side field, is true.

Besides the fields described above, it still has to be set the *Interval* (only for cyclic tasks), that is, the time interval which the task is called to run (maximum time for MainTask is equal to 750 ms and the minimum time is equal to 5 ms). It is recommended to set a task interval in a way that the cycle (execution) time be, at most, 80% of the interval.

The *watchdog* of the CPU is configured to prevent the locking of the user tasks.

The field *"Time"* sets the maximum time allowed for the task execution. If the task takes longer than the watchdog configured time, the application will go to STOP and enter into exception for watchdog.

The field *"Sensitivity"* refers to how many times the watchdog will be achieved to activate the bWatchdogReset diagnostics. If the task Cycle Time reaches the Sensitivity field value multiplied by the Time field, the diagnostics will also be indicated.

Attention should be given to the fact that the watchdog of the CPU is not used to protect the user application from surge at cycle time, but of crashes.

In order to protect the system regarding to possible configuration error, the MasterTool Xtorm checks the watchdog (Software Watchdog) and the minimum and maximum limits of the task cycle time (Interval) in all cyclic tasks, during the compilation. It is important to highlight that the user will have to be careful when changing the pre-defined values by the project profiles as in this way it may put in risk the system execution. So, it is recommended to use the default values, with the watchdog configured at 1000 ms.
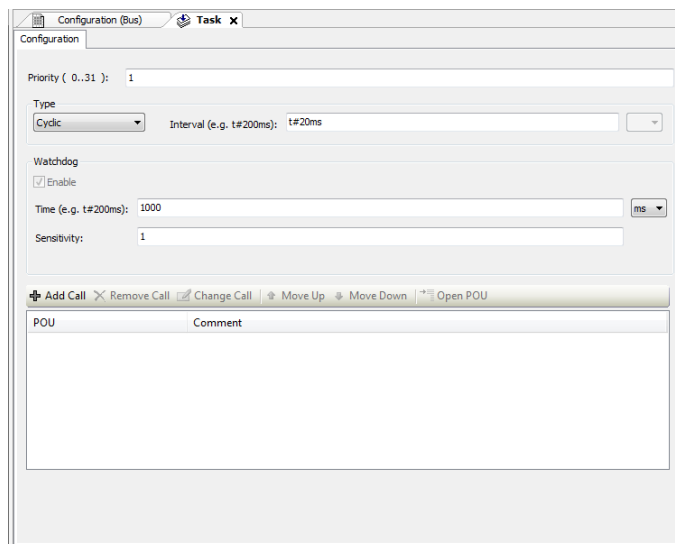
Figure 171: Configuring the Created Task

The table below shows the verifications done by MasterTool Xtorm, with the field *Sensitivity* equal to 1. For the Custom Profile, no consistency is made in the task interval and watchdog time.

Task	Type	Minimum Interval	Maximum Interval
MainTask	Cyclic	5 ms	750 ms
CyclicTask	Cyclic	5 ms	2147483 ms
TimeInterruptTask	Cyclic	500 μs	2147483 ms

Table 215: Maximum Allowed Settings

5.6.2. POU – Task Connection

As described previously, for a POU to be executed in the application, it must be connected to a task. In the project profiles (except Custom), the POUs are already associated to its respective tasks. In case new POUs are created, make sure they are connected to the tasks.

To associate a created POU, just access the desired task by double clicking on it in the device tree, and click on *"Add POU"*. After this, a screen called *"Input Assistant"* will appear, on which the desired POU should be selected, as shown in the figure below.
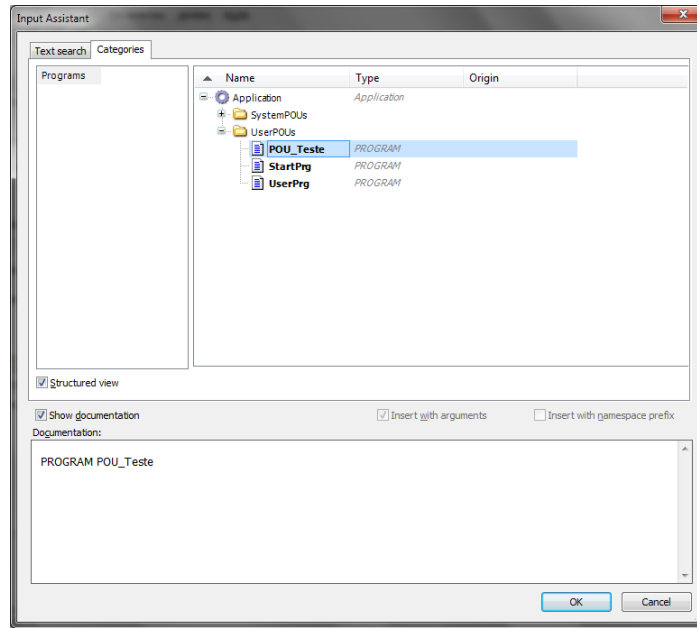
Figure 172: Connecting POUs to Tasks

Another option is to call a POU inside a POU already instantiated in the program, here is an example where the *POU_Test* is called inside the area intended to code of the UserPrg POU:

```
POU_Test();
```

5.6.3. Maximum Number of Tasks

The maximum number of tasks the user can create is only defined for the Custom profile, the only one that has this permission. The others already have their tasks created and configured.

The table below describes the maximum IEC task quantity per CPU and project profile, where the protocol instances are also considered communication tasks by the CPU.

		HX3040	
	Task Type	**UTR**	**Custom**
Configuration Task (Tarefa WHSB)	**Cyclic**	1	1
User Task	**Cyclic**	1	15
	Triggered by Event	0	15
	Triggered by External Event	0	15
	Freewheeling	0	15
	Triggered by State	0	15
NETs – Client or Server Instances	**Cyclic**	4	
COM (n) – Master or Slave Instances	**Cyclic**	1	
TOTAL		16	32

Table 216: Maximum Number of IEC Tasks

Notes:
Values: The number defined for each task type represents the maximum values allowed

altus

WHSB Task: WHSB is a system task that must be considered so the total value is not surpassed. For a redundant RTU profile, the user may create also the ProtTask, thus getting up to 2 cyclic tasks.

NET (n) - Client or Server Instances: "n" represents the Ethernet interface. The maximum value defined by system Ethernet interfaces, including the expansion modules when these are applied. E.g., MODBUS protocol instances.

COM (n) - Master or Slave Instances: "n" represents the serial interface number. Even with expansion modules, the value shown on the table will be the maximum per interface. E.g., MODBUS protocol instances.

Total: The total value does not represent the sum of all profile tasks, but the maximum value allowed per CPU. Therefore, the user can create several task types, while the established numbers for each one and the total value are not surpassed

5.7. CPU Configuration

The configuration of the Hadron Xtorm CPU is based on configuring the parameters of the CPU about the Watchdog, hot swap, time synchronization, internal points, engineering conversion, alarms, and event grouping.

The user must double-click on the Hadron Xtorm CPU, located in the device tree as shown in the figure below, and configure the fields as described in section CPU Configuration.
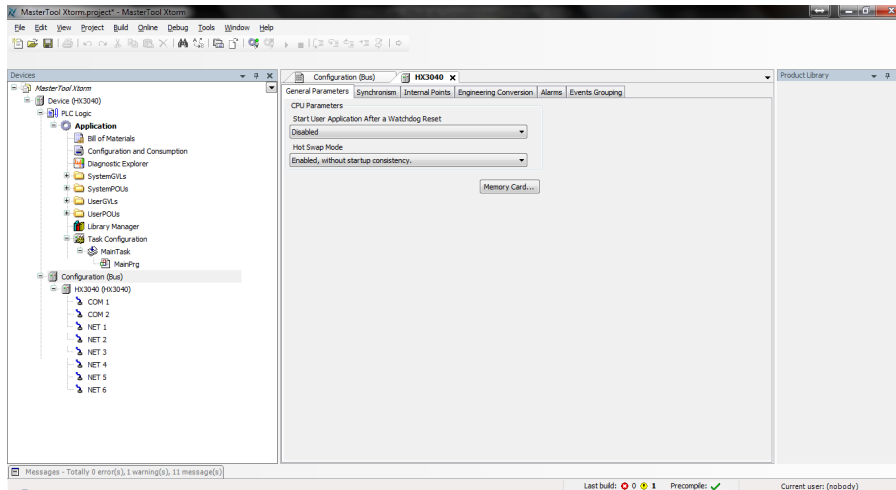
Figure 173: CPU Configuration

In addition, for communication between the CPU and the MasterTool Xtorm to be possible, the Ethernet interface NET 1 must be configured, according to section Ethernet Interfaces Configuration. Double-clicking the NET 1 icon of the CPU in the device tree will open a new tab for configuring the communication network to which the module is connected.
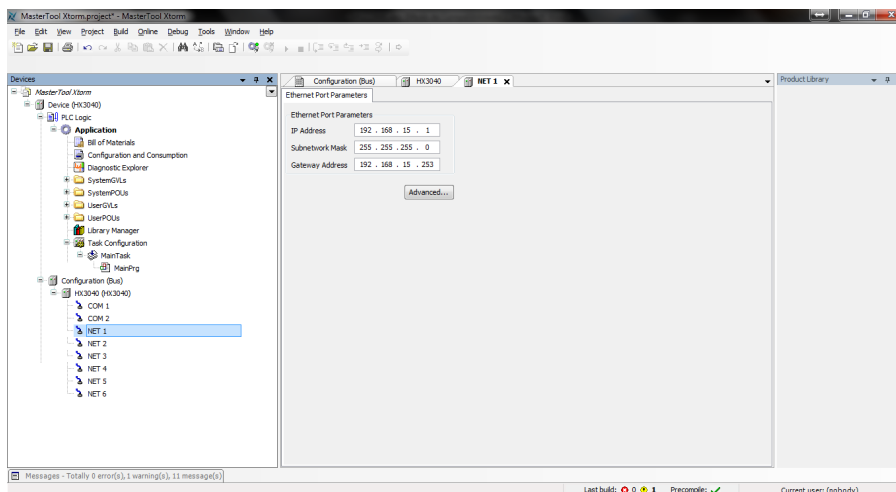
Figure 174: Configuring the CPU Communication Port

If the CPU with the configured IP is not found on the network or the currently active CPU has a different IP than the active one configured in the project, a message will appear on the screen during the Login, requesting the user to change the old IP to the configured one (*Yes*) or *No* to not send the project.
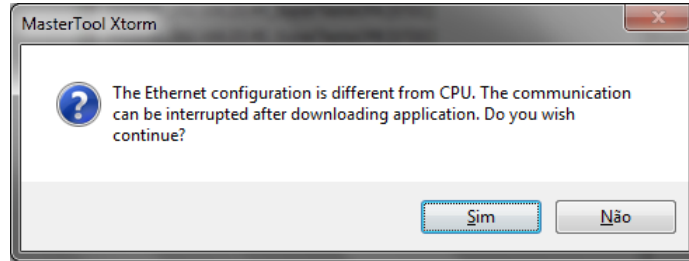
Figure 175: IP Configuration Warning

5.8. Libraries

There are several features of the programming tool that are available through libraries. Therefore, they must be inserted into the project to be able to use them. The insertion procedure is quite simple, it is necessary to select the item *"Library Manager"*, available in the treeview of the menu *Device* and select *"Add Library"*, as shown in the figure below:
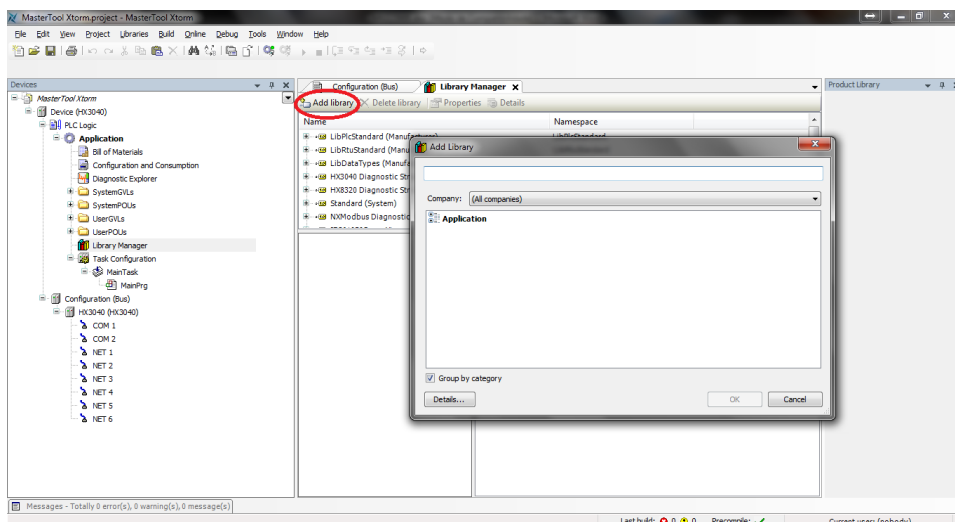
Figure 176: Inclusion of a Library in the Project

Then select the desired library to include in the project, and press the *OK* button.

5.9. Inserting a Protocol Instance

The Hadron Xtorm Series CPU itself, according to the Protocols section, offers protocols, such as MODBUS. Simply add and configure (according to the Protocols Configuration section) the desired protocol instance on the communication interface.

Below are described two cases of MODBUS protocol insertion, one in the serial interface and another in the Ethernet interface, a case of DNP3 Server protocol insertion and a case of IEC 60870-5-104 Server protocol insertion. The insertion of Client protocols is done in a similar way.

5.9.1. MODBUS RTU

The first step to configure MODBUS RTU, in slave mode, is to include the instance in the desired COM (in this case, COM 1). Right-click on the *COM* and select *"Add Device..."*, as shown in the figure below.

altus
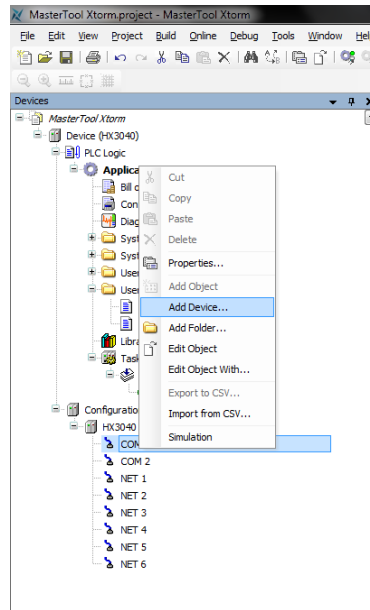
Figure 177: Inserting the MODBUS RTU Instance

Then, the protocols available to the user will appear on the screen. In this case, select the *"MODBUS Slave"* to expand the option tree, select the device and click *Add Device*, as shown in the figure below:
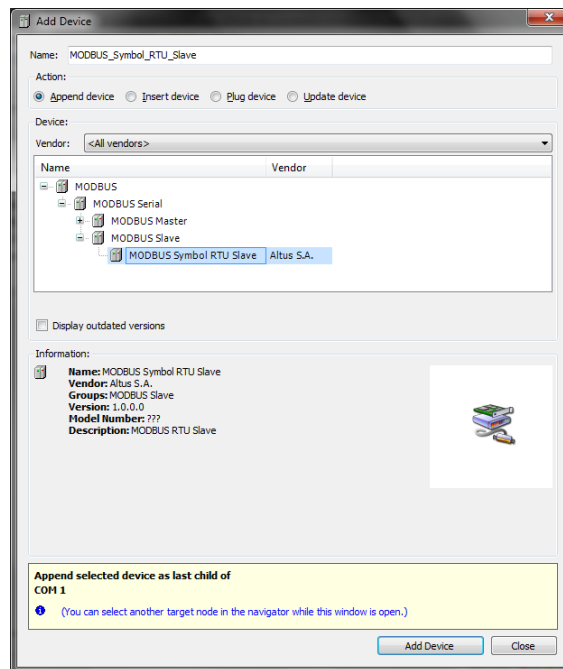
Figure 178: Selecting the MODBUS RTU Protocol

5.9.2. MODBUS Ethernet

The first step to configure MODBUS Ethernet, in client mode, is to include the instance in the desired NET (in this case NET 1, the HX3040 CPU has six Ethernet interfaces). Right-click on the *NET* and select *"Add Device..."*, as shown in the figure below.
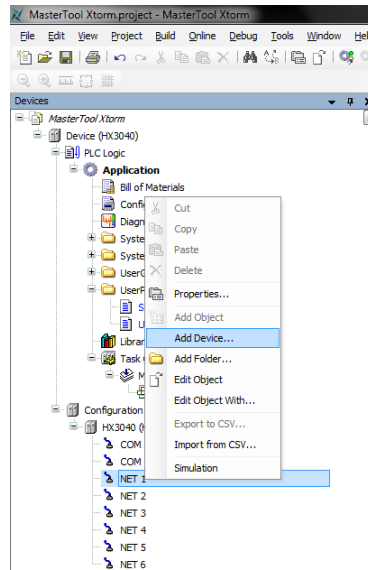
Figure 179: Inserting the MODBUS Ethernet Instance

After that, the protocols available to the user will appear on the screen. In this case, select *"MODBUS > MODBUS Ethernet > MODBUS Client > MODBUS Symbol Client"* and click *Add Device*, as shown in the figure below:
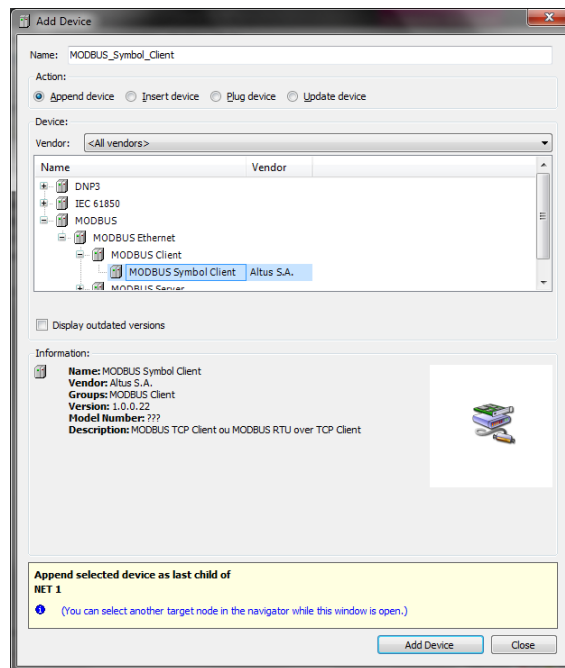
Figure 180: Selecting the MODBUS Ethernet Protocol

5.9.3. DNP3 Server

The first step to configure DNP3, in server mode, is to include the instance in the desired NET (in this case NET 1, the HX3040 CPU has six Ethernet interfaces). Right-click on the NET and select *"Add Device"* as shown in the figure below:

altus
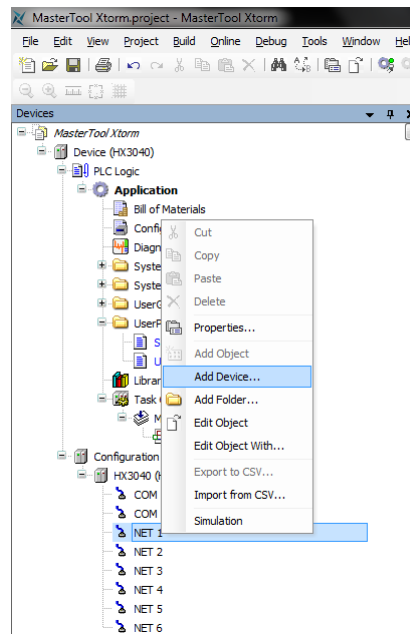
Figure 181: Inserting the DNP3 Server Instance

Then, the protocols available to the user will appear on the screen. In this case, select the *"DNP3 > DNP3Ethernet > DNP3 Server > DNP3 Server"* and click *Add Device*, as shown in the figure below:
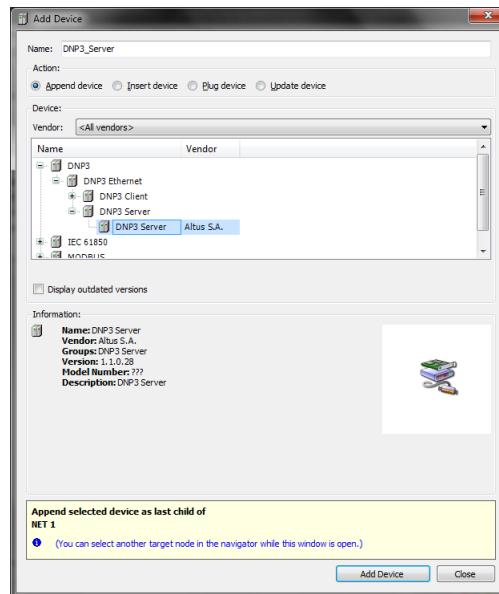
Figure 182: Selecting the DNP3 Server Protocol

5.9.4. IEC 60870-5-104 Server

The first step to configure the IEC 60870-5-104 communication driver, in server mode, is to include the instance in the desired NET (in this case NET 1, the HX3040 CPU has six Ethernet interfaces). Right-click on the *NET* and select *"Add Device..."*, as shown in the figure below:

altus
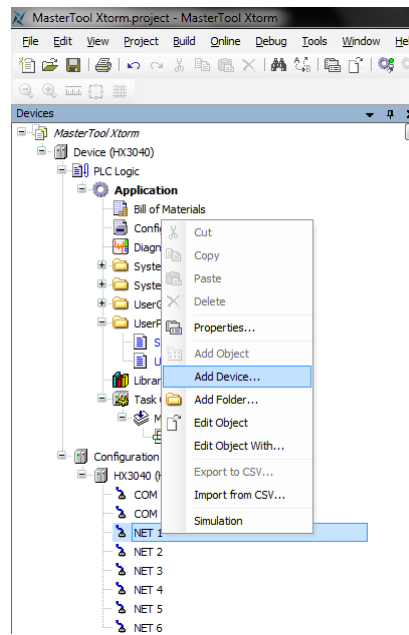
Figure 183: Inserting the IEC 60870-5-104 Server Instance

Then, the protocols available to the user will appear on the screen. In this case, select the *"IEC 60870-5-104 > IEC 60870-5-104 Server > IEC 60870-5-104 Server"* and click *Add Device*, as shown in the figure below:
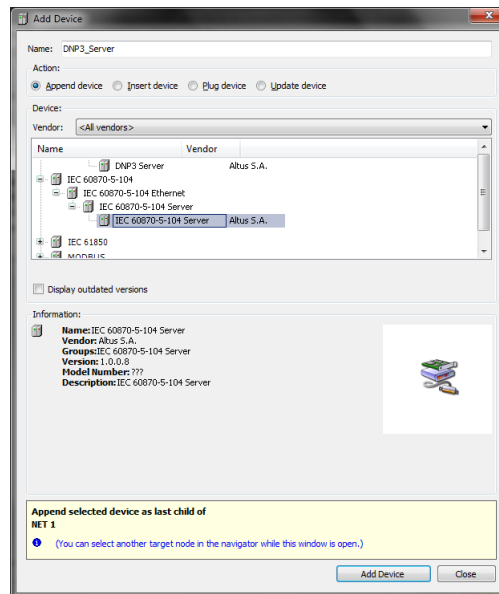
Figure 184: Selecting the IEC 60870-5-104 Server Protocol

5.10.   Finding the Network

As there is the possibility of more than one CPU to be connected to the network, the user must find all communication units and select the desired one.

Initially, double-click "Device" in the device tree and select the Gateway on the *"Communications Settings"* tab. If there is no Gateway or in case you want to add a new one, click *"Add Gateway"* and set its IP. For mapping the network available devices, click *"Scan Network"*.

Then, wait until the MasterTool Xtorm software searches the available network CPUs.
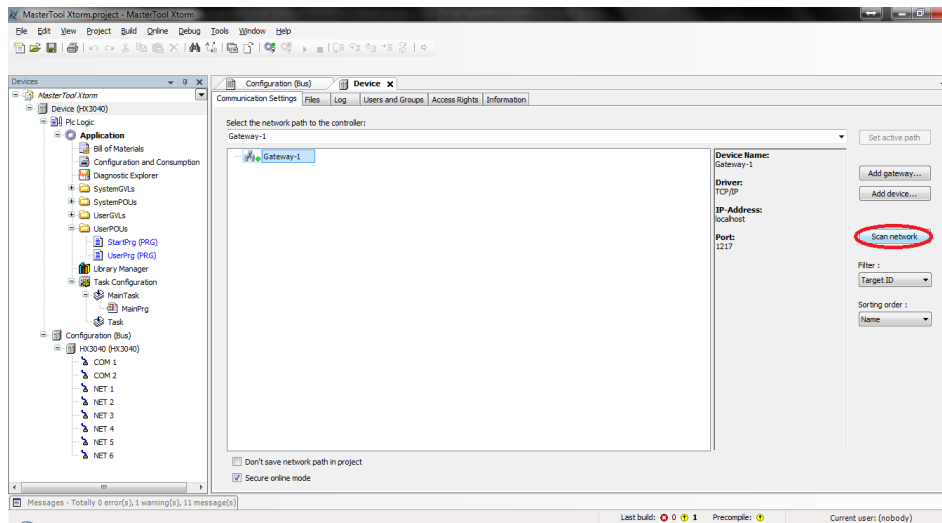
Figure 185: Finding the CPU

Then, select the desired CPU. Click *"Set active path"* in order to activate the CPU and inform the configuration software to which CPU it should communicate with and send the project.
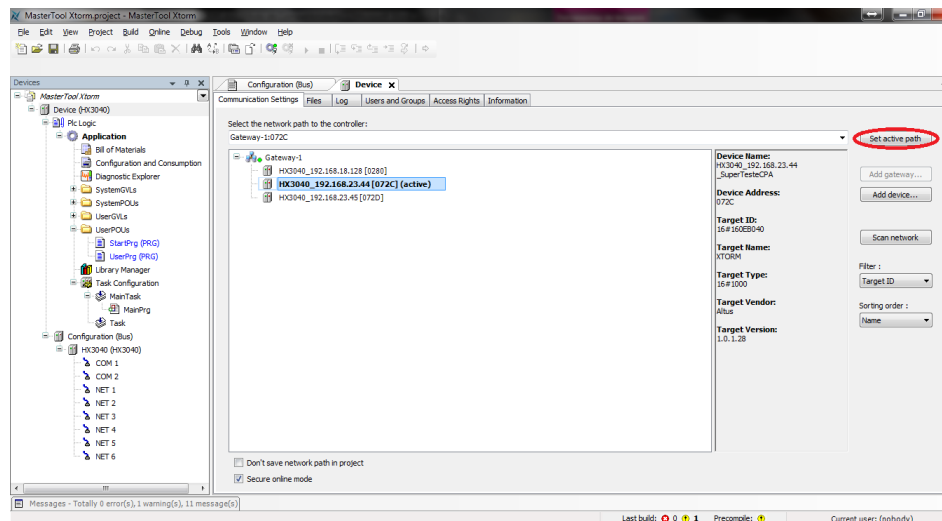
Figure 186: Activating the CPU

If necessary, change the device default name. For that, click with the right mouse button on the desired device and select *"Change Node Name"*. Notice that after a name change, the device will not return to the default name under any circumstances.

5.11. Compiling a Project

In order to verify the created application, the user must compile the project. This is the most effective way to find problems or receive warnings about some mistakes made during product configuration and application editing. To perform this procedure, simply access the *Build* menu and click *"Generate Code"*, as shown in the figure below.

altus
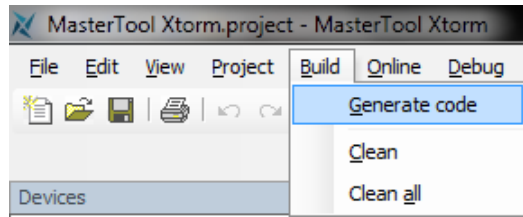
Figure 187: Compiling the Project

After the processing time, which varies according to the size of the user's application, the errors and warning messages, if any, will be displayed below, as shown in the figure below.
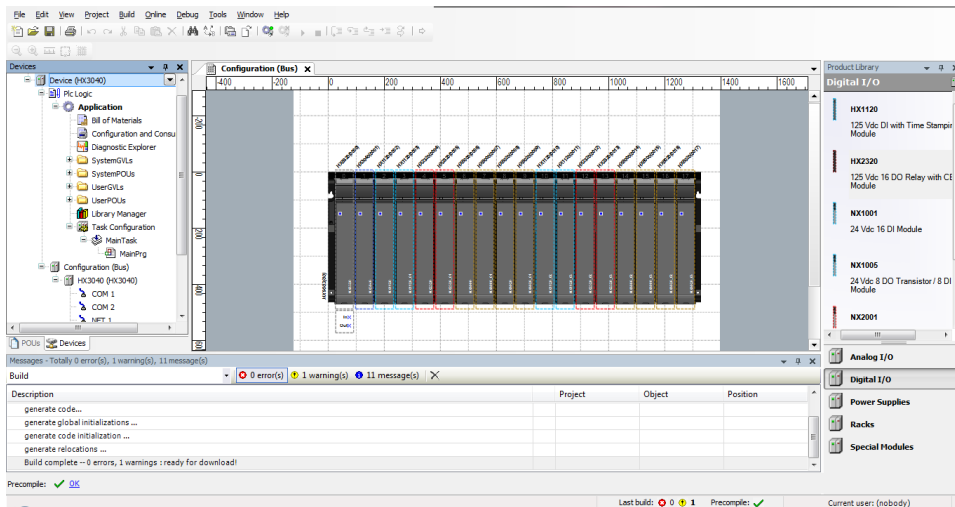
Figure 188: Compilation Messages

Note: If the errors and messages are not visible on the screen, simply access the *View* menu and click *Messages*, as shown in the figure below.
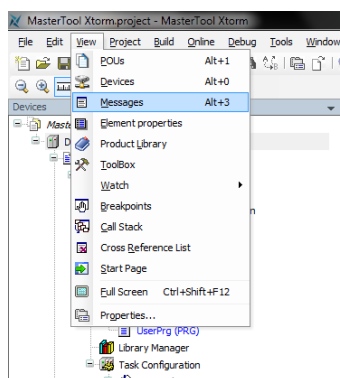
Figure 189: Including the Messages on the Screen

5.12. Login

After compiling the application and fixing all the errors found, it is time to send the project to the CPU. To do this, simply perform the *Login* operation in the MasterTool Xtorm software. This operation may take a few seconds, depending on the size of the generated file. To perform the Login, simply access the *Online* menu and click on the *"Login"* option, as shown in the example of the figure below.

altus
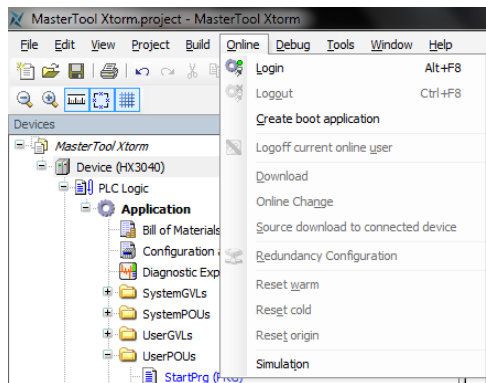
Figure 190: Sending the Project to the CPU

After the command execution, some user interface messages may appear, due to differences between the old project and the new one that is being sent, or simply because there was a variation in some variable.

The figure below shows the message that MasterTool Xtorm will present in case the new project is different from the project already existent inside the CPU. The available options are the following:

- *Login with Online Change:* Execute the Login and send the new project without the current CPU application being stopped (see section Run Mode), updating the changes when a new cycle is executed.
- *Login with Download:* Execute the Login and send the new project with the CPU stopped (see section Stop Mode). When the application is started, the updates will have already been done.
- *Login Without Any Change:* Executes the login without sending the new project.
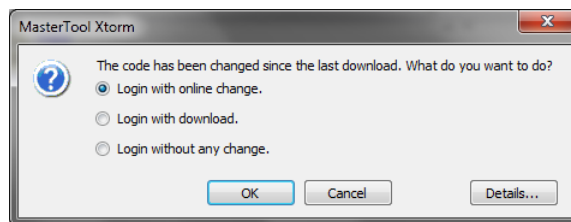
Figure 191: CPU Project Updating

The figure below shows the message displayed by MasterTool Xtorm when changes have been made only in application variables, so that the new project cannot be uploaded and updated in a new cycle of the running CPU (see section Run Mode). Thus, MasterTool Xtorm requests the user whether the Login should be executed as a download and the CPU stopped (see section Stop Mode) or the operation should be cancelled.

Note: The *"Details..."* button shows the changes made to the application.
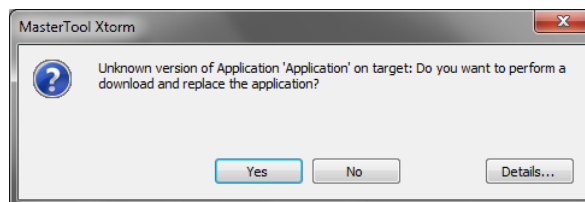
Figure 192: Variable Changes

If this is the first application sent to the CPU, the message in the figure below will appear on the MasterTool Xtorm screen.

altus
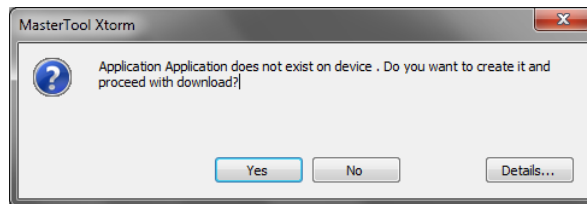
Figure 193: First Application Sending

5.13. Run Mode

Right after the project has been sent to the CPU, the application will not be performed immediately (only if an online change has been made). For that to happen, select the *"Start"* command. This function allows the user to control the execution of the application sent to the CPU. Furthermore, it allows initial values to be pre-configured, so they can be updated on the first cycle in the CPU.

To select such functionality, access *"Debug"* and click *"Start"*, as shown in the figure below.
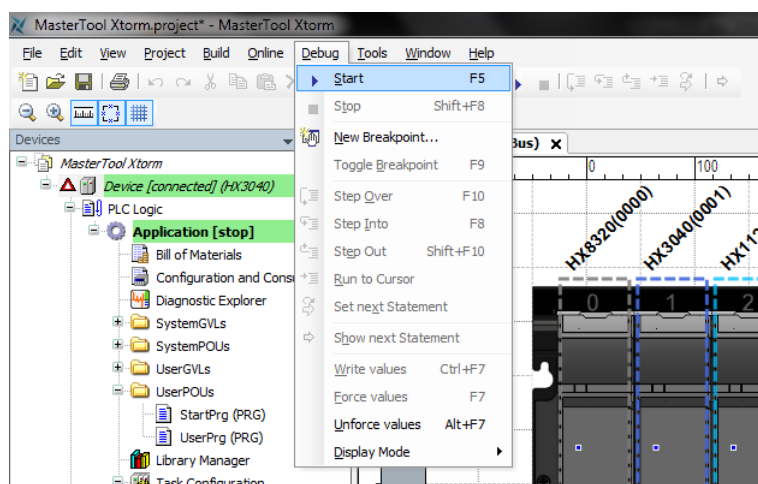
Figure 194: Starting the Application

The figure below shows the application running. In case the POU tab is selected, the created variables are listed on a monitoring window, in which the values can be forced and visualized by the user. In case the variables are forced through the F7 command on the keyboard, the CPU will indicate this condition on the graphic display. For further details, see the section Graphic Display.

It is important to note that when using the MODBUS RTU Slave and MODBUS Ethernet Server protocols and the *"Read Only"* option of the configured relations is not selected, the forced write (F7) command must be performed on the variables available in the monitoring window, because the write command (CTRL + F7) allows the variables to be overwritten when new readings are performed.

altus
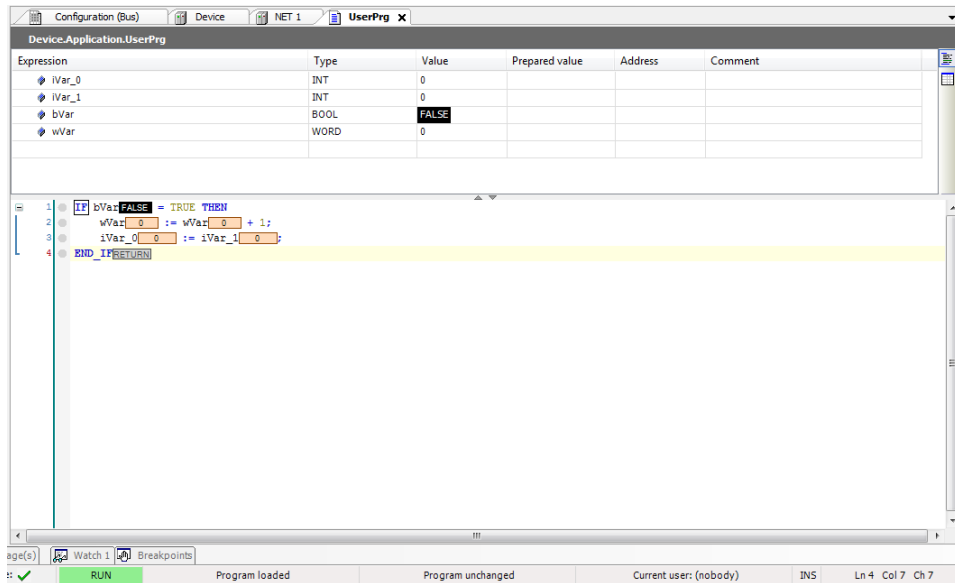
Figure 195: Program in Execution

In case the CPU is initialized with an application already internally stored, it automatically goes to Run Mode, without the need for a MasterTool Xtorm command.

5.14.   Stop Mode

To stop the execution of the CPU, without the MasterTool Xtorm software losing connection with it, the user must select the *"Stop"* option available in the *Debug* menu, as shown in the figure below.
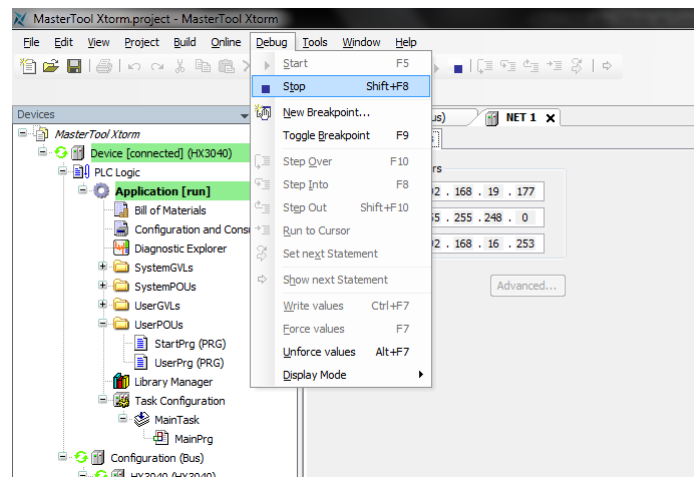
Figure 196: Stopping the Application

In case the CPU is initialized without the stored application, it automatically goes to Stop Mode, as it happens when a software exception occurs.

5.15.   Monitoring, Writing and Forcing Variables

After Logging into a CPU, the user can monitor the project variables in 3 different ways:

- Right in the POU (in the declaration area of the variables or in the program area).

- Through the monitoring window (*View menu -> Watch*).
- Using the *Trace* functionality, which saves and displays variable values internally in the CPU.

> **ATTENTION**
>
> In online mode, there is a limit of 25.000 inputs for monitorable variables in POUs edited with ST Editor. The user will get a compile error message notifying the limit is exceeded.

In addition, the user can also write or force values in project variables.

The writing command (CTRL + F7) writes a value to a variable, and this value can be overwritten by the instructions executed in the application in the next execution cycle.

On the other hand, a forced writing command (F7) will write a value to the variable, without allowing this value to be changed until the forced variables are released.

> **ATTENTION**
>
> The variables forcing can be done in the CPU only in the Online mode.
> Diagnostic variables cannot be forced, only written, because the CPU provides diagnostics that should be able to be overwritten

When a forced writing is done into a redundant variable of the Active CPU, the MainTask execution time will be impacted, in both Active and Stand-by CPUs. This occurs because both CPUs will exchange information about the forced variables in each cycle. Therefore, when forcing variables in a redundant system, the user should consider the time added to the task execution time. The table below exemplifies the medium execution time added to the MainTask in this case:

Execution Time	Active CPU			Stand-by CPU		
	50 ms	**100 ms**	**200 ms**	**50 ms**	**100 ms**	**200 ms**
Increase with 10 forcing	2,4 %	2,2 %	1,7 %	4,0 %	3,4 %	2,0 %
Increase with 50 forcing	12,0 %	9,2 %	6,0 %	18,0 %	12,0 %	8,0 %
Increase with 128 forcing	26,0 %	21,0 %	16,0 %	56,0 %	34,0 %	22,5 %

Table 217: The Influence of the Variables Forcing in a Redundant CPU

For further information about variable forcing in a redundant system, see Redundant Data Synchronization.

> **ATTENTION**
>
> When a CPU presents forced variables and it is de-energized, the variables will lose the forcing in the next initialization.
> The limit of forcing for the Hadron Xtorm CPUs is 128 variables, regardless of the CPU model and its configuration.

5.16. Variables Used in Several Sources

A source is a point of an I/O module or a communication driver that writes to a variable.

When a project is compiled, the Xtorm Mastertool checks all points and reports whether any variables are being used in more than one source. This verification is performed only for the fields in the table below.

altus

Módulo / Driver	Functionality	Field / Source
Validated Fields		
HX3040	Grouping of Events	Variable Name Excessive Delay Alarm
MODBUS device of the MODBUS Client	Mappings	Variable Name Quality
MODBUS device of the MODBUS Master	Mappings	Variable Name Quality
Outstation of the DNP3 Client	Mappings	Variable Name Quality
I/O Module	Bus: I/O Mapping	Variable

Table 218: Variables used in more than one Source

5.17. Logout

If the user chooses to finish the communication with the CPU, the *"Logout"* command, located in the *Online* menu, must be used, as shown in the figure below.
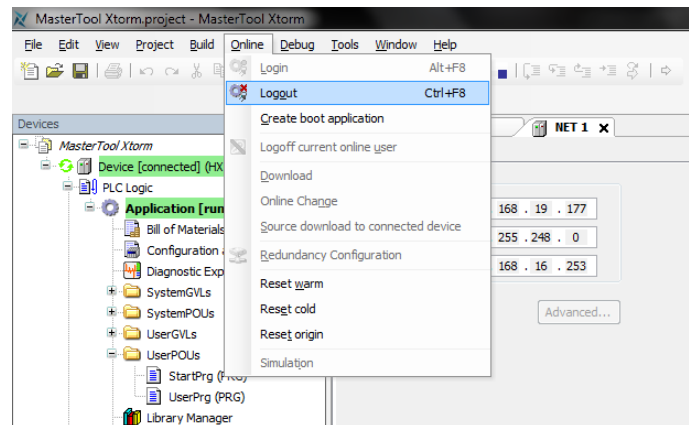
Figure 197: Interrupting the Communication with the CPU

5.18. Simulation Mode

MasterTool IEC XE has an important simulation feature, which allows the user to test its application without the equipment use, which provides a higher flexibility for the program development. However, some specific resources cannot be simulated, depending on the CPUs hardware.

The following resources are unavailable in the simulation mode:

- RTC Clock
- Bus Scan
- I/O Modules
- Serial Ports
- Ethernet communication
- MODBUS communication protocols
- DNP3 communication protocols
- MMS communication protocols
- GOOSE communication protocols
- Operations on memory card

altus

- Diagnostics in variables
- Queueing and grouping events
- Other functions that access the CPU hardware

For this reason, the simulation mode has to be used to test the application logic regardless of the access functions to the hardware. These resources should be tested with the hardware to ensure the application functioning.

In order to switch the MasterTool Xtorm to Simulation Mode it is necessary to select this option in the *Communication Menu* as shown in the figure below. After that a warning is displayed in the bottom bar of MasterTool Xtorm indicating that the tool is operating in Simulation Mode.
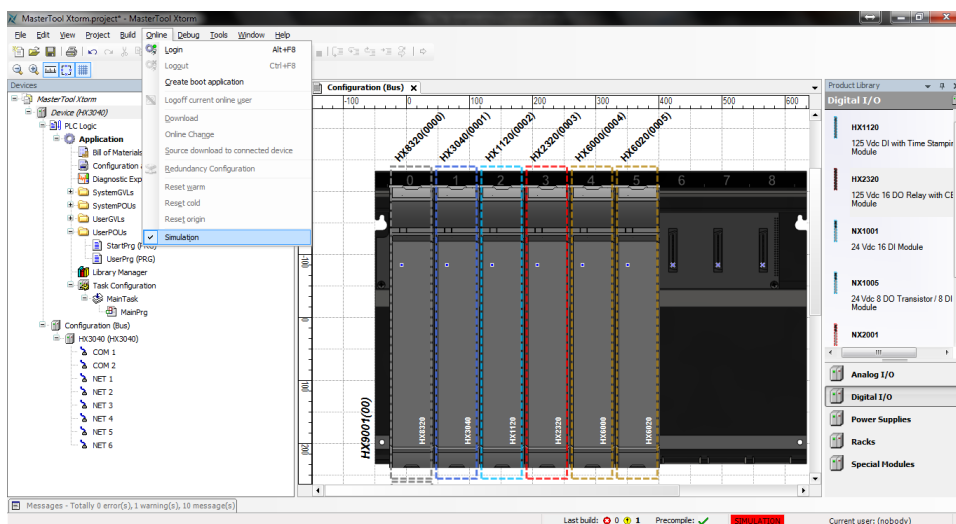
Figure 198: Simulation Mode

In Simulation Mode the application runs on a virtual device on the computer where MasterTool Xtorm is installed. For this reason some characteristics presented are related to the hardware architecture of the computer and not to the Hadron Xtorm Series CPU. The main characteristic in this sense is related to the data format in the direct representation memory areas. The Simulation Mode operates with the little endian format where the first memory address is the least significant of the data. On the other hand the Hadron Xtorm Series CPU operates with the big endian format where the first memory address is the most significant of the data.

In this case the same data written for example in %QD0, will be written differently in the simulation and in the Hadron Xtorm CPU. If the data written is 16#1234ABCD the distribution of the data in the CPU memory will be as follows:

- %QW0 = 16#1234
- %QW2 = 16#ABCD
- %QB0 = 16#12
- %QB1 = 16#34
- %QB2 = 16#AB
- %QB3 = 16#CD

For the same data written in %QD0 in Simulation Mode the distribution of data in memory will be as follows:

- %QW0 = 16#ABCD
- %QW2 = 16#1234
- %QB0 = 16#CD
- %QB1 = 16#AB
- %QB2 = 16#34
- %QB3 = 16#12

In view of these differences and to facilitate application development using the resources of MasterTool Xtorm and the Hadron Xtorm Series CPU it is recommended to use symbolic variables. In this case the differences between Simulation Mode and the behavior with the Hadron Xtorm Series CPU are not verified. Therefore the best practice is to avoid using direct representation variables whenever possible to avoid rework when developing logic that will be tested in simulation and then loaded into a CPU.

Simulation mode can also be used to simulate a redundant design, however, it will have the same limitations as mentioned above, and only logic that does not depend on the hardware can be tested. In this case, the *NonSkippedPrg* and *UserPrg* POUs will always be executed, as if the simulated CPU was the Active CPU.

5.19. Project Upload

Hadron Xtorm Series CPUs allow the project storing in the product memory, which can be uploaded and reused through the MasterTool Xtorm software.

To store a project in the CPU's memory, the CPU must be connected (Login) and the option of sending archive, together with the application, must be selected.

To recover the previously stored project, the options must be selected, as shown in the figure below.
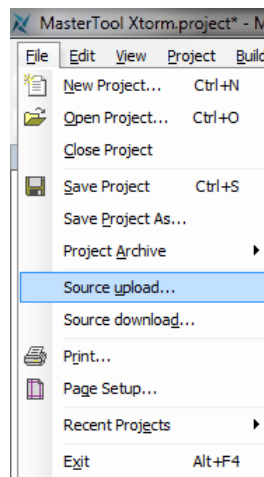
Figure 199: Project Upload Options

Then, select the desired CPU and click *OK*, as shown in the figure below.
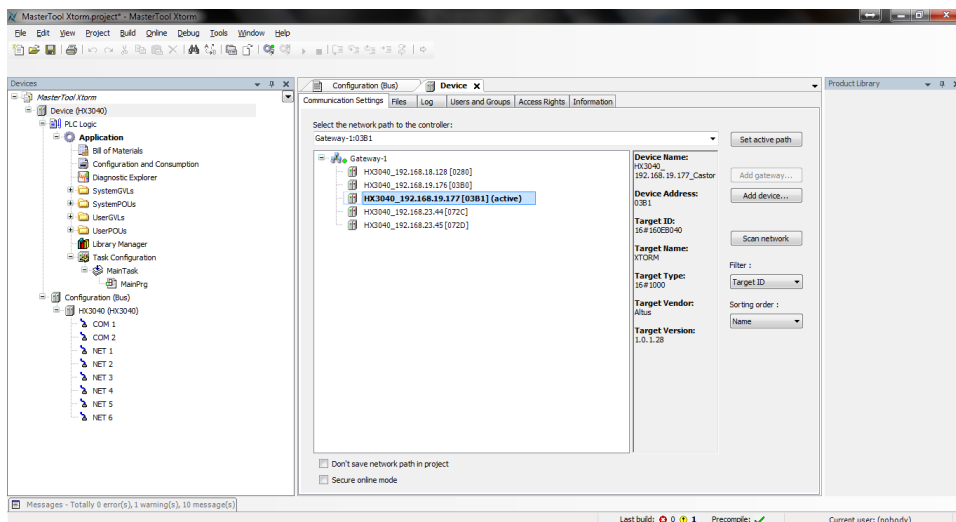
Figure 200: CPU Selection

> **ATTENTION**
>
> The memory size area to store a project in Hadron Xtorm CPUs is defined in Memory section.

> **ATTENTION**
>
> The upload retrieves the last stored project in the controller as described in the former paragraphs. In case the download refers to the execution of only a specific application, it will not be possible to retrieve it through this procedure (Upload).

5.20.   CPU Operating States

5.20.1.   Run

The Run mode indicates that all CPU's application tasks are running.

5.20.2.   Stop

The Stop mode indicates that the application tasks are stopped. The variables of the tasks are kept with their current value and the output variables take specific values, which are defined by the user.

When a CPU switches to Stop mode, all the variables in the application tasks will be lost, except for the persistent ones. The output variables will take the value defined by the user and then their value will switch for the safe state. The output variables will regain the value defined by the user as soon as the new application is loaded.

5.20.3.   Breakpoint

When a debug mark is reached in a task, that task is interrupted. All other active tasks in the application will not be interrupted, but will continue their execution. In this mode it is possible to step through a program in Online mode. A step-by-step can be executed and the positions of the debugging interrupts depend on the editor.

Note:

When in breakpoint mode (BRKP), the bus service is completely stopped. Thus, the CPU that controls the bus presents some specific behaviors, which are described below:

- The hot swap of any module is not supported. The module will have no parameter in case there is of hot swap in Breakpoint.
- The diagnosis of all modules will be kept frozen in the application.
- The diagnostic button of the modules is unavailable, so there is no harm if the user presses it.

5.20.4.   Exception

When a CPU is in Exception it indicates that some improper operation occurred in one of the active tasks of the application. The task that caused the Exception will be suspended and the other tasks switch to Stop mode. The user must set the CPU to a new start condition in order to take the tasks off this state and reset them to Execution mode. In other words, for the application to get back in Run mode it is necessary to restart the CPU or perform a Reset (Warm, Cold, and Origin).

5.20.5.   Reset Warm

This command puts the CPU in Stop mode and starts all the application tasks variables, except the persistent and retentive ones. The variables, which were started with a specific value, will take this exact value, while the other ones will take the standard starting value (zero).

5.20.6.   Reset Cold

This command puts the CPU in Stop mode and starts all the application tasks variables, except the persistent ones. The variables, which were started with a specific value, will take this exact value, while the other ones will take the standard starting value (zero).

5.20.7. Reset Origin

This command removes all the application tasks variables, including the persistent type variables and deletes the CPU application

Notes:

Reset: This command disables the breakpoints that were defined in the application.

Comando: To execute a Warm, Cold or Origin Reset command, the MasterTool must be in Online mode (logged in to the CPU).

5.20.8. Reset Process Command (IEC 60870-5-104)

This reset process command can be requested by the IEC 60870-5-104 clients. After responding to the client, the CPU enters a reboot procedure, like a power on cycle.

In the case of redundant CPUs, the reset process command is synchronized with the Non-Active CPU, resulting in the restart of the two CPUs.

The IEC 60870-5-104 standard provides a qualifier value (0..255) with the reset process command, but this "parameter" is not considered by the CPU.

5.21. Programs (POUs) and Global Variable Lists (GVLs)

The project created by MasterTool Xtorm contains a set of program modules (POUs) and global variables lists (GVLs) that aims to facilitate the programming and utilization of the controller. The following sections describe the main elements that are part of this standard project structure.

5.21.1. MainPrg Program

The MainTask is connected to a single POU (program type), called MainPrg. The MainPrg program is created automatically.

The following code refers to the MainPrg, in ST language:

```
(*Main POU associated with MainTask that calls StartPrg,
UserPrg, NonSkippedPrg and the system POUs AlarmPrg and EngineeringPrg.
This POU is blocked to edit.*)

PROGRAM MainPrg
VAR
   isFirstCycle : BOOL := TRUE;
END_VAR

IF isFirstCycle THEN
   StartPrg();
   isFirstCycle := FALSE;
ELSE
   EngineeringPrg();
   AlarmPrg();
   UserPrg();
END_IF;
```

MainPrg calls other four POUs of program type, named *StartPrg, EngineeringPrg, AlarmPrg* and *UserPrg*. While *EngineeringPrg, AlarmPrg* and *UserPrg* are always called, the *StartPrg* is only called once in the PLC application start.

Different from programs *EngineeringPrg* and *AlarmPrg*, which, like *MainPrg*, are blocked for modification, programs *StartPrg* and *UserPrg* can be modified by the user. Initially, when the project is created through the *Wizard*, these two editable programs are created "empty", but the user will be able to insert code in them.

altus

5.21.2. StartPrg Program

In this POU the user might create logics, loops, start variables, etc. That will be executed only one time in the first PLC's cycle, before execute *UserPrg* POU by the first time. And not being called again during the project execution.

In case the user load a new application, or if the PLC gets powered off, as well as in *Reset Origin, Reset Cold* and *Reset Warm* conditions, this POU is going to be executed again.

5.21.3. EngineeringPrg Program

This POU is blocked for editing and is generated automatically by the program, it maps the variables used for engineering conversions, characteristic of the Engineering Conversion CPU, described in section Engineering Conversion.

5.21.4. Programa AlarmPrg

This POU is also blocked for editing. It automatically generates the code responsible for alarm assignment. This alarm can be created in the settings of the Alarms tab of the CPU. For more information, see section Alarms.

5.21.5. Programa UserPrg

In this POU the user must create the main application, responsible for controlling its process. This POU is called by the main POU (MainPrg).

The user can also create additional POUs (program, functions or functional block), and call or instantiate them inside UserPrg POU, for the purpose of structuring your program. It is also possible to call functions and instantiate functional blocks defined in libraries.

5.21.6. Programa ProtPrg

The ProtTask is associated with a single POU of type program, called ProtPrg. The ProtPrg program is created only if the user chooses to use it when creating the project.

This program is blocked for editing and is responsible for calling the UserProtPrg POU.

5.21.7. Programa UserProtPrg

The UserProtPrg program is intended to control processes with a higher priority than UserPrg. That is, in this POU the user must insert its process logics that require a quick action of the CPU.

This program is called by ProtPrg, which, in turn, is associated with the ProtTask task.

For more information about the ProtTask, see section ProtTask Configurations.

5.21.8. GVL Disables

The *Disables* GVL contains the MODBUS Master/Client by symbolic mapping requisition disabling variables. It is not mandatory, but it is recommended to use the automatic generation of these variables, which is done by clicking on the button *Generate Disabling Variables* in the device requisition tab. These variables are declared as type BOOL and follow the following structure:

Requisition disabling variables declaration:

```
[Device Name]_DISABLE_[Requisition Number] : BOOL;
```

Where:

Device name: Name that shows on Tree View to the MODBUS device.

Requisition Number: Requisition number that was declared on the MODBUS device requisition table following the sequence from up to down, starting on 0001.

Example:

```
Disables
VAR_GLOBAL
  MODBUS_Device_DISABLE_0001   : BOOL;
  MODBUS_Device_DISABLE_0002   : BOOL;
  MODBUS_Device_DISABLE_0003   : BOOL;
  MODBUS_Device_1_DISABLE_0001  : BOOL;
  MODBUS_Device_1_DISABLE_0002  : BOOL;
END_VAR
```

The automatic generation through the button *Generate Disabling Variables* only create variables, and don't remove automatically. This way, in case any relation is removed, its respective disabling variable must be removed manually.

The *Disables* GVL is editable, therefore the requisition disabling variables can be created manually without need of following the model created by the automatic declaration and can be used both ways at same time, but must always be of BOOL type. And it is need to take care to do not delete or change the automatic declared variables, cause them can being used for some MODBUS device. If the variable be deleted or changed then an error is going to be generated while the project is being compiled. To correct the automatically declared variable name, it must be followed the model exemplified above according to the device and the requisition to which they belong.

The following picture shows an example of the presentation of this GVL when in *Online* mode. If the variable values are TRUE it means that the requisition to which the variables belong is disabled and the opposite is valid when the variable value is FALSE.

Expressão	Tipo	Valor	Valor Preparado	Comentário
MODBUS_Device_DISABLE_0001	BOOL	TRUE		
MODBUS_Device_DISABLE_0002	BOOL	FALSE		
MODBUS_Device_DISABLE_0003	BOOL	TRUE		
MODBUS_Device_1_DISABLE_0001	BOOL	TRUE		
MODBUS_Device_1_DISABLE_0002	BOOL	FALSE		

Figure 201: GVL Disable in Online Mode

5.21.9. GVL IOQualities

The *IOQualities* GVL contains the quality variables of I/O modules declared on the CPU's bus. This GVL is not editable and the variables are automatically declared as *LibDataTypes.QUALITY* type arrays, and dimensions according to I/Os quantities of the module to which it belongs when that is added to the project.

Example:

```
VAR_GLOBAL
  QUALITY_HX1120: ARRAY[0..31]  OF LibDataTypes.QUALITY;
  QUALITY_HX2320: ARRAY[0..15]  OF LibDataTypes.QUALITY;
  QUALITY_HX6000: ARRAY[0..15]  OF LibDataTypes.QUALITY;
  QUALITY_HX6020: ARRAY[0..7] OF LibDataTypes.QUALITY;
END_VAR
```

Once the application is in *RUN* it is possible to watch the I/O modules quality variables values that were added to the project through *IOQualities GVL*.

5.21.10. GVL Module_Diagnostics

The *Module_Diagnostics* GVL declares the diagnostic variables of the modules used in the project, except for the CPU and communication drivers. This GVL is not editable, so the variables are automatically declared with the type specified by the module to which it belongs, when it is added to the project.

Example:

```
Module_Diagnostics
VAR_GLOBAL
  DG_HX1120 : T_DIAG_HX1120_1;
  DG_HX2320 : T_DIAG_HX2320_1;
  DG_HX6000 : T_DIAG_HX6000_1;
  DG_HX6020 : T_DIAG_HX6020_1;
  DG_HX8320 : T_DIAG_HX8320_1;
END_VAR
```

By sending an application to the HX3040 CPU and passing it to *RUN*. Through the GVL *Module_Diagnostics* it is possible to monitor the diagnostics of each module that has been added to the project, as can be seen in the figure below.

Figure 202: GVL Modules_Diagnostic in Online Mode

5.21.11. GVL Qualities

In the *Qualities* GVL the quality variables of the internal variable mappings, MODBUS Master/Client and DNP3 Client are declared. It is not mandatory, but it is recommended to use the automatic generation of these variables, which is done by clicking the *Generate Quality Variables* button in the device mappings tab. These variables are declared as *LibDataTypes.QUALITY* and follow the following structure:

Quality mapping variable declaration:

```
[Device Name]_QUALITY_[Mapping Number]: LibDataTypes.QUALITY;
```

Where:

Device Name: Name that appear at the Tree View to the device.

Mapping Number: Number of the mapping that was declared on the device mapping table, following the up to down sequence, starting with 0001.

Example:

```
Qualities
VAR_GLOBAL
  MODBUS_Device_QUALITY_0001: LibDataTypes.QUALITY;
  MODBUS_Device_QUALITY_0002: LibDataTypes.QUALITY;
  MODBUS_Device_QUALITY_0003: LibDataTypes.QUALITY;
  Outstation_QUALITY_0001: LibDataTypes.QUALITY;
END_VAR
```

The GVL *Quality* is editable, so the quality variables of the mappings can be created manually without having to follow the template created by the automatic declaration and can be used both ways at the same time, but they must always be of type *LibDataTypes.QUALITY* and care must be taken to not delete or change the automatically declared variables, because they may be being used by some device. If the variable is deleted or changed an error will be generated when building the project. To correct the name of an automatically declared variable, you must follow the model exemplified above according to the device and mapping to which it belongs.

To the MODBUS communication devices the quality variables behave on the way showed at table 50.

ATTENTION

If a variable of the DNP3 Client or MODBUS Master/Client symbolic mapping drivers is mapped to the DNP3 Server or IEC 60870-5-104 Server drivers, the quality variables of the DNP3 or MODBUS mappings must have been created for valid quality events to be generated for such points of the DNP3 or IEC 60870-5-104 servers. Otherwise, "bad" quality events will not be generated for the clients of the DNP3 and IEC 60870-5-104 servers in situations that the DNP3 Client or MODBUS Master/Client cannot communicate with their slaves/servers, for example.

By sending an application to the HX3040 CPU and putting it in *RUN*. Through GVL *Qualities* you can monitor the diagnostics variable values of the MODBUS Mater/Client and DNP3 Client devices mappings, as shown in the figure below.

Expression	Type	Value
⊟ MODBUS_Device_QUALITY_0001	Rtu_Standard.QUALITY	
◈ VALIDITY	QUALITY_VALIDITY	VALIDITY_INVALID
⊟ ◈ FLAGS	QUALITY_FLAGS	
◈ FLAG_OUT_OF_RANGE	BOOL	FALSE
◈ FLAG_INACCURATE	BOOL	FALSE
◈ FLAG_OLD_DATA	BOOL	FALSE
◈ FLAG_FAILURE	BOOL	FALSE
◈ FLAG_OPERATOR_BLOCKED	BOOL	FALSE
◈ FLAG_TEST	BOOL	FALSE
◈ FLAG_RESERVED_0	BOOL	FALSE
◈ FLAG_RESERVED_1	BOOL	FALSE
◈ FLAG_RESTART	BOOL	TRUE
◈ FLAG_COMM_FAIL	BOOL	FALSE
◈ FLAG_REMOTE_SUBSTITUTED	BOOL	FALSE
◈ FLAG_LOCAL_SUBSTITUTED	BOOL	FALSE
◈ FLAG_FILTER	BOOL	FALSE
◈ FLAG_OVERFLOW	BOOL	FALSE
◈ FLAG_REFERENCE_ERROR	BOOL	FALSE
◈ FLAG_INCONSISTENT	BOOL	FALSE
⊟ MODBUS_Device_QUALITY_0002	Rtu_Standard.QUALITY	
◈ VALIDITY	QUALITY_VALIDITY	VALIDITY_INVALID
⊞ ◈ FLAGS	QUALITY_FLAGS	
⊟ MODBUS_Device_QUALITY_0003	Rtu_Standard.QUALITY	
◈ VALIDITY	QUALITY_VALIDITY	VALIDITY_INVALID
⊞ ◈ FLAGS	QUALITY_FLAGS	
⊟ Outstation_QUALITY_0001	Rtu_Standard.QUALITY	
◈ VALIDITY	QUALITY_VALIDITY	VALIDITY_QUESTIONABLE
⊞ ◈ FLAGS	QUALITY_FLAGS	

Figure 203: GVL Qualities in Online Mode

5.21.12. GVL ReqDiagnostics

The *ReqDiagnostics* GVL contains the requisition diagnostics variables of symbolic mapping MODBUS Master/Client and DNP3 Client. It is not mandatory, but recommended the use of these variables' automatic generation, what is done by clicking in the button *Generate Diagnostics Variables* in device requests tab. These variables declaration follow the following structure:

Requisition diagnostic variable declaration:

```
[Device Name]_REQDG_[Mapping Number]: [Variable Type];
```

Where:

Device Name: Name that appear at the Tree View to the device.

Mapping Number: Number of the mapping that was declared on the device mapping table, following the up to down sequence, starting with 0001.

Variable Type: *"T_DIAG_MODBUS_RTU_MAPPING_1"* to MODBUS Master, *"T_DIAG_MODBUS_ETH_ MAPPING_1"* to MODBUS Client and *"T_DIAG_DNP_CLIENT_REQUEST_1"* to DNP3 Client.

Example:

```
ReqDiagnostics
VAR_GLOBAL
  MODBUS_Device_REQDG_0001  : T_DIAG_MODBUS_RTU_MAPPING_1;
  MODBUS_Device_REQDG_0002  : T_DIAG_MODBUS_RTU_MAPPING_1;
  MODBUS_Device_REQDG_0003  : T_DIAG_MODBUS_RTU_MAPPING_1;
  MODBUS_Device_1_REQDG_0001  : T_DIAG_MODBUS_ETH_MAPPING_1;
  MODBUS_Device_1_REQDG_0002  : T_DIAG_MODBUS_ETH_MAPPING_1;
  Outstation_REQDG_0001   : T_DIAG_DNP_CLIENT_REQUEST_1;
END_VAR
```

The *ReqDiagnostics* GVL is editable, so the diagnostic variables of the requests can be created manually and do not need to follow the model created by the automatic declaration. Actually, you can use both ways at once, but make sure to use variables related to the device, and ensure not to delete or change the variables automatically declared, as they might be in use by a MODBUS device. If you delete or change some variable, the system generates an error during the project compilation. To correct the name of a variable declared automatically, follow the model exemplified above according to the device and requests to which it belongs.

By sending an application to the HX3040 CPU and putting it in *RUN*. Through the *ReqDiagnostics* GVL you can monitor the diagnostics variable values of the MODBUS Master/Client and DNP3 Client devices mappings, as shown it the figure below.

Figure 204: GVL ReqDiagnostics in Online Mode

5.21.13.   GVL System_Diagnostics

The *System_Diagnostics* GVL declares the diagnostic variables of the CPU and communication drivers. This GVL is not editable, so the variables are automatically declared with the type specified by the device to which it belongs, when it is added to the project.

Example:

```
VAR_GLOBAL
   DG_HX3040          : T_DIAG_HX3040_1;
   DG_DNP3_Client          : T_DIAG_DNP_CLIENT_1;
   DG_MODBUS_Symbol_Client   : T_DIAG_MODBUS_ETH_CLIENT_1;
   DG_MODBUS_Symbol_RTU_Master : T_DIAG_MODBUS_RTU_MASTER_1;
END_VAR
```

By sending an application to the HX3040 CPU and putting it in *RUN*. Through GVL *System_Diagnostics* you can monitor the diagnostics variable values of the CPU and communication devices MODBUS and DNP3 Client as well, as shown in the figure below:

Expression	Type	Value
⊟ 🌐 DG_HX3040	T_DIAG_HX3040_1	
⊞ ◈ tSummarized	T_DIAG_SUMMARIZED_1	
⊞ ◈ tDetailed	T_DIAG_DETAILED_1	
⊞ 🌐 DG_DNP3_Client	T_DIAG_DNP_CLIENT_1	
⊟ 🌐 DG_MODBUS_Symbol_Client	T_DIAG_MODBUS_ETH_CLIENT_1	
⊞ ◈ tDiag	T_DIAG_MODBUS_DIAGNOSTICS	
◈ byDiag_1_reserved	BYTE	0
⊟ ◈ tCommand	T_DIAG_MODBUS_COMMANDS	
◈ bStop	BOOL	FALSE
◈ bRestart	BOOL	FALSE
◈ bResetCounter	BOOL	FALSE
◈ bDiag_19_reserved	BOOL	FALSE
◈ bDiag_20_reserved	BOOL	FALSE
◈ bDiag_21_reserved	BOOL	FALSE
◈ bDiag_22_reserved	BOOL	FALSE
◈ bDiag_23_reserved	BOOL	FALSE
◈ byDiag_3_reserved	BYTE	0
⊞ ◈ tStat	T_DIAG_MODBUS_ETH_CLIENT_STATS	
⊟ 🌐 DG_MODBUS_Symbol_RTU_Master	T_DIAG_MODBUS_RTU_MASTER_1	
⊞ ◈ tDiag	T_DIAG_MODBUS_DIAGNOSTICS	
◈ eErrorCode	SERIAL_STATUS	NO_ERROR
⊞ ◈ tCommand	T_DIAG_MODBUS_COMMANDS	
◈ byDiag_03_reserved	BYTE	0
⊟ ◈ tStat	T_DIAG_MODBUS_RTU_MASTER_STATS	
◈ wTXRequsts	WORD	0
◈ wRXNormalResponses	WORD	0
◈ wRXExceptionResponses	WORD	0
◈ wRXIllegalResponses	WORD	0
◈ wRXOverrunErrors	WORD	0
◈ wRXIncompleteFrames	WORD	0
◈ wCTSTimeOutErrors	WORD	0
◈ wDiag_18_reserved	WORD	0

Figure 205: GVL System_Diagnostics in Online Mode

6. HX3040 Redundancy

6.1. Introduction

This chapter describes the Hadron Xtorm Series CPUs redundancy. These CPUs work with hotstandby, so the controllers are doubled. One controller is usually in Active state and in control of the process, while the other, despite of being in Stand-by, and keeps the synchronism with the active controller. This process occurs through a synchronism interface, which is specifically designed for this function, and it is placed on the rack. This is a redundant interface, which presents two independent channels (A and B). In case of a failure in the active controller damaging its process control, the stand-by controller switches automatically to Active, within a very short time, in order not to disturb the process and cause any discontinuities in its outputs.

The *hot-standby* redundancy is a method used to increase failure tolerance and, consequently, increase the availability of automation systems. The basic idea is to ensure that no simple failure in duplicated components causes an interruption in the process control.

The hot-standby redundancy is applied on:

- Energy Generation Plants
- Substations

Each CPU can have one or more network protocols configured to provide a communication with the control center. In case of a failure in the Active CPU, the Standby takes over the connection control. If the networks are duplicated the availability is even higher.

The Hadron Xtorm Series CPUs hot-standby redundancy is not applied to I/O modules. In case the user whishes the I/O module redundancy, one can handle it in the application level. As an example, the user can duplicate or even triplicate an analog input module and create a vote scheme to define which input will be regarded at a given point in the application.

The figure below shows a typical example of redundant architecture using the HX3040 CPU:

1. Ethernet network topology
2. Both CPUs present the same configuration
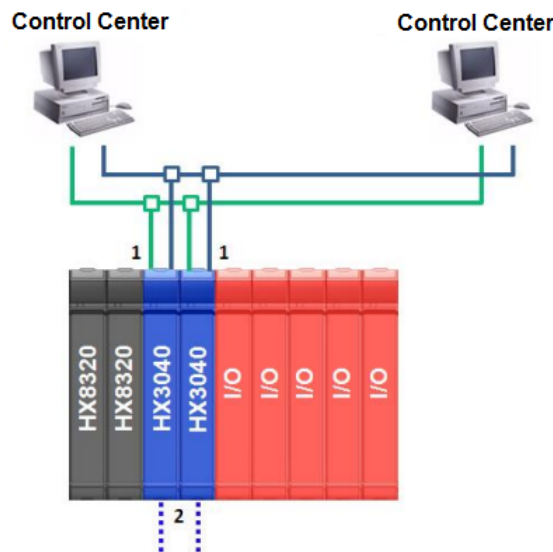
Figure 206: Example of Redundant Architecture with HX3040

6.2. Configurations of a Redundant CPU

To configure the CPU to the redundant mode, perform the following steps:

1. Create a project by selecting the option to use redundant CPUs. To do this, see the section Wizard for a New Redundant Project Creation;
2. Configure the Ethernet interfaces. To do this, see the section Ethernet Interfaces Configuration;
3. Optionally, instantiate DNP and MODBUS protocols. The settings for each protocol can be found in their respective sections.

The descriptions of each configuration are described in the following sections of this chapter.

altus

6.2.1. Identification of a HX3040 CPU

Throughout this manual, the pair of redundant CPUs will be stated as CPUA and CPUB. The HX3040 CPU, which contains a project with CPU redundancy, is stated as:

- UCPA: the CPU inserted in position 2 of the bus.
- UCPB: the CPU inserted in position 3 of the bus.

6.2.2. General Characteristics of a Redundant CPU

	Redundant CPU General Features
Allowed CPUs	HX3040
Redundancy types	Hot-standby
Failure tolerances	The CPU supports, at least, simple failures in doubled equipment in the racks. In specific cases, it can support multiple failures
5 redundancy states	- Not-configured: initial state, also considered when the CPU is off or is not executing the MainTask. - Starting: temporary status obtained after the Not-configured state, where some tests will define the next situation (Inactive, Active, Stand-by or back to Notconfigured). - Inactive: state reached after some given types of failures or programming maintenance. - Active: controls the user process. - Stand-by: ready to switch to Active and control the user process, in case there is such demand (e.g. Active CPU failure).
Main failures that cause switch-over between Active and Stand-by CPU. The Stand-by CPU switches to Active and the Active CPU can switch to Inactive or Not-Configured	- Supplying failure. - CPU (stop in the MainTask execution). - Failure of both synchronism channels among the CPUs
Commands that cause switch-over between the Active CPU and the Stand-by CPU	- Commands via CPUs display - Commands received from MasterTool Xtorm or a SCADA system, through this RTU (local) or the other RTU (remote). - Commands generated by the user application, e.g. as a function of other diagnostics (e.g. Ethernet communication failure), through this RTU (local) or the other RTU (remote).
Main failures that prevent a CPU from remaining in the Stand-by state, or assuming the Stand-by state. Such failures cause this CPU to assume the Not-Configured or Inactive states	- Supplying failure. - CPU (stop in the MainTask execution). - Failure in one of synchronism channels (A or B). - Failure in the synchronism service for redundancy data. - Different project than the one from the Active CPU. - Firmware version incompatible with the Active CPU.
Commands that drive the CPU out of the Stand-by state	- Commands via redundant CPUs display - Commands received from MasterTool Xtorm or a SCADA system, through this RTU (local) or the other RTU (remote). - Commands generated by the user application, e.g. as a function of other diagnostics (e.g. Ethernet communication failure), through this RTU (local) or the other RTU (remote).
Switchover time	- Typically one to three MainTask cycles, depending on the stimulus for state change (command or failure).

altus

	Redundant CPU General Features
No discontinuities switchover (bump-less)	- A switch-over does not cause discontinuities in the controller outputs, nor in internal variables.
Redundancy overhead (MainTask cycle CPU consuming increased by redundancy).	- Maximum value automatically calculated by the MasterTool and informed to the user, considering an empty redundant forcing list. - Typical average value of 15 ms for 100 Kbytes of redundant data and a queue of 4500 events.
CPU display	Among other diagnostics, it shows the redundancy state (Active, Standby, Inactive, Not-Configured, Starting). Through the menu options, it allows switchover commands or transitions between redundancy states for maintenance.
Redundancy diagnostics	- It indicates failures both in CPUA and CPUB, regardless of their states (Active or Inactive). - It prevents "obscure failures". - It allows a quick maintenance, which is essential to get high availability.
Redundancy commands	- It allows the execution of the same CPU display actions, among other commands (e.g. switchover command). - It can be executed in the local CPU, or transmitted to the other RTU (remote) via synchronism channels between the CPUs. - It can be received through the MasterTool Xtorm or a SCADA system. - It can be executed through the user application.
Redundancy events	It registers diagnostics and redundancy commands changes, with timestamp, allowing an investigation of the switchover causes.
IRIG-B, DNP3 and SNTP (Simple Network Time Protocol)	It allows the events to have a precise timestamp adjusted to the world hour. It also synchronizes the CPU real time clock for other applications.
MasterTool screens for Diagnostics, Commands and Events of Redundancy LOG	MasterTool Xtorm programmer offers special screens for commands and redundancy diagnostics, as well as to show the redundancy LOGs.
User data synchronization	At each MainTask cycle, CPUA and CPUB exchange diagnostics and commands. The user has 128 redundant data bytes available within the redundant CPU diagnostics. The synchronism occurs through the channels between the CPUs. This way, a CPU knows the diagnostics and commands of the other.
Redundant data synchronization	At each MainTask cycle, the Active CPU copies redundant data to the Inactive CPU through the synchronism channels. Non-redundant data are not synchronized.
Redundant forcing list synchronization	At each MainTask cycle, the Active CPU copies the redundant forcing list to the Non-Active CPU through the synchronism channels between the CPUs. This list includes only redundant forced variables. In this way, UCPA and UCPB may have different non-redundant data sets forced, since these forcings are not synchronized. The event queue is also synchronized in this CPU procedure, even if the list is not enabled by the user it will be synchronized between the CPUs even if it is not used by the user. For more information see chapter Redundant Data Synchronization.

	Redundant CPU General Features
Single project for CPUA and CPUB	There is a single common project for the UCPA and UCPB, generated by the MasterTool. The project is composed by the application project (executable code) and the *project archive* (source code).
CPU identification	The CPUs are identified as UCPA or UCPB automatically, depending on the position in which they are inserted. When inserting a CPU in position 2, it receives the designation UCPA, and in position 3, UCPB. This identification is not part of the application project generated by MasterTool Xtorm. The identification of each CPU enables the feature of having a unique project for both UCPA and UCPB.
Automatic synchronization of the project	If the Active CPU's project becomes different from the Non-Active CPU's project, it is copied from the Active CPU to the Non-Active CPU. This synchronization can take several MainTask cycles. Remember that the project is composed of the application project (executable code) and the source code (project archive), both of which are synchronized.
Private IP addresses for CPUA and CPUB	In order to obtain specific diagnostics of a CPU it is possible for the user to connect to a specific CPU (A or B) using its private IP address, for instance. For that, the user should use the Web interface or the MasterTool programming tool. It is only possible to check the diagnostics associated to communication protocols in the active CPU.
Active IP	Name of a strategy that allows Ethernet clients connect to a server in the redundant CPU using always the same IP address. This prevents the necessity of complex scripts to change the IP address in case of switchovers caused by redundancy. The Active IP address will always be associated to the NET(i) interface of the Active CPU.
NIC Teaming	Name of a strategy that allows two CPU Ethernet interfaces to form a redundant pair using the same IP address. Thus, it makes easier the building of redundant Ethernet networks. Furthermore, there is no need for the clients (connected to a NIC Teaming pair) to implement complex scripts in order to switch IP addresses.
Cyclic user task	Only one user task is allowed (MainTask). It is designed to be the primary task, with the bigger part of the user logic.

altus

	Redundant CPU General Features
Main POUs	At a redundant project creation, MasterTool automatically generates two empty programs POUs, which must be filled by the user. POUs associated to MainTask: - NonSkippedPrg: This program POU is executed in both CPUs (A and B), regardless of the redundancy state (Active or Inactive). It is used for diagnostics and special commands management. - UserPrg: This program POU is executed only in the Active CPU It is designed for the final user's process control. - StartPrg: This program POU is executed in both CPUs (A and B), only in the first cycle of MainTask. POUs associated to ProtTask: - MainPrg: This POU calls other POUs and administrates them. It cannot be edited. - AlarmPrg: This POU will be associated to the alarms, which are being used in the user's application. It cannot be edited. -EngineeringPrg: As the AlarmPrg, it is also automatically created, except that it deals with the Project engineering processes. It cannot be edited.

Table 219: Redundant CPU General Features

6.3.   Operation Principles

In this section, the functions of a redundant CPU, its behavior and states are described. In addition, programming and configuration concepts and constraints that will be used in the following chapters are presented.

6.3.1.   Single Redundant Project

Due to the identification register previously described, there is a single project for the redundant CPU, identical for both CPUA and CPUB

Configuration parameters that must be different for CPUA and CPUB (e.g. Ethernet interface IP addresses) are doubled in the redundant CPU project (one for the CPUA and another for the CPUB). Each CPU will take into account the correspondent one, after analyzing its identification register.

6.3.2.   Redundant Project Structure

6.3.2.1.   Redundancy Template

A redundant CPU project is automatically created from a model, named Redundancy Template. Tasks and basic POUs (program) are also created. The following sections describe the creation process.

6.3.2.2.   Cyclic Tasks: MainTask and ProtTask

The redundant CPU project can present two tasks: MainTask and ProtTask, both cyclic. The next item is optional and is created by the Wizard. The user can adjust the task cycle time.

6.3.2.3.   MainPrg Program

The MainTask is connected to a single POU (program type), called MainPrg. The MainPrg program is created automatically.

The following code refers to the MainPrg, in ST language:

altus

```
(*Main POU associated with MainTask that calls StartPrg,
UserPrg, NonSkippedPrg and the system POUs AlarmPrg and EngineeringPrg.
This POU is blocked to edit.*)

PROGRAM MainPrg
VAR
    isFirstCycle : BOOL := TRUE;
END_VAR

IF isFirstCycle THEN
    StartPrg();
    isFirstCycle := FALSE;
ELSE
  NonSkippedPrg();
  IF fbRedundancyManagement.bActiveCPU THEN
     EngineeringPrg();
     AlarmPrg();
     UserPrg();
  END_IF;
END_IF;
```

MainPrg calls two other POUs of program type:

- In the first execution cycle, only *StartPrg* is called. It is executed in both Active and Non-Active CPUs because the variables in MainPrg are not redundant.
- After the first execution cycle, *NonSkippedPrg* is always called, since it is executed in both Active and Non-Active POUs. The POUs *EngineeringPrg, AlarmPrg* and *UserPrg* are only called when the condition *" fbRedundancyManagement.bActiveCPU = TRUE"* is true, i.e. when the CPU is in an Active state.

Therefore, the *NonSkippedPrg* program will always be executed in both CPUs (UCPA and UCPB), regardless of the redundancy state of this CPU. The *EngineeringPrg, AlarmPrg* and *UserPrg* programs will execute only on the CPU that is in Active state.

As opposed to the *MainPrg* program, which cannot and should not be modified, the user can modify the other programs. Initially, when the redundant project is created from the *Redundancy Template*, these programs are created "empty", but the user can insert code into them.

6.3.2.4. UserPrg Program

The main goal of this program, which is executed only in the active CPU, is to control the final user process.

This program normally acts over the redundant variables, among which are the direct representation variables %I and %Q associated with the local I/O system. For more information, see the section MainTask Configurations.

6.3.2.5. NonSkippedPrg Program

This program runs in both CPUs (CPUA and CPUB) regardless of the redundancy state. It is typically used for functions such as:

- Manage switchover conditions, which normally are not contemplated by, default by the redundant CPU, that is, they can vary from user to user. E.g., a given user will be able to execute a switchover to the Stand-by CPU if the Active CPU is not in communication with the SCADA system. As for other user, for instance, one would not whish a switchover on this situation.
- Other activities, which, for some reason, need to be executed in both the Active CPU and the Standby CPU.

For further information, see the section NonSkippedPrg Program.

6.3.2.6. ProtPrg Program

The ProtTask is associated to a single POU (program type). This program is created in case the user defines its use during the project creation for redundancy purposes.

The following code refers to the ProtPrg, in ST language:

```
NonSkippedProtPrg();
IF fbRedundancyManagement.bActiveCPU THEN
   UserProtPrg();
END_IF
```

ProtPrg call other POUs of program type:

NonSkippedProtPrg is always called, as it runs in both Active and Non Active CPU and presents a higher priority than NonSkippedPrg. As for the UserProtPrg it is called only whenever the condition *"fbRedundancyManagement.bActiveCPU = TRUE"* is true; in other words, when the CPU is in active state.

Therefore, NonSkippedPrg will always run in both CPUs (CPUA and CPUB), regardless of its redundancy state. As for the UserProtPrg program, it will run only in the active CPU. This POU presents a higher priority than UserPrg.

Oppositely to the ProtPrg, which cannot and must not be modified, the user may modify other programs. When the redundant project is created from the Redundancy Template, the programs are "empty", although it is possible to the user insert them a code.

6.3.2.7. UserProtPrg Program

The UserProtPrg program is intended to control processes with a higher priority than UserPrg. That is, in this POU the user must insert its process logics that require a quick action of the CPU.

This program is called by *ProtPrg*, which, in turn, is associated with the *ProtTask* task.

For more information about the *ProtTask*, see section ProtTask Configurations.

6.3.2.8. NonSkippedProtPrg Program

This program, which runs in both CPUs regardless of their redundancy state, features a higher priority than "NonSkipped-Prg". It is commonly used for functions such as:

- Activities that, for some reason, need to run in the Active CPU and in the Stand-by CPU with high priority.

For further information, see the NonSkippedProtPrg Program section.

6.3.2.9. Redundant and Non-redundant Variables

The variables in a redundant CPU can be classified between redundant and non-redundant. Redundant variables are copied from the Active CPU to the Non-Active CPU at the beginning of each MainTask cycle through the synchronism channels between the CPUs. On the other hand, non-redundant variables are not copied between the CPUs and therefore may have different values in the Active CPU and Non-redundant CPU.

The non-redundant variables are used to store information that is private to each CPU (UCPA or UCPB), such as communication driver diagnostics and CPU diagnostics, including redundancy diagnostics (redundancy status of this CPU, etc.).

Redundant variables regard the shared information and the process control. The variables associated to I/O modules are typical examples of redundant variables.

6.3.2.10. Redundant %I Variables

Typically, the direct representation input variables (%I) are allocated to store the signals read from the digital and analog input modules of the local bus.

In projects with CPU redundancy that have input modules on the local bus, there is a range of %I variables, which is automatically redundant. This range always starts with the %I variable of address 0 and extends to the %I most used in local input modules.

Projects with CPU Redundancy that do not have input modules on the local bus will not have redundant direct representation input variables %I.

6.3.2.11.   Redundant *%Q* Variables

Typically, the direct representation output variables (%Q) are allocated to store the signals written in the digital and analog output modules of the local bus.

In projects with CPU redundancy that have output modules on the local bus, there is a range of %Q variables, which is automatically redundant. This range always starts with the %Q variable of most used address in local output modules.

Projects with CPU Redundancy that do not have output modules on the local bus will not have redundant direct representation output variables %Q.

6.3.2.12.   Redundant and Non-Redundant Symbolic Variables

Besides the direct representation variables (%I, %Q) which are automatically allocated, the user can explicitly declare symbolic variables, inside of POUs or GVLs. The maximum size allowed for redundant symbolic variables allocation is 512kbytes.

> **ATTENTION**
>
> Symbolic variables must not be mistaken with symbolic variables addressed through the AT directive. These are mere symbolic names assigned to direct representation variables (%I, %Q and %M), using the "AT" declaration. Thus, AT variables do not allocate any symbolic variables memory.

Symbolic variables are redundant in the following cases:

- When declared in POUs of type *"program"* created in the user application, except for the *NonSkippedPrg* and *NonSkippedProtPrg* programs.
- When declared in redundant GVLs created in the user application.

Symbolic variables are not redundant in the following cases:

- When declared in the *NonSkippedPrg* and *NonSkippedProtPrg* programs described previously.
- When declared in POUs of type *"function"*. Note that such types of POUs should normally allocate variables only on the pile (non-static), which consequently need not be redundant. Even though the user may declare static variables *(VAR STATIC)* within POUs of type *"function"*, this is considered bad programming practice. Such static variables, if created, will be considered non-redundant.
- When declared in POUs of type *"functional block"*. Note that merely declaring a *"functional block"* does not allocate memory (what allocates memory is instantiating a Functional Block).

Note that function blocks instances when declared inside POUs of program type or inside GVLs, behave as symbolic variables; in other words, they allocate redundant memory. As the symbolic variables, when function block instances are declared in the programs *NonSkippedPrg* and *NonSkippedProtPrg*, or when the GVL is not marked as redundant, such instances are non-redundant.

6.3.3.   Multiple Mapping

If the user wishes to map the variables of redundancy command in more than one communication port (COMx or NETx) it is necessary to implement a control process within the application.

The control logic to be implemented must write in the variables of redundancy command based on the value (commands) coming from each communication port (COMx or NETx). Besides that, the control logic must restart the variables of the communication ports, as the redundancy control just restarts its own command variables.

The following is an example of this implementation:

```
VAR
  var_StandBy_command_Ethernet_relation:  : BOOL;
  var_StandBy_command_Serial_relation:    : BOOL;
END_VAR

//Logic to put the local CPU in StandBy
IF var_StandBy_command_Ethernet_relation = TRUE THEN
  DG_HX3040_01.RedCmdLoc.bStandbyLocal:=TRUE;
  var_StandBy_command_Ethernet_relation:=FALSE;
```

```
  END_IF
  IF var_StandBy_command_Serial_relation = TRUE THEN
    DG_HX3040_01.RedCmdLoc.bStandbyLocal:=TRUE;
    var_StandBy_command_Serial_relation:=FALSE;
  END_IF
```

Above there is an example of logic in ST language, where the redundancy switchover commands can be executed through two variables from different communication ports.

Where:

var_StandBy_command_Ethernet_relation: Bool type variable assigned to an Ethernet communication Coil that will execute the command to put the local CPU in Stand-By.

var_StandBy_command_Serial_relation: Bool type variable assigned to a Serial communication Coil that will execute the command to put the local CPU in Stand-By.

DG_HX3040_01.tRedundancy.RedCmdLoc.bStandbyLocal: This command executes an action similar to command "Switch to Stand-by" triggered by the display menu, in the local CPU.

6.3.4. Diagnostics, Commands and User Data Structure

Each CPU has several data structures related to redundancy. The following structures belong to the DG_HX3040_01 symbolic variable, which is automatically allocated and is available to the user:

- *RedDgnLoc:* Provides diagnostics from the local CPU related to the redundancy, as the CPU redundancy state, for instance.
- *RedDgnRem:* It is a copy from the other CPU RedDgnLoc, which was received through the redundant synchronism channels. Thus, this local CPU can access the remote CPU's diagnostics.
- *RedCmdLoc:* Provides commands, which must be applied on the CPU with the "Local" suffix or on the CPU with "Remote" suffix. E.g., the StandbyLocal field of this data structure corresponds to a command, which must be executed in the local CPU, while the StandbyRemote field corresponds to a command, which must be executed in the remote CPU.
- *RedCmdRem:* It is a RedCmdLoc CPU's copy, received through the redundant synchronism channels. It is used only for visualization or information purposes.
- *RedUsrLoc:* It contains 128 data bytes freely filled by the user (e.g. communication diagnostics with a SCADA system). These 128 data bytes can be interchanged with the Remote CPU.
- *RedUsrRem:* It is a RedUsrLoc CPU's copy, received through the synchronism channels.

In the Maintenance section, the following subsections will provide more detail about the Redundancy Diagnostics Structure.

- Redundancy Diagnostics
- Redundancy Commands
- User Information Exchanged among CPUA and CPUB

6.3.5. Cyclic Synchronization Services Through Redundancy Synchronism Channels

This section describes the three synchronization services, which occur cyclically in a redundant CPU between CPUA and CPUB, through two internal synchronism channels. These channels are designed to synchronize redundant variables, diagnostics, redundant user memory area, events queue, project synchronization and commands.

These services are executed at the beginning of each MainTask cycle, in the following order:

- Diagnostics Exchange and Commands (First)
- Redundant Data Synchronization (Second)

These services are executed in both STOP and RUN mode. Variables, which are changed and forced through monitoring in MasterTool, are synchronized between the CPUs, even when in STOP mode.

6.3.5.1. Diagnostics and Commands Exchange

This service is responsible by the interchange of the following data structures, in each MainTask cycle:

- Copying *RedDgnLoc* from CPUA to *RedDgnRem* of CPUB
- Copying *RedCmdLoc* from CPUA to *RedCmdRem* of CPUB
- Copying *RedUsrLoc* from CPUA to *RedUsrRem* of CPUB
- Copying *RedDgnLoc* from CPUB to *RedDgnRem* of CPUA
- Copying *RedCmdLoc* from CPUB to *RedCmdRem* of CPUA
- Copying *RedUsrLoc* from CPUB to *RedUsrRem* of CPUA

This service will always be executed, as long as there are two CPUs in the bus (in the correct slots) and as long as the application present in both CPUs is redundant.

6.3.5.2. Redundant Data Synchronization

This service is responsible for transferring data from the Active CPU to the Non-Active CPU, which includes:

- Events queue
- Redundant variables (symbolic and direct representation variables - %I and %Q)
- Redundant variables forcing list

For this service to be executed, several conditions must be met:

- The previous synchronization service in this MainTask cycle (Diagnostics and Commands Exchange) must be successfully completed.
- If this CPU is in Active state, the other must be in Non-Active state. On the other hand, if this CPU is in Non-Active state, the other must be in Active state.
- The projects of the two CPUs must be identical. If a CPU is exchanged (hot-swap) and the new CPU does not contain an identical program, this synchronization will only be performed after Project Synchronization replaces the program of the new CPU.

> **ATTENTION**
>
> The synchronization of the event queue is always performed for all event contained in it, whether these events were generated from redundant or non-redundant variables.

> **ATTENTION**
>
> The redundant forcing list contains only information about redundant variables. In each one of the CPUs (A and B), there may be a different forcing list of non-redundant variables. The forcing of non-redundant variables is not synchronized between the CPUs. In order to force a non-redundant variable, you must first Login into the CPU and then force it.

6.3.6. Sporadic Synchronization Services through Redundancy Synchronism Channels

The following synchronization services are executed sporadically. That is, they are not executed every cycle of the Main-Task, but another task in the system executes these sporadic services in the background.

6.3.6.1. Project Synchronization

This service synchronizes the projects in both CPUs (Active and Non-Active). This situation occurs only when dealing with different projects.

The synchronization always goes from the Active to the Non-Active CPU. It is enough that one out of the two synchronism channels is operational for this service to be executed.

When the synchronization is enabled, the following files and services will be synchronized:

- Project application (executable code)
- Project archive (source code)
- User and groups

- Access rights
- Trace

The synchronization service will start within thirty seconds after one of the CPUs goes into Active state, and after its beginning, the project CRC will be checked at every five seconds.

When a synchronization is started (if the CPU is in RUN mode), the Non-Active CPU goes into Stop mode, at the Not-Configured state. After all required files were transferred, the Non-Active CPU goes to the same mode of the Active CPU (STOP or Run), at the Starting state.

The time the synchronization will take to be fully executed depends on the project size. In average, a transfer rate between the synchronism channels is approximately 500 Kbytes/s.

In case the synchronization is interrupted (communication loss between synchronism channels) during the files transferring from the Active CPU to the Non-Active CPU, the procedure is aborted and restarted when the communication is restored. Only after the conclusion of the whole procedure, the Non-Active CPU goes to the same mode of the Active CPU (STOP or RUN).

In addition to keeping projects synchronized, Project Synchronization will also prevent the Non-Active CPU from assuming states after Initializing, if the CRC is different or some Online Change is pending in the Active CPU.

> **ATTENTION**
>
> Project synchronization will effect as a download in the Non-Active CPU.

6.3.7. Redundant Ethernet Networks with NIC Teaming

Each CPU may have one or more network protocols configured for communication with the control center.
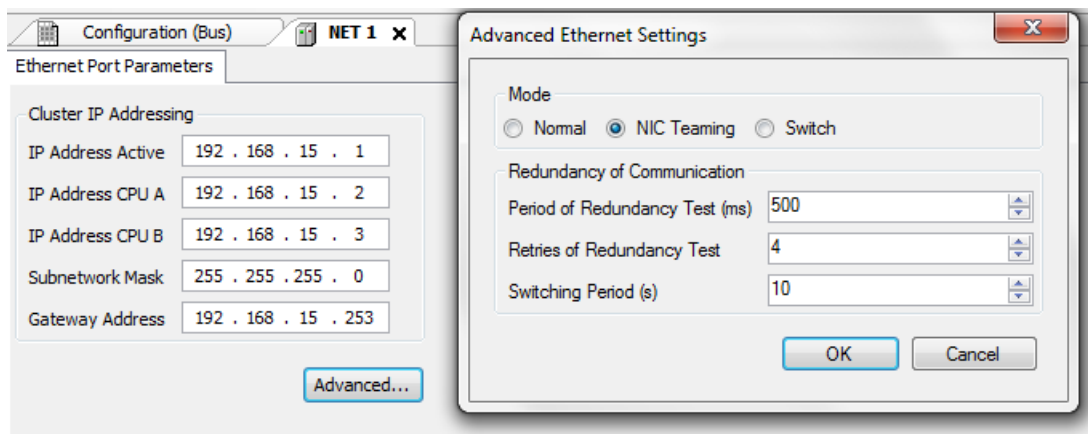
Figure 207: Redundant Ethernet Network with NIC Teaming

In this example the HX3040 CPU connects to the supervisory network (SCADAs). The two Ethernet ports of the HX3040 CPU (NET 1 and NET 2) form a redundant NIC Teaming pair, interconnected in two different switches (Switch 1 and Switch 2). At some point, these two switches must be interconnected, so that there is connection between the two NIC Teaming ports and even higher availability (against double failures).

This Ethernet architecture enables high availability of system communication and is strongly recommended for bridging failures on Ethernet ports, cables, and switches.

> **ATTENTION**
>
> If two modules or Ethernet interfaces form a NIC Teaming redundant pair, the basic parameters configuration and the addition of communication protocols (MODBUS, DNP3, etc.) will be only possible in the first interface. The second interface will have its configuration parameters blocked for edition.

More details on configuration and diagnostics of NIC Teaming ports are provided in the HX3040 CPU Ethernet Ports Configuration (NET 1 to NET 6) section.

6.3.8. IP Change Methods

Projects carried out with a Hadron Xtorm Series redundant CPU provide a method for IP change of the Ethernet ports NET1 to NET6. This method, named Active IP, defines the ports' behavior, regarding its IP, according to the current CPU redundancy state (Active or Non Active) and its identification (CPUA or CPUB).

Each NET interface requires three IP addresses. For NIC Teaming or Switch mode, the system requires three IP addresses for the pair of NETs rather than making individual settings.

6.3.8.1. Active IP

This method is the method used in the NETs of the redundant HX3040 CPU. In this method there is one IP for the Active CPU and two other IPs, one for the CPUA and one for the CPUB. On redundant HX3040 UCP NETs, the Active IP Address will be related to the Active CPU interface.

The MODBUS communication protocol, both client and server, use the local IP of the CPU, while for the DNP3 protocol the Active IP is used to communicate.

For communication with the Mastertool, the specific IP address of the CPUA or CPUB must be used.

Figure 208: Active IP Method – Redundant HX3040

Parameters that must be considered in Active IP method of a HX3040 redundant CPU:

- *IP Address Active:* IP address related to the CPU when it is in Active state;
- *IP Address CPU A:* CPUA communication address, no matter its state;
- *IP Address CPU B:* CPUB communication address, no matter its state;
- *Subnetwork Mask:* Subnet mask of the CPUs on the Ethernet bus;
- *Gateway Address:* Gateway address of the Ethernet bus subnet.

6.3.9. NIC Teaming and Active IP Combined Use

If a given port pair forms a NIC Teaming pair in a redundant CPU, these ports can implement NIC Teaming and Active IP strategies at the same time.

For example, if the NET 1 and NET 2 ports of the HX3040 CPU form a NIC Teaming pair, then:

- *IP Address CPU A:* IP address of the ports NET 1 + NET 2 of CPU, which is in position 2 of the bus.
- *IP Address CPU B:* IP address of the ports NET 1 + NET 2 of CPU, which is in position 3 of the bus.
- *IP Address Active:* IP address of the ports NET 1 + NET 2 of that CPU which is in Active state.

Thus, the Hadron Xtorm Series combines the excellent availability of NIC Teaming strategy with the practicality of Active IP strategy, releasing scripts in SCADA systems or other clients connected to servers in the Active CPU.

6.3.10. Redundant CPU States

In a redundant system, the CPU (CPUA or CPUB) may present the following states:

- Active
- Stand-by
- Inactive
- Non-Configured
- Starting

> **ATTENTION**
>
> Frequently this manual will use the designation *"Not-Active"* for any state other than Active, that is, to designate any of the other four states: *Stand-by, Inactive, Non-Configured* and *Starting*.

In the following sections, these five states are briefly described. For further details regarding the redundant CPU see the Transition between Redundancy States section, which also describes the states machine and the transition causes among them.

6.3.10.1. Non-Configured State

This is the initial redundancy state. The CPU is in this state when:

- The CPU is OFF (convention);
- Before starting the MainTask;
- Before there is a switch into the Starting state;
- In case there is a restart through commands like Reset warm, Reset cold or Reset origin.

In case the MainTask runs in Non-Configured states, the CPU performs the following tasks:

- The bus modules are not controlled by the CPU;
- The cyclic synchronization services are executed as long as the requirements for that are fulfilled (see section Cyclic Synchronization Services Through Redundancy Synchronism Channels;
- The sporadic synchronization services can also be executed (see section Sporadic Synchronization Services through Redundancy Synchronism Channels).

The CPU will be blocked in the Non-Configured state if the other CPU is in Active state, and this CPU project is different from the Active CPU project. In a situation other than that, the CPU goes from the Non-Configured state for the Starting one, as soon as the configuration request arrives.

Sometimes, the CPU goes into the Non-Configured state when it has already received an automatic configuration request. In this case, the user does not need to make a new request for changing the state. This happens at the CPU energizing, for instance.

In other situations, the user must request this configuration manually, e.g. pressing a button over the CPU's display menu. Manual configuration requests usually are not necessary in maintenance situations (before leaving the Non-Configured state). E.g. if the CPU has not reached the NonConfigured state due to some failure.

After leaving the Non-Configured state, it is possible for the CPU to go back to it in events such as:

- Restarting (Reset warm, cold or origin)
- CPU switch off
- Different projects between the current CPU and the Active CPU

6.3.10.2. Starting State

Unlike the other 4 states, which can last indefinitely, the Starting state is temporary and goes on for few seconds. The CPU reaches the Starting state from the Non-Configured one, through a configuration request.

At the beginning of the Starting state, several actions, tests and verifications are executed, in order to decide which will be the next state:

- The CPU accounts only for the local bus, with no interference in the modules control.
- The CPU checks whether its identification is correct or not (CPUA or CPUB, according to where the CPU is inserted).
- The CPU checks if there are problems in the configuration parameters extracted from MasterTool project.
- The CPU executes cyclic synchronization services as long as the conditions for its execution are true (see section Cyclic Synchronization Services Through Redundancy Synchronism Channels.
- The CPU checks the compatibility of the firmware version between both CPUs.
- The CPU checks if the projects from both CPUs are equal.
- In case the other CPU is in Active mode, the current state analyses the possibility of establishing a passive communication, through the local bus of the modules. The passive mode is used to test the transmission and reception circuits and the physical layer, to avoid hidden flaws.
- In case the other CPU is in unknown state due to failures in the redundant synchronism channels, it analyses the possibility of establishing a passive communication through the local bus of modules.

Depending on the results of these verifications and tests, the CPU switches from the Starting state to any of the other ones.

6.3.10.3. Active State

In this state, the CPU controls the automated process, using the UserPrg program, which runs only in this state. The Active CPU also updates the remote I/O modules, keeping them in an operational mode and reading the inputs and writing the outputs.

The Active CPU also verifies its internal diagnostics and the user switchover requests to determine if a switchover is necessary. The CPU leaves the Active state only if it is aware that the other CPU is in Stand-by mode, ready to take the control.

However, there are some situations where the Active CPU can leave the Active state even with no certainty about the other CPU's state. That is the case when the CPU is removed of the rack, for instance.

6.3.10.4. Stand-By State

In this state the CPU is ready to be switched to the Active state in case there is a request for that (a failure in the current Active CPU, for example).

The Stand-by CPU also verifies its own diagnostics and can be switched to the Non-Configured or Inactive state, in case some failures occur.

The bus control remains in the passive state. The passive mode is used to test the transmission and reception circuits and the physical layer to avoid hidden flaws. Total failure may cause a switch to the Inactive state.

6.3.10.5. Inactive State

The CPU reaches this state after some failures, or due to a manual request before a programmed maintenance.

The bus control is kept in passive state. The passive mode is used to test the transmission and reception circuits and the physical layer in order to avoid hidden flaws.

Before switching to another state, correct the diagnosed failures or execute the maintenance. After that, make the transition for the Non Configured state by requesting a configuration. Then, switch to the Starting state. Now, it is possible for the CPU:

- Return to the Inactive state, if failures remain happening;
- Return to the Not-Configured state, in case of other failure types;
- Go to Stand-by state, if the other CPU is in Active state;
- Go to Active state, if the other CPU is not in Active state.

6.3.11. Commands of the Redundancy Menu of the CPU's Display

This section describes the functions of redundancy menu commands, accessed by the CPU's display.

The command *"Switch to Stand-by"* has the following functions:

- Request a switch from the *Active* state to the *Stand-by* state, which may be useful to perform some scheduled maintenance in the Active CPU. After the Active CPU switches to Stand-by (and consequently the Stand-by CPU switches to Active), it is possible to switch it from Stand-by to Inactive using the *"Switch to Inactive"* command, and then perform scheduled maintenance in the Inactive state.
- Request a configuration that causes a switch from the *Non-Configured* state to the *Starting* state, typically after repairing failures that caused the transition to the Non-Configured state. After the Starting state, the CPU is typically expected to switch to Stand-by state (or Active, if the other CPU is not in the Active state).
- Request a switch from the *Inactive* state to the *Non-Configured* state already requesting a configuration. This typically occurs after correcting failures that caused the transition to the Inactive state. After the Non-Configured state, the configuration should already lead to the Starting state. After the Starting state, the CPU is normally expected to switch to Stand-by state (or Active, if the other CPU is not in Active state).

The *"Switch to Inactive"* command requests a switch from the Stand-by state to the Inactive state, which can be useful for performing some scheduled maintenance in the Stand-by CPU. After this maintenance, the command *"Switch to Stand-by"* can be used to attempt to return to the Stand-by state, passing through the Non-Configured and Starting states (see previous description of the functions of the *"Switch to Stand-by"* command).

> **ATTENTION**
>
> There are alternative ways to generate the same effects as the *"Switch to Stand-by"* and *"Switch to Inactive"* commands. The commands generated by this CPU or by the remote CPU can be used, as preliminarily described in the Diagnostics, Commands and User Data Structure section. A more detailed description of these commands can be found in the Redundancy Commands section.

6.3.12. Transition between Redundancy States

The following figure shows the state machine of redundancy, illustrating all possible transitions between redundancy states.
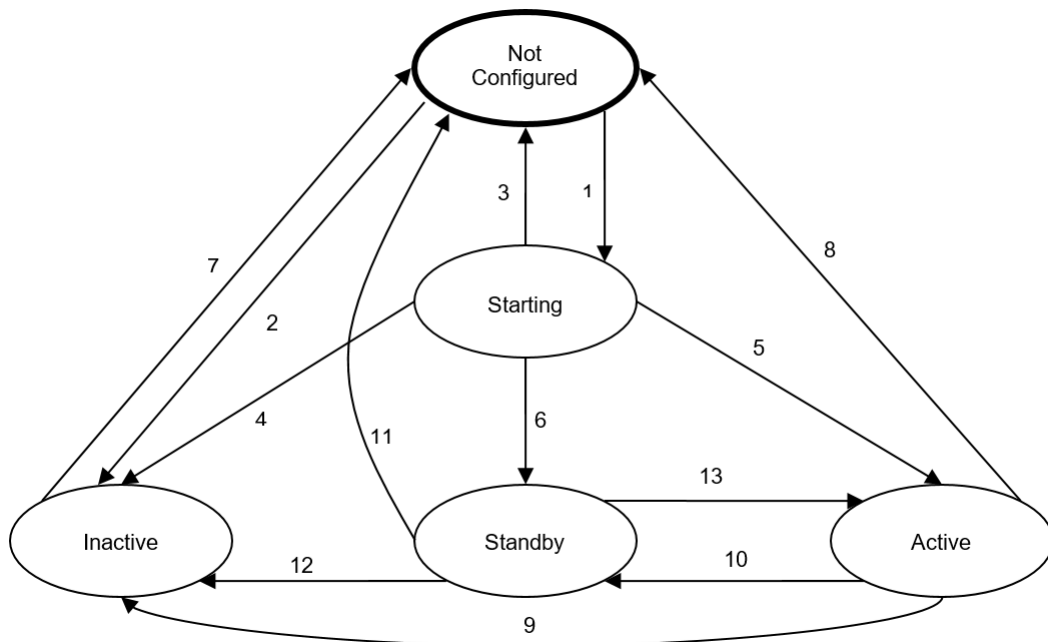
Figure 209: Redundancy Machine State

The following sections describe all these transitions as well as the causes that can trigger them. To correctly interpret the operation of this state machine, notice the following rules and sequences:

- Transitions that originate from the same state should be evaluated in the sequence given by the transition number. For example, transitions, 3, 4, 5 and 6 arise from the Starting state. In this example, the CPU first evaluates transition 3, then 4, 5, and finally 6. If the transition 3 is triggered, transitions 4, 5 and 6 and will not be evaluated.
- Within a particular subsection, several conditions can trigger a transition. Such conditions should be evaluated in the order they appear in the subsection. By the moment any of that become true, the transition arises. If a condition causes the transition, the next conditions will not be evaluated.
- The transitions occur in both situations of MainTask (STOP and RUN).
- There are several cases in which the commands available on the menu of the CPU display cause transitions. However, there are alternative internal commands, which are the derived from this CPU or the other CPU (via redundancy internal link). For further details, see Diagnostics, Commands and User Data Structure. Although these alternative commands are not mentioned in the following subsections, be aware that they can cause the same transitions as the commands, which are available in the display menu.
- The Reset Origin command removes the application; thus, the redundancy state is removed from the display.

6.3.12.1. Transition 1 – Non-Configured to Starting

- There is a prior configuration request by the moment the CPU entered in Non Configured state. This happens in the CPU startup and also in other situations, described in the following subsections.
- There is a request for the STAND-BY command during the Non-Configured state. Such situation requires a manual configuration. Typically, the user requests the STAND-BY control after repairing flaws that previously led this CPU to a Non-Configured or Inactive state.

6.3.12.2. Transition 2 – Non-Configured to Inactive

- There are some software exceptions in the current CPU (watchdog, access violation, illegal instruction).

6.3.12.3. Transition 3 – Starting to Non-Configured

- The current CPU is turned off or rebooted (Reset Warm, Reset Cold or Reset Origin).
- The current CPU is inserted in an incorrect position.
- There are configuration logic errors in the project received from MasterTool Xtorm.
- The other CPU is in Active state, and the firmware version of the current CPU is incompatible with the firmware version of the CPU Active.
- The CPU, which is in Active state, has a different project than the one from the current CPU. Besides going to Non-Configured state, the CPU requests a configuration. Thus, after the projects synchronization, the CPU automatically toggles from the Non-Configured to the Initializing state.

6.3.12.4. Transition 4 – Starting to Inactive

- There are some software exceptions in the current CPU (watchdog, access violation, illegal instruction).
- Some of the sync channels of the internal link redundancy (channel A or B) is in failure. The current CPU knows that this failure was caused by internal hardware or software components (internal failures of A or B channels).
- The other CPU is in Active state. However, you cannot synchronize the redundant data or the list of redundant forcing.
- The redundancy internal link is not able to find out the other CPU's state, even though it can monitor some activity on the bus. Therefore, although the internal link Redundancy does not work properly, the other CPU is controlling the process.
- The RUN/STOP states of the application are different from the Active CPU. For some failure reason, the CPU did not synchronized these application states to the other Active CPU.

6.3.12.5. Transition 5 – Starting to Active

- The other CPU is in Non-Active state. Before making this transition, this condition must remain true for some time. When CPUA and CPUB are energized simultaneously, the CPU that first finishes the system startup takes over as Active.
- The redundancy internal link is not able to find out the other CPU's state. Additionally, the CPU cannot monitor any activity on the bus. Therefore, this CPU is missing or it is out of run (software exception). For security reasons, the stand-by CPU switches to Active. This condition should be kept for some time before making this transition.

6.3.12.6. Transition 6 - Starting to Stand-by

- The other CPU is in Active state and the synchronization services of redundant data are working properly (event queue, redundant variables and list of forcing redundant variables).

6.3.12.7. Transition 7 – Inactive to Not-Configured

- The current CPU is turned off or rebooted (Reset Warm, Reset Cold or Reset Origin).
- There is a request for the STAND-BY command. Besides going to Non-Configured state, the CPU requests a configuration. Typically, the user requests the STAND-BY control after repairing flaws that previously led this CPU to an Inactive state.

6.3.12.8. Transition 8 – Active to Not-Configured

- The current CPU is turned off or rebooted (Reset Warm, Reset Cold or Reset Origin).

6.3.12.9. Transition 9 – Active to Inactive

- There are some software exceptions in the current CPU (watchdog, access violation, illegal instruction).
- The current CPU cannot control the bus and it is aware that the other CPU is in Stand-by. This condition is not evaluated for the first 2 seconds of the Active state.

6.3.12.10. Transition 10 - Active to Stand-by

- Both CPUs, for some reason, are in Active state. As this situation must be resolved, after a given time, CPUA switches to Stand-by. CPUB does the same in a shorter period of time. Thus, in case of conflict, CPUA has priority to keep on in Active state.
- The user pressed the STAND-BY button, and the current CPU knows that the other one is in Stand-by. This condition is not evaluated for the first 2 seconds of the Active state.

6.3.12.11. Transition 11 – Stand-by to Non-Configured

- The current CPU is turned off or rebooted (Reset Warm, Reset Cold or Reset Origin).
- The CPU, which is in Active state, has a different project than the one from the current CPU. Besides going to Non-Configured state, the CPU requests a configuration. There are cases in which the CPU remains Non-Configured (when the archive is sent to the Active CPU). In such cases, it is necessary a manual synchronization command. After the projects synchronization, the CPU automatically toggles from the Non-Configured to the Initializing state.
- The other CPU is in Active state, and the firmware version of the current CPU is incompatible with the firmware version of the CPU Active.

6.3.12.12. Transition 12 – Stand-by to Inactive

- There are some software exceptions in the current CPU (watchdog, access violation, illegal instruction).
- The user performs a transition to Inactive state. This is typically done for maintenance purposes on the Non-Active CPU. Avoid doing scheduled maintenance on the Stand-by CPU; if necessary, though, switch it into Inactive state.
- The other CPU is in Active state. However, the synchronization service of redundant data did not work properly in the last three MainTask cycles.

6.3.12.13. Transition 13 – Stand-by to Active

- The status of the other CPU is unknown due to failures on the redundancy internal link or the synchronization service of redundant diagnostics did not work properly in the last two MainTask cycles.
- The state of the other CPU is known and it is different from the Active one.
- This CPU did not detect any activity on the bus.

6.3.13. First Instants in Active State

There are several transitions, which could normally take the CPU out from the Active state. Such transitions are not evaluated in first 2 seconds, though (see previous subsections). During this period, for instance, the command *"Switch to Stand-by"* is useless.

Only two conditions allow the CPU to exit the Active state in the first 2 seconds. These conditions are as follows:

- The current CPU is turned off or rebooted (Reset Warm, Reset Cold or Reset Origin), which caused its transition to Non-Configured state. Another situation is the occurrence of some software exceptions in the current CPU (watchdog, access violation, illegal instruction), which caused its transition to Inactive state.

- Both CPUs, for some reason, are in Active state. As this situation must be resolved, after a given time, CPUA switches to Stand-by. CPUB does the same in a shorter period of time. Thus, in case of conflict, CPUA has priority to keep on in Active state.

Also, in the first instants that a CPU assumes the Active state, some diagnostics that are not redundant may not be valid. The method for disregarding these possibly "invalid" diagnostics are described in section Non-Redundant Diagnostics Reading.

6.3.14. Common Failures which Cause Automatic Switchovers between CPUs

This section lists the most common faults, which automatically cause a switchover between CPUs (Active to Non-active / Stand-by to Active). These failures trigger a subset of those transitions examined on the previous section Transition between Redundancy States.

- Failure in the internal power circuit of the Active CPU.
- The Active CPU cannot access to the rack bus.
- Failures in the Active CPU, such as:
  - Watchdog.
  - Reboot (Reset Warm, Reset Cold or Reset Origin).
  - Failure of the bus interfaces on one or both synchronization channels (NETA and NETB).
- Software Exception in the Active CPU (watchdog job, memory access error, etc).

6.3.15. Failures Associated to Switchovers between CPUs Managed by the User

Among the situations examined in Transition between Redundancy States section, some allow the user to manage switchovers between CPUs, due to failures that usually do not generate automatically switchovers.

There are rare cases that depend on the client's philosophy. Take as example a situation in which the SCADA system loses communication with the Active CPU

Some customers would prefer a manual switchover, where the operator executes the command from the menu on the CPU display. The switchover would retake the communication with the new Active CPU

Another possibility would be the Active CPU itself detect the loss of communication with the SCADA, and activate a command in RedCmdLocal using the RedCmdLocal data structures to carry an equivalent command to the one placed on the CPU display. This would be a fully automatic solution that would dismiss any intervention by the operator, and it would typically be implemented in UserPrg POU.

Using data structures like those mentioned in section Diagnostics, Commands and User Data Structure, it is possible to exchange diagnostics and commands between the CPUs via internal redundancy link, and in this way the user can perform special redundancy management for failures that normally would not cause switchovers. More details about these data structures are in the sections:

- Redundancy Diagnostics
- Redundancy Commands
- User Information Exchanged among CPUA and CPUB

The example below shows how the user can manage failures and perform a switchover in two cases. The former regards a "link down" error in the Ethernet interfaces of the Active CPU (broken network cable; use the code in the UserPrg POU). The later refers to the control of communication protocol errors in the serial ports:

```
//The CPU checks if NIC Teaming is enabled or not.
IF ((DG_HX3040.tDetailed.Ethernet.NET[1].szIP = '0.0.0.0') OR (DG_HX3040.
   tDetailed.Ethernet.NET[2].szIP = '0.0.0.0')) THEN
  //NIC Teaming enabled: error in both NETs to perform a switchover.
  IF (DG_HX3040.tDetailed.Ethernet.NET[1].bLinkDown AND  DG_HX3040.tDetailed.
   Ethernet.NET[2].bLinkDown) THEN
  //Local CPU in StandBy.
    DG_HX3040_01.RedCmdLoc.bStandbyLocal := TRUE;
  END_IF
  ELSE
  //NIC Teaming disabled: error in one NET to perform a switchover.
  IF (DG_HX3040.tDetailed.Ethernet.NET[1].bLinkDown OR  DG_HX3040.tDetailed.
   Ethernet.NET[2].bLinkDown) THEN
```

```
      //Puts the Local CPU in StandBy.
      DG_HX3040_01.RedCmdLoc.bStandbyLocal := TRUE;
   END_IF
 END_IF


 -------------------------------------------------------------------
 IF ((DG_HX3040.tDetailed.Serial.COM[1].byProtocol <> 0) OR (DG_HX3040.tDetailed.
    Serial.COM[2].byProtocol <> 0))THEN
   //If a communication error occurs, the CPU performs a switchover.
   IF MODBUS_Device_REQDG_0001.byStatus.bCommError THEN
     //Local CPU in StandBy.
     DG_HX3040_01.RedCmdLoc.bStandbyLocal := TRUE;
   END_IF
 END_IF
```

Notes:
- When two interfaces form a NIC Teaming pair, the inactive interface will always have the IP address 0.0.0.0. This is not a valid IP address and you cannot configure an interface manually with this address.
- The MODBUS_Device_REQDG_0001 diagnostic used in the previous logic can vary its name as you create these diagnostics. Preferably, use all diagnostic variables of each request of the project for this logic.

6.3.16. Failure Tolerance

The primary purpose of a redundant CPU is to increase the system availability. Availability is the ratio between the times in that the system is working properly and the total time from the system implementation. For example, if a system has been operational for 10 years, and during this period, it has been on hold for one year due to faults, and then its availability was only 90%. Critical systems generally do not support such availability; they can even require an amount of 99.99% or higher

In order to achieve such level of availability, the user can make use of the following strategies:

1. Use reliable components, with high MTBF (mean time between failures), which will contribute to increasing the MTBF system as a whole.;
2. Use of redundancy, at least for the most critical components or for components with low MTBF. In this case, the system tolerates eventual failures avoiding interruptions. In redundant architectures, it would be necessary a failure in both components to make the system unavailable;
3. High diagnostic coverage, especially redundant components. The redundancy of components is hardly useful for increasing availability when it is not possible find out which redundant component has failed. In this case, the first failure in one component does not compromise the system, but it remains hidden. Eventually a second failure will happen, and that will compromise the system, since the first one has not been repaired yet. Although it is possible for the system to hide some failures, for its sake, the best choice is diagnose all faults of the redundant components;
4. It is also important that non-redundant components have broad coverage of diagnostic, since it is not rare that system continues working even with a non-redundant component failure. That is the case when the system does not ask for this given component. Take for example a relay with open contact: as it rarely triggers its coil, the system will not detect its fault until the moment that it is prompted to close.
5. Low repair time for non-redundant components. The failure of a non-redundant component may compromise the system, and during the repair, the system will be unavailable.
6. Repair or replace a redundant component without stopping the system. If there is this possibility, there is a great increase of availability. If not, you must plan a system stop in order to replace the component, and thus the repair time will count as downtime.
7. Low time repair for redundant components. The failure of a redundant component may not compromise the system, but for its repair, eventual fails in the redundant pair may occur. For this reason, repair the fault right after its diagnosis. The longer the repair time, the more likely to happen a second failure in the redundant component during the repair of the first failure, which would undermine the system. Therefore, the longer the time of repair, the lower system availability.
8. Schedule periodic offline tests in the components, so as to detect non diagnosable faults. The goal is to detect hidden faults, particularly in redundant components (or even in simple ones, as long they, which are not usually requested, as a safety relay, for example. Sometimes offline tests involve stops in the system, which reduces availability. Special occasions such as programmed shutdowns for maintenance, for example, are good opportunities for this procedure. The longer the period between off-line testing, the longer the time in which a fault may be hidden, and hence higher the probability of eventual failures compromise the system, decreasing the system availability.

HX3040 CPU accounts for these principles when creating a project of redundant RTUs.

The following subsections analyze different types of faults and the tolerance to them, as well as the possibility of switchovers.

6.3.16.1. Simple Failure with Unavailability

As some components are not duplicated, they cannot support even single failure without causing some kind of unavailability. The redundant RTU with CPU HX3040 manage the following components:

- I/O Modules
- Racks (HX9001 or HX9003)

The unavailability of an I/O module does not undermine the whole system availability. It constitutes in a partial unavailability, regarding the control loops, which use this given I/O module.

Although there is no projection of I/O modules redundancy, the user application can manage it in special cases. For example, the user can enter three modules of different analog inputs, and implement a voting scheme between trios of analog inputs to any critical system. However, such solutions, as emphasized, should be managed by the user. There is no automated support for it. Such solutions generally involve also the redundancy in the field transducers and actuators.

6.3.16.2. Simple Failure without Unavailability Causing a Switchover

Redundant HX3040 CPUs may support single failures without causing unavailability, but they do cause switchovers.

6.3.16.3. Single Failure without Unavailability

HX8300 and HX8320 redundant power supplies support single failures without causing unavailability but doing a switchover.

6.3.17. Redundancy Overhead

Redundant applications demand a higher processing time when compared to an equivalent nonredundant application.

This additional time is mainly due to the execution of the cyclic synchronization services, described in section subsnamerefsubsubsec:CyclicSynchronizationServicesthroughRedundancySynchronismChannels, plus a lower time for the redundancy management itself (state machine, etc.). The total additional time due to redundancy (redundancy overhead) is estimated by the MasterTool and displayed in the Messages window after building the redundant CPU project.

The time for data transfer depends on the amount of data and follows an approximate ratio of 6.4 ms for each 100Kb of data. For the effective calculation of the maximum transfer time, a minimum time of 7ms (constant) is estimated and the remainder is calculated using the total amount of data (including the 117000 byte event queue). Thus, typically in an empty redundant application the maximum transfer time is calculated as 7ms + 6.4ms/Kbyte 117 Kbytes = 15ms. This time is represented by Mastertool as *"maximum redundancy overhead"*.

> **ATTENTION**
>
> MasterTool calculates the overhead considering a forcing list of empty of redundant variables.

The MainTask cycle time should be adjusted by taking into account the "maximum redundancy overhead" time calculated by Mastertool, plus 30% off. That is, in the case of 15ms, the cycle time must be at least 20ms. However, in redundant applications, the minimum time allowed is 50ms, due to the time needed for processing the event queue in the switchover.

> **ATTENTION**
>
> If the 30% time gap is not respected, the system will not ensure proper redundancy operation. What will normally occur is a change of state of one of the CPUs to INACTIVE, due to data synchronization failures between the CPUs.

In addition, the user must define a range for MainTask regarding:

- The time required to run the main POUs (NonSkippedPrg and UserPrg). This time is typically measured after the project development (discounting the additional time for redundancy);
- Time required for detection and generation of internal point events (for example, the occurrence of 1000 deadband analog point events in the same cycle can take up to 15ms);
- Some time off of the MainTask cycle, for execution of other CPU tasks (operating system, I/O drivers, etc). The percentage of this time off may vary according to the required performance of these other tasks. For example, if the MODBUS communication with the SCADA system needs to allocate a lot of processing to achieve satisfactory performance, this time off should be increased;

- The time required to execute the protection POUs (NonSkippedProtPrg and UserProtPrg), when used. Since the priority of the ProtTask associated with these POUs is higher than the priority of the MainTask, the frequency/interval of the ProtTask must be taken into consideration. The average consumption time of the ProtTask should be a maximum of 20% of its cycle time. This is equivalent to 800us if it is working with a 4ms interval. If this is not respected, the system will not be able to perform the redundant data transfer. The ProtTask should also not have surges much higher than the 20% of the cycle time, as this can even generate a switchover. Thus it is recommended not to have logic with loops that may diverge and consume too much time. For more information, see the section Cyclic Tasks: MainTask and ProtTask.

> **ATTENTION**
>
> Depending on the memory alignment, the number of bytes used to calculate the redundancy overhead may be larger than the total number of bytes declared in the variables.

6.4. Redundant CPU Programming

6.4.1. Wizard for a New Redundant Project Creation

To create a new redundant project, click *File -> New Projet*, and select the *MasterTool Standard Project*.

Initially, enter the project name and the directory where you want to store it, as shown in figure below.
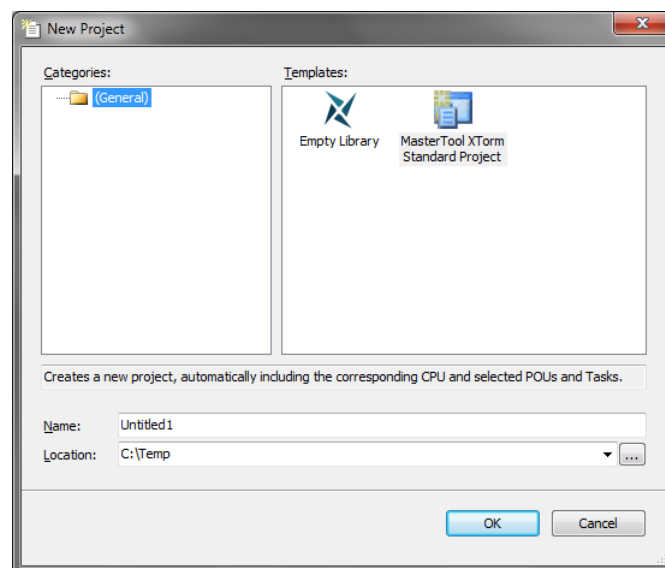
Figure 210: Project Classification

Next, the Wizard that generates the redundancy project prompts some information regarding the desired settings.

The first point is the hardware initial configuration of the RTU:

- Select the CPU model: As the redundancy is implemented only in HX3040, it shall be selected by the user;
- Select the rack model: There are two options of racks available, the choice depends on the number of used modules;
- Select the power supply model;
- Select the CPU redundancy configuration;
- Select the redundancy configuration of the power supply (this type of configuration for the power supply does not depend on the CPU redundancy configuration).

altus
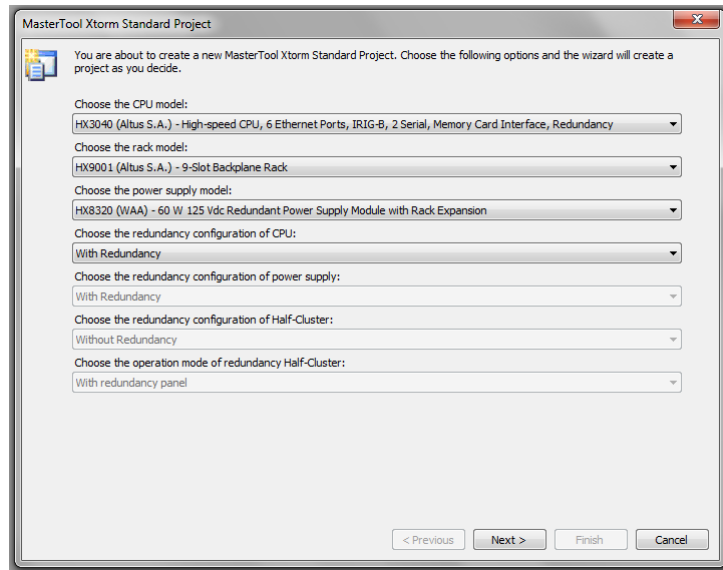
Figure 211: Hardware Initial Configuration

Then, define the amount and types of the application I/Os:

- Select the quantity of digital input points;
- Select the quantity of digital output points;
- Select the quantity of V/I analog input points;
- Select the quantity of RTD analog input points.
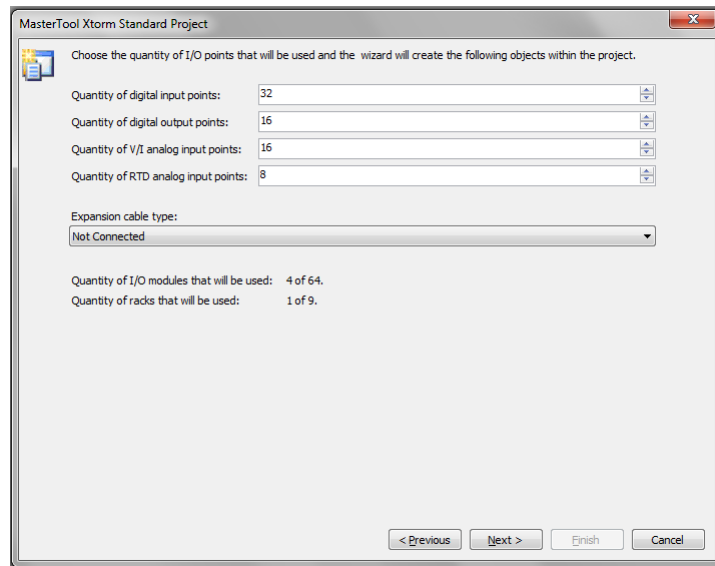
Figure 212: I/O Points Configuration

Next, select the project profile and the standard language for the program creation:

- Select project profile: Only the project profile "Profile for RTU" can be used for redundancy, so the selection option is locked for editing;
- Select the default language for all programs: the language selected by the user will be the default for all programs, but the user can choose to use any other language for a specific POU.
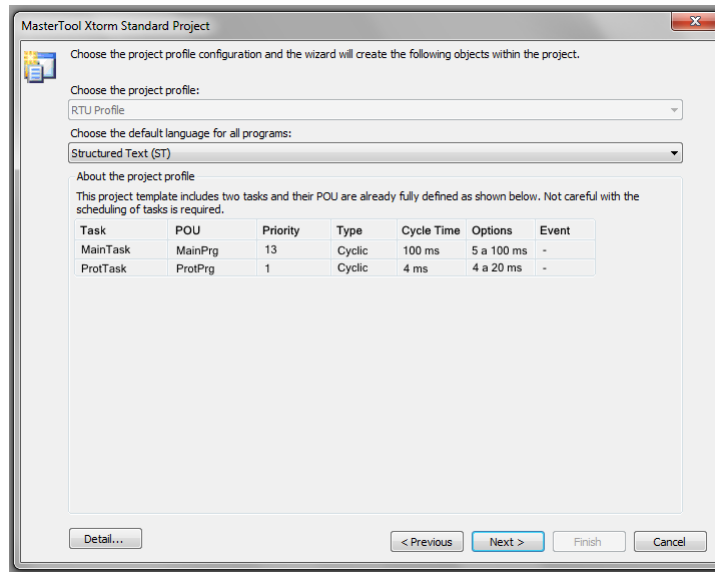
Figure 213: Project Profile and Default Language

To conclude, select the language for common programs and for the ones associated with redundancy:

- Programs associated with MainTask (MainPrg);
- Programs associated with cyclic tasks: ST only. MasterTool disables the other options;
- Programs associated with redundancy main tasks.
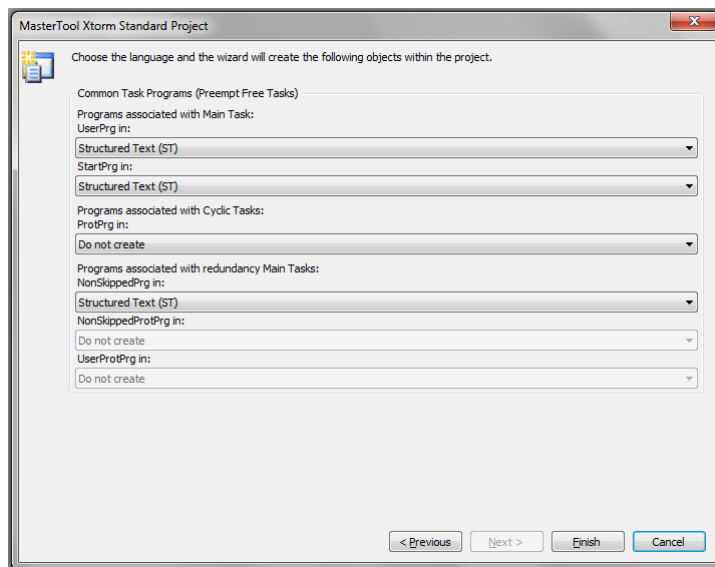
Figure 214: Language for Specific Programs

ATTENTION

The POUs UserPrg and NonSkippedPrg are created automatically, empty, in the selected language. When creating POUs manually, you can use any of the available languages, except for redundant POUs. Such units cannot be written in SFC language since it uses the IEC timer as background. For more information, see Limitations in the Programming of a Redundant CPU.

> **ATTENTION**
>
> The MainPrg POU will always be automatically generated in ST language, and it cannot be changed by the user. This POU calls UserPrg (only in Active CPU) and NonSkippedPrg (both CPUs). Equivalently, ProtPrg, when requested by the user to be created, it is automatically generated in ST language, and cannot be changed by the user either. This POU calls UserProtPrg (only in Active CPU) and NonSkippedProtPrg (both CPUs).

After the previous procedure, the Wizard generates the initial project, defining the system based in the hardware initial configuration:

- Rack
- Power supply (positions 0 and 1)
- HX3040 CPU (positions 2 and 3)
- I/O Modules (remaining positions in the rack)

6.4.2. Project Configuration with CPU Redundancy

The Wizard is always used to generate the first version of a redundant project. This ensures that the initial version of the project will be generated quickly and correctly.

However, it is possible that some modifications are necessary, such as the insertion of new I/O modules. The following chapters show how to add and configure them.

The following section describes the rules and precautions, which must be followed in order to get a proper redundant project.

6.4.2.1. Fixed Configuration in Positions 0 to 3 of the Rack

In the positions 0 to 3 of the selected rack, install the following modules:

- Redundant power supply (positions 0 and 1)
- Redundant HX3040 CPU (positions 2 and 3)
- I/O Modules (from position 4 on)

Do not remove these modules from the original project generated by the Wizard.

Any different configuration in such positions will result in an error notified by the MasterTool when compiling the project.

6.4.3. HX3040 CPU Ethernet Ports Configuration (NET 1 to NET 6)

6.4.3.1. IP Address Configuration

The figure below shows the settings of the NET 1 port of the HX3040 CPU. To open this screen, doubleclick the desired NET in the device tree.

Figure 215: NET 1 Ethernet Port Parameters

Then follow the basic parameters for the desired interface. The addressing is in accordance with the Active IP exchange method, as described in Active IP.

> **ATTENTION**
>
> The three IP addresses (Active, CPUA and CPUB) and the Gateway address must belong to the same subnet.

> **ATTENTION**
>
> The odd NETs (1, 3 and 5) have the Advanced button, while the even ones (2, 4 and 6) do not. Through this button, you can set the NETs redundancy (NIC Teaming) and select the Switch mode as well.

6.4.4. I/O Drivers Configuration

By default, I/O drivers are all communication drivers used by the CPU as Modbus Client, Server, Master, Slave, DNP3 Client drivers, Outstation, IEC61850, etc. These drivers work differently on CPUs with redundancy.

For an I/O Driver to run, the CPU must be in RUN mode and in Active redundancy state. This means that a CPU in Starting, Inactive or Stand-by modes will not have its I/O Drivers running, that is, its clients and servers (MODBUS, DNP3, etc.) are stopped.

Thus, It´s not possible to use the communication via I/O Drivers in a Non-Active state CPU.

6.4.5. MainTask Configurations

The settings screen associated with the MainTask (main task of a redundant CPU), which is cyclical, can be accessed by clicking on the correspondent item in the device tree.

Set the two following parameters:

- MainTask interval
- Watchdog time

For the proper adjustment of the MainTask interval, notice:

- The interval must be low enough to effectively control the process, noting the response times of all control loops.
- The interval must be high enough to accommodate at least the sum of the two following times:
  - The maximum runtime of NonSkippedPrg and UserPrg together
  - The time required to manage redundancy (redundancy overhead)
- In addition, the interval must have an additional clearance necessary for that other processes have time to run (Ethernet communication with SCADA systems, etc.)

MasterTool is able to calculate the time required to manage the redundancy (redundancy overhead), after the project is finished (POUs fully developed, and redundant memory areas defined).

As for the maximum execution time of NonSkippedPrg and UserPrg, you can measure it after the POUs development. After each compilation of the project, MasterTool sums the calculated redundancy overhead plus the parameter reporting the POUs time (NonSkippedPrg and UserPrg). Then, it checks if the parameterized minimum clearance is being obeyed.

For example:

- Parameters set on MainTask screen:
  - MainTask interval: 100 ms
  - Estimated time for POUs NonSkippedPrg + UserPrg: 10 ms
  - Minimum clearance: 30%
- Overhead calculated for redundancy: 50 ms

In this case, the total time taken is 60 ms (10 ms + 50 ms), which consists of 60% of MainTask cycle (100 ms). The clearance is 40% and, thus the minimum clearance of 30% being observed.

6.4.5.1. StartPrg Program

In this POU, the user can create logics and loops, and start variables as well. Such variables will run only once at the first cycle of each RTU, and thus will not be called again during the project execution.

If the user loads a new application, or if the PLC is turned on again, as well as under Reset Origin, Reset Cold and Reset Warm conditions, this POU will be executed again.

6.4.5.2. UserPrg Program

In this POU, the user must create the main application, which is responsible for controlling the process. This POU is called by the main POU (MainPrg), and it runs only in the Active CPU.

The user can also create additional POUs (program, function or function block), and call them or instantiate them within the UserPrg for structuring purposes. You can also call functions and instantiate function blocks defined in libraries.

Notice that all symbolic variables and function blocks instances defined in UserPrg will be redundant variables.

Symbolic variables which are defined in additional POUs of program type, even when called within UserPrg will only be redundant variables if you make them so (selecting this option in the Redundancy Configuration object). By default, all user-created POUs are initially redundant.

> **ATTENTION**
>
> Do not use VAR_TEMP variables in the redundant program.

6.4.5.3. NonSkippedPrg Program

This POU is intended for controls that must be performed on both CPUs (CPUA and CPUB), regardless of their state redundancy. This POU is called by the main POU (MainPrg).

Notice that all symbolic variables and function blocks instances defined in NonSkippedPrg will be non-redundant variables.

The user can also create additional POUs (program, function or function block), and call them or instantiate them within the NonSkippedPrg for structuring purposes of your program. You can also call functions and instantiate function blocks defined in libraries.

> **ATTENTION**
>
> When calling additional POUs of type program within NonSkippedPrg, uncheck them in the Redundancy Configuration window of MasterTool. By default the symbolic variables declared within these POUs will be redundant, and within NonSkippedPrg non-redundant variables are normally desired. Normally the NonSkippedPrg code is small and does not require calling additional program type POUs to structure it, but if structuring is required, it is recommended to use functional blocks or functions.

> **ATTENTION**
>
> It is not recommended to use functional blocks TOF_RET, TON_RET, TOF and TON in the NonSkippedPrg program. See section Limitations in the Programming of a Redundant CPU.

6.4.5.4. GVL Disables

The *Disables* GVL contains the MODBUS Master/Client by symbolic mapping requisition disabling variables. It is not mandatory, but it is recommended to use the automatic generation of these variables, which is done clicking in the button *Generate Disabling Variables* in device requisition tab. These variables are declared as type BOOL and follow the following structure:

Requisition disabling variables declaration:

```
[Device Name]_DISABLE_[Requisition Number] : BOOL;
```

Where:

Device name: Name that shows on Tree View to the MODBUS device.

Requisition Number: Requisition number that was declared on the MODBUS device requisition table following the sequence from up to down, starting on 0001.

Example:

Device.Application.Disables

altus

```
VAR_GLOBAL
  MODBUS_Device_DISABLE_0001   : BOOL;
  MODBUS_Device_DISABLE_0002   : BOOL;
  MODBUS_Device_DISABLE_0003   : BOOL;
  MODBUS_Device_1_DISABLE_0001   : BOOL;
  MODBUS_Device_1_DISABLE_0002   : BOOL;
END_VAR
```

The automatic generation through button *Generate Disabling Variables* only create variables, and don't remove automatically. This way, in case any relation is removed, its respective disabling variable must be removed manually.

The *Disables* GVL is editable, therefore the requisition disabling variables can be created manually without need of following the model created by the automatic declaration and can be used both ways at same time, but must always be of BOOL type. And it is need to take care to do not delete or change the automatic declared variables, cause them can being used for some MODBUS device. If the variable be deleted or changed then an error is going to be generated while the project is being compiled. To correct the automatically declared variable name, it must be followed the model exemplified above according to the device and the requisition to which they belong.

The following picture shows an example of the presentation of this GVL when in *Online* mode. If the variable values are TRUE it means that the requisition to which the variables belong is disabled and the opposite is valid when the variable value is FALSE.

Expressão	Tipo	Valor	Valor Preparado	Comentário
MODBUS_Device_DISABLE_0001	BOOL	TRUE		
MODBUS_Device_DISABLE_0002	BOOL	FALSE		
MODBUS_Device_DISABLE_0003	BOOL	TRUE		
MODBUS_Device_1_DISABLE_0001	BOOL	TRUE		
MODBUS_Device_1_DISABLE_0002	BOOL	FALSE		

Figure 216: GVL Disable in Online Mode

6.4.5.5. GVL IOQualities

The *IOQualities* GVL contains the quality variables of I/O modules declared on CPU's bus. This GVL is not editable and the variables are automatically declared as *LibDataTypes.QUALITY* type arrays, and dimensions according to I/Os quantities of the module to which it belongs when that is added to the project.

Example: Device.Application.IOQualities

```
VAR_GLOBAL
  QUALITY_HX1120: ARRAY[0..31]  OF LibDataTypes.QUALITY;
  QUALITY_HX2320: ARRAY[0..15]  OF LibDataTypes.QUALITY;
  QUALITY_HX6000: ARRAY[0..15]  OF LibDataTypes.QUALITY;
  QUALITY_HX6020: ARRAY[0..7] OF LibDataTypes.QUALITY;
END_VAR
```

Once the application is in *RUN* it is possible to watch the I/O modules quality variables values that were added to the project through *IOQualities* GVL.

6.4.5.6. GVL Module_Diagnostics

The *Module_Diagnostics* GVL declares the diagnostic variables of the modules used in the project, except for the CPU and communication drivers. This GVL is not editable, so the variables are automatically declared with the type specified by the module to which it belongs, when it is added to the project.

Example:

```
Module_Diagnostics
VAR_GLOBAL
  DG_HX1120 : T_DIAG_HX1120_1;
  DG_HX2320 : T_DIAG_HX2320_1;
  DG_HX6000 : T_DIAG_HX6000_1;
  DG_HX6020 : T_DIAG_HX6020_1;
  DG_HX8320 : T_DIAG_HX8320_1;
END_VAR
```

By sending an application to the HX3040 CPU and passing it to *RUN*. Through the GVL *Module_Diagnostics* it is possible to monitor the diagnostics of each module that has been added to the project, as can be seen in the figure below.

Figure 217: GVL Modules_Diagnostic in Online Mode

6.4.5.7. GVL Qualities

In the *Qualities* GVL the quality variables of the internal variable mappings, MODBUS Master/Client and DNP3 Client are declared. It is not mandatory, but it is recommended to use the automatic generation of these variables, which is done by clicking the *Generate Quality Variables* button in the device mappings tab. These variables are declared as *LibDataTypes.QUALITY* and follow the following structure:

Quality mapping variable declaration:

```
[Device Name]_QUALITY_[Mapping Number]: LibDataTypes.QUALITY;
```

altus

Where:

Device Name: Name that appear at the Tree View to the device.

Mapping Number: Number of the mapping that was declared on the device mapping table, following the up to down sequence, starting with 0001.

Example:

```
Qualities
VAR_GLOBAL
  MODBUS_Device_QUALITY_0001: LibDataTypes.QUALITY;
  MODBUS_Device_QUALITY_0002: LibDataTypes.QUALITY;
  MODBUS_Device_QUALITY_0003: LibDataTypes.QUALITY;
  Outstation_QUALITY_0001: LibDataTypes.QUALITY;
END_VAR
```

The GVL *Quality* is editable, so the quality variables of the mappings can be created manually without having to follow the template created by the automatic declaration and can be used both ways at the same time, but they must always be of type *LibDataTypes.QUALITY* and care must be taken to not delete or change the automatically declared variables, because they may be being used by some device. If the variable is deleted or changed an error will be generated when building the project. To correct the name of an automatically declared variable, you must follow the model exemplified above according to the device and mapping to which it belongs.

To the MODBUS communication devices the quality variables behave on the way showed at table 50.

> **ATTENTION**
>
> If a variable of the DNP3 Client or MODBUS Master/Client symbolic mapping drivers is mapped to the DNP3 Server or IEC 60870-5-104 Server drivers, the quality variables of the DNP3 or MODBUS mappings must have been created for valid quality events to be generated for such points of the DNP3 or IEC 60870-5-104 servers. Otherwise, "bad" quality events will not be generated for the clients of the DNP3 and IEC 60870-5-104 servers in situations that the DNP3 Client or MODBUS Master/Client cannot communicate with their slaves/servers, for example.

By sending an application to the HX3040 CPU and putting it in *RUN*. Through GVL *Qualities* you can monitor the diagnostics variable values of the MODBUS Mater/Client and DNP3 Client devices mappings, as shown in the figure below.

Figure 218: GVL Qualities in Online Mode

altus

6.4.5.8. GVL ReqDiagnostics

The *ReqDiagnostics* GVL contains the requisition diagnostics variables of symbolic mapping MODBUS Master/Client and DNP3 Client. It is not mandatory, but recommended the use of these variables' automatic generation, what is done by clicking in the button *Generate Diagnostics Variables* in device requests tab. These variables declaration follow the following structure:

Requisition diagnostic variable declaration:

```
[Device Name]_REQDG_[Requisition Number]: [Variable Type];
```

Where:

Device Name: Name that appear at the Tree View to the device.

Mapping Number: Number of the mapping that was declared on the device mapping table, following the up to down sequence, starting with 0001.

Variable Type: *"T_DIAG_MODBUS_RTU_MAPPING_1"* to MODBUS Master, *"T_DIAG_MODBUS_ETH_ MAPPING_1"* to MODBUS Client and *"T_DIAG_DNP_CLIENT_REQUEST_1"* to DNP3 Client.

Example:

```
ReqDiagnostics
VAR_GLOBAL
  MODBUS_Device_REQDG_0001  : T_DIAG_MODBUS_RTU_MAPPING_1;
  MODBUS_Device_REQDG_0002  : T_DIAG_MODBUS_RTU_MAPPING_1;
  MODBUS_Device_REQDG_0003  : T_DIAG_MODBUS_RTU_MAPPING_1;
  MODBUS_Device_1_REQDG_0001  : T_DIAG_MODBUS_ETH_MAPPING_1;
  MODBUS_Device_1_REQDG_0002  : T_DIAG_MODBUS_ETH_MAPPING_1;
  Outstation_REQDG_0001   : T_DIAG_DNP_CLIENT_REQUEST_1;
END_VAR
```

The *ReqDiagnostics* GVL is editable, so the diagnostic variables of the requests can be created manually and do not need to follow the model created by the automatic declaration. Actually, you can use both ways at once, but make sure to use variables related to the device, and ensure not to delete or change the variables automatically declared, as they might be in use by a MODBUS device. If you delete or change some variable, the system generates an error during the project compilation. To correct the name of a variable declared automatically, follow the model exemplified above according to the device and requests to which it belongs.

By sending an application to the HX3040 CPU and putting it in *RUN*. Through the *ReqDiagnostics* GVL you can monitor the diagnostics variable values of the MODBUS Master/Client and DNP3 Client devices mappings, as shown it the figure below.

Expression	Type	Value
⊟ 🌐 MODBUS_Device_REQDG_0001	T_DIAG_MODBUS_RTU_MAPPING_1	
⊟ ◈ byStatus	T_DIAG_MODBUS_RTU_MAPPING_STATUS	
◈ bCommIdle	BOOL	FALSE
◈ bCommExecuting	BOOL	FALSE
◈ bCommPostponed	BOOL	FALSE
◈ bCommDisabled	BOOL	FALSE
◈ bCommOk	BOOL	FALSE
◈ bCommError	BOOL	FALSE
◈ bDiag_6_reserved	BOOL	FALSE
◈ bDiag_7_reserved	BOOL	FALSE
◈ eLastErrorCode	MASTER_ERROR_CODE	NO_ERROR
◈ eLastExceptionCode	MODBUS_EXCEPTION	NO_EXCEPTION
◈ byDiag_3_reserved	BYTE	0
◈ wCommCounter	WORD	0
◈ wCommErrorCounter	WORD	0
⊞ 🌐 MODBUS_Device_REQDG_0002	T_DIAG_MODBUS_RTU_MAPPING_1	
⊞ 🌐 MODBUS_Device_REQDG_0003	T_DIAG_MODBUS_RTU_MAPPING_1	
⊞ 🌐 MODBUS_Device_1_REQDG_0001	T_DIAG_MODBUS_ETH_MAPPING_1	
⊞ 🌐 MODBUS_Device_1_REQDG_0002	T_DIAG_MODBUS_ETH_MAPPING_1	
⊟ 🌐 Outstation_REQDG_0001	T_DIAG_DNP_CLIENT_REQUEST_1	
◈ eConnectionStatus	CONNECTION_STATUS	CONNECTING
◈ eRequestStatus	REQUEST_STATUS	SUCCESS
⊟ ◈ tIIN	T_DIAG_DNP_CLIENT_IIN_BITS	
◈ ALL_STATIONS	BOOL	FALSE
◈ CLASS_1_EVENTS	BOOL	FALSE
◈ CLASS_2_EVENTS	BOOL	FALSE
◈ CLASS_3_EVENTS	BOOL	FALSE
◈ NEED_TIME	BOOL	FALSE
◈ LOCAL_CONTROL	BOOL	FALSE
◈ DEVICE_TROUBLE	BOOL	FALSE
◈ DEVICE_RESTART	BOOL	FALSE
◈ NO_FUNC_CODE_SUPPORT	BOOL	FALSE
◈ OBJECT_UNKNOWN	BOOL	FALSE
◈ PARAMETER_ERROR	BOOL	FALSE

Figure 219: GVL ReqDiagnostics in Online Mode

6.4.5.9. GVL System_Diagnostics

The *System_Diagnostics* GVL declares the diagnostic variables of the CPU and communication drivers. This GVL is not editable, so the variables are automatically declared with the type specified by the device to which it belongs, when it is added to the project.

Example:

```
VAR_GLOBAL
   DG_HX3040           : T_DIAG_HX3040_1;
   DG_DNP3_Client         : T_DIAG_DNP_CLIENT_1;
   DG_MODBUS_Symbol_Client   : T_DIAG_MODBUS_ETH_CLIENT_1;
   DG_MODBUS_Symbol_RTU_Master : T_DIAG_MODBUS_RTU_MASTER_1;
END_VAR
```

By sending an application to the HX3040 CPU and putting it in *RUN*. Through GVL *System_Diagnostics* you can monitor the diagnostics variable values of the CPU and communication devices MODBUS and DNP3 Client as well, as shown in the figure below:

Expression	Type	Value
⊟ 🌐 DG_HX3040	T_DIAG_HX3040_1	
⊞ 🔷 tSummarized	T_DIAG_SUMMARIZED_1	
⊞ 🔷 tDetailed	T_DIAG_DETAILED_1	
⊞ 🌐 DG_DNP3_Client	T_DIAG_DNP_CLIENT_1	
⊟ 🌐 DG_MODBUS_Symbol_Client	T_DIAG_MODBUS_ETH_CLIENT_1	
⊞ 🔷 tDiag	T_DIAG_MODBUS_DIAGNOSTICS	
🔷 byDiag_1_reserved	BYTE	0
⊟ 🔷 tCommand	T_DIAG_MODBUS_COMMANDS	
🔷 bStop	BOOL	FALSE
🔷 bRestart	BOOL	FALSE
🔷 bResetCounter	BOOL	FALSE
🔷 bDiag_19_reserved	BOOL	FALSE
🔷 bDiag_20_reserved	BOOL	FALSE
🔷 bDiag_21_reserved	BOOL	FALSE
🔷 bDiag_22_reserved	BOOL	FALSE
🔷 bDiag_23_reserved	BOOL	FALSE
🔷 byDiag_3_reserved	BYTE	0
⊞ 🔷 tStat	T_DIAG_MODBUS_ETH_CLIENT_STATS	
⊟ 🌐 DG_MODBUS_Symbol_RTU_Master	T_DIAG_MODBUS_RTU_MASTER_1	
⊞ 🔷 tDiag	T_DIAG_MODBUS_DIAGNOSTICS	
🔷 eErrorCode	SERIAL_STATUS	NO_ERROR
⊞ 🔷 tCommand	T_DIAG_MODBUS_COMMANDS	
🔷 byDiag_03_reserved	BYTE	0
⊟ 🔷 tStat	T_DIAG_MODBUS_RTU_MASTER_STATS	
🔷 wTXRequsts	WORD	0
🔷 wRXNormalResponses	WORD	0
🔷 wRXExceptionResponses	WORD	0
🔷 wRXIllegalResponses	WORD	0
🔷 wRXOverrunErrors	WORD	0
🔷 wRXIncompleteFrames	WORD	0
🔷 wCTSTimeOutErrors	WORD	0
🔷 wDiag_18_reserved	WORD	0

Figure 220: GVL System_Diagnostics in Online Mode

6.4.6. ProtTask Configurations

The *"ProtTask"* is responsible for executing a single programming unit of type *"Program"*, whose name is *"ProtPrg"*. This, in turn, can call other programming units and is responsible for executing the project protection routines. The *"ProtTask"* is a cyclic task with priority set as 1.

6.4.6.1. UserProtPrg Program

This POU is performed only in the Active CPU and is designed to control the user process. *"UserProtPrg"* is created only in projects with CPU redundancy and its priority is higher when compared to *"UserPrg"*.

6.4.6.2. NonSkippedProtPrg Program

This POU is similar to *"NonSkippedPrg"*, except that it features a higher priority. The *"NonSkippedProtPrg"* is created only in projects with CPU redundancy.

6.4.7. GVLs with Redundant Symbolic Variables

The user can create other GVLs different from the previously listed, in order to declare redundant symbolic variables. For that, after the GVL creation, it's necessary to mark it in the object *Redundancy Configuration*, in the project devices tree. By default, all GVLs created by the user are, initially, redundant.

> **ATTENTION**
>
> For good practice it's recommended to avoid the AT directive use in GVLs which have redundant symbolic variables declaration to prevent variable mapping in non-redundant areas.

6.4.8. Program POUs with Redundant Symbolic Variables

The user can declare redundant symbolic variables in POUs from the program type, with exception of the *NonSkippedPrg* and *NonSkippedProtPrg* POUs where the symbolic variables declared are considered redundant.

In order to define a new POU as redundant, it must be marked in the *Redundancy Configuration* object after its creation, in the project devices tree. By default, all POUs created by the user are, initially, redundant.

> **ATTENTION**
>
> For good practice it's recommended to avoid the AT directive use in POUs which have redundant symbolic variables declaration to prevent variable mapping in non-redundant areas.

6.4.9. Breakpoints Use in Redundant Systems

For redundant systems, you can only use breakpoints in the Active CPU if the other one is outside the bus or it is in Inactive state. MasterTool blocks the operation if it meets this condition.

6.4.10. Limitations in the Programming of a Redundant CPU

Redundant CPUs present some limitations regarding the programming. The following subsections discuss such restrictions.

6.4.10.1. Limitations in GVLs and Redundant POUs

For GVLs or redundant program POUs make sure to attend the following rules for the correct operation of the CPUs:

- Do not use VAR_TEMP variables.
- Do not mix several variable types (VAR, VAR RETAIN, VAR PERSISTENT, VAR CONSTANT etc.) The GVL or POU must contain one kind in each.
- Do not mix declaration of symbolic variables with ATs in GVLs. Create separate GVLs declaring AT variables in the first one and the symbolic variables in the second one.
- Do not store the address of a variable in a redundant variable (use a redundant variable as a pointer to an address), as the addresses of the variables may be different in CPUA and CPUB.
- Do not use the function blocks for RTC reading and writing in redundant POUs. More details can be found on the section RTC Clock

6.4.10.2. Limitations in the Non-Redundant Program (StartPrg, NonSkippedPrg and NonSkippedProtPrg)

For non-redundant POUs *StartPrg, NonSkippedPrg* and *NonSkippedProtPrg*, make sure to attend the following rules for the correct operation of the CPUs:

- Do not use TON and TOF traditional function blocks, as they use the IEC timer. When the Standby CPU goes into Active state (while the other CPU exits the Active state), the IEC timer will be synchronized, causing a discontinuity in the timer value. Preferably, use *TON_NR* and *TOF_NR* functional blocks, which are available in *NextoStandard* library. See section Non-Redundant Timer.
- Do not use program POUs written in SFC (Sequential Function Chart), as they use the IEC timer for transitions timing.
- Do not mix declaration of symbolic variables with ATs in GVLs. Create separate GVLs declaring AT variables in the first one and the symbolic variables in the second one.

6.4.11. Getting the Redundancy State of a CPU

You can check the redundancy status of a CPU from the Redundancy Diagnostics Structure.

```
VAR
  eRedStateLocal : REDUNDANCY_STATE;
END_VAR

eRedStateLocal := DG_HX3040_01.RedDgnLoc.RedundancyDiagnostics.eRedState;
```

altus

Thus, the user can control the logics that depend on the CPU redundancy state. The amount received is a numeric value given in byte. The table below indicates the meanings of all possible states:

eRedState	
NOT_CONFIGURED	0
STARTING	2
STANDBY	3
ACTIVE	4
INACTIVE	5
NOT_AVAILABLE	6

Table 220: Redundancy States

6.4.12. Non-Redundant Diagnostics Reading

A redundant project, besides presenting redundant diagnostics (Redundancy Diagnostics Structure and diagnosis of bus modules), also features non-redundant diagnostics (HX3040 CPU specific diagnostics). These diagnostics are not exchanged between the CPUs. For example, if you want to know the link status of the Ethernet ports of the other CPU in the program, use the available bytes in the user data structure of the redundancy diagnostics. See User Information Exchanged among CPUA and CPUB.

6.5. Programs Load in a Redundant CPU

The Redundant CPU Programming section dealt with issues related to the project development of a redundant CPU with HX3040 CPU.

This section brings the requested methods and steps for loading this project into a redundant CPU, considering situations such as:

- Load the project into a brand new HX3040 CPU, or into a CPU containing an unknown project.
- Online changes without process control interruption.
- Offline changes with process control interruption during a programmed system shutdown.

6.5.1. Initial Load of a Redundant Project

This section describes the steps required to make the first load of a redundant project into the HX3040 CPU. This procedure is necessary, for instance, for a new CPU or for a CPU containing an unknown project.

> **ATTENTION**
>
> Perform the following steps for only one of the two CPUs (CPUA and CPUB) that comprise the redundant system. Introduce the second CPU in the bus and it will automatically get synchronized to the operating CPU, as long it is in Active state.

6.5.1.1. Step 1 – Find out the IP Address for MasterTool Connection

The first step is to find out the IP address of the NET 1 channel of this CPU, for connection to MasterTool.

This must be done through the display and button of the HX3040 CPU, as described in Informative Menu and of CPU's Configuration. The NETWORK menu reports the IP addresses of all NET ports of the HX3040 CPU, especially NET1 since it is the only one used for communication with MasterTool.

6.5.1.2. Step 2 – Check Network and Computer's IP for Programming

So that all the next stages may be followed, there is the need for the computer that will program and send the project to the UCP HX3040 to be on the same ethernet subnet and to have a different and unique IP than the one that will be configured for the UCP.

6.5.1.3. Step 3 – Check IP Addresses Conflict

Before performing the next step, make sure that there is no other equipment on the network with the same IP address discovered in the previous steps. The user must initially verify that the Windows firewall is disabled, as instructed on Microsoft's official website (http://windows.microsoft.com/pt-br/windows/turn-windows-firewall-on-off#turn-windows-firewall-on-off= windows-7), some operating systems block these messages by default and require special care. Once this is done, the IP can be discovered, for example by disconnecting the CPU from the network and running a *"ping"* command on its IP address. Since the CPU is disconnected, this *"ping"* is expected to fail. If the *"ping"* responds, there is another device with the same IP address.

If the IP address is already in use by another device on the network, the next and some following steps must be performed using a crossover cable to directly connect the PC running the MasterTool Xtorm software to the CPU, thus avoiding IP address conflicts. In one of the next steps, when loading the project on the CPU, the definitive IP addresses of the CPU will be updated (see section HX3040 CPU Ethernet Ports Configuration (NET 1 to NET 6)).

6.5.1.4. Step 4 – Prepare Connection to MasterTool (Set Active Path)

Double-click the Device (HX3040) in the device tree and access the *"Communication Settings"* tab. Then, click on the Gateway, and press the *"Map Network"* button to list all the CPUs detected by the MasterTool on the network.

At this point, it is expected to find a list of CPs whose ID contains the IP address discovered in the first step and the type of the CPU (HX3040). If the user has previously changed the name of the CPU on the network, this name will be the one displayed at this point. The section Finding the Network describes in more detail the possible IDs that may be seen in this list, as well as the selection and sending of the project.

6.5.1.5. Step 5 – Load of the Redundant Project

This step describes the load of the redundant project on the CPU. This project should be prepared as described in section Redundant CPU Programming.

A basic redundant project can be prepared according to the following subsections of this section:

- Wizard for a New Redundant Project Creation
- HX3040 CPU Ethernet Ports Configuration (NET 1 to NET 6)

Obviously, it is also possible to make a complete redundant project and then load it into CPUA and CPUB, for example, if the hardware of these CPUs are not available during development of the MasterTool project.

The first load of a redundant project on a CPU must still be made using that IP address discovered in the first step of this procedure, and selected in the steps that follow this procedure.

The project load is made through the menu Online/Login.

Note:

After sending the project to one of the CPUs, the other one will download the project automatically by means of internal sync channel.

ATTENTION

Within the project developed with the MasterTool and loaded on the CPU in this step, it was defined new IP addresses for the interface NET 1 of CPUA and CPUB, as well as an IP address for the interface NET of CPU 1 active (Active IP address) – see section HX3040 CPU Ethernet Ports Configuration (NET 1 to NET 6).

Therefore, after this initial charge, that IP address discovered in the first step of this procedure normally is no longer valid. This change of IP address on NET 1 will cause a loss of connection of MasterTool with CPU, which will be notified, only the NET1 can be used to communicate with the CPU via MasterTool.

For more details about how to reconnect the MasterTool see MasterTool Connection with a HX3040 CPU from a Redundant CPU.

6.5.2. MasterTool Connection with a HX3040 CPU from a Redundant CPU

After performing the procedure described in section Initial Load of a Redundant Project on the two CPUs (CPUA and CPUB), the connection to the MasterTool, through the NET 1 interface of the HX3040 CPU, can be made through one of the following addresses:

---

1. CPUA IP address: NET 1 address exclusively for CPUA.
2. CPUB IP address: NET 1 address exclusively for CPUB.

Regardless of the State of the CPU, the MasterTool can only connect to it using the unique address NET1 of the CPU, configured at IP address of CPUx. However, if the CPU is in the active state, all other services will be able to connect to the CPU by both the IP address of CPUx and the active IP address.

To connect to a particular CPU, the user must double-click on the Device (HX3040) in the device tree, enter the tab *"Communication Settings"*, click on the Gateway, and press the button *"Map Network"* to list all PLCs detected by MasterTool on the network.

In this list, the user can find the following default IDs, if the CPU name on the network has not been changed previously by the user:

1. *HX3040_<IP address>_PLCA:* CPUA identification. In this case, the field *<IP address>* must match the IP address of CPUA configured in the project.
2. *HX3040_<IP address>_PLCB:* CPUB identification. In this case, the field *<IP address>* must match the IP address of CPUB configured in the project.

Next, the user must select the CPU on this list where the MasterTool should connect, and press the button *"Set Active Path"*. Later, when running the command from the menu *Online -> Login*, the MasterTool connects to this CPU.

> **ATTENTION**
>
> The MasterTool can only connect to one CPU at a time. To connect to various CPUs, multiple instances of the MasterTool must be opened, taking care always to open the correct project in each instance, for example, connecting to CPUA in an instance of the MasterTool, opening the same project and connecting the CPUB. Changes should not be made on the condition of multiple instances of the same project open.

6.5.3. Load of Changes to a Redundant Project

After the two CPUs (CPUA and CPUB) that compose the redundant system have already received an initial load, as described in section Initial Load of a Redundant Project, it is possible to load subsequent modifications of the project, as such modifications are required.

MasterTool's connection to CPUs to run the load of modifications must be made as described in section MasterTool Connection with a HX3040 CPU from a Redundant CPU. This section explains how the user can connect to a specific CPU (CPUA or CPUB), or to the active/non-active CPU.

The changes should always be loaded into Active CPU, which will transfer them automatically to the non-active CPU, through the channels of synchronism. Therefore, the MasterTool typically must use the unique IP address of CPU that is in the active state (IP address of the CPUx), to connect to port NET 1 of the HX3040 Active CPU.

> **ATTENTION**
>
> It is not possible to load a project on non-active CPU, while the other CPU is in the active state. The user can only upload a project or perform modifications online in a non-active CPU if the other CPU is not in active state (typically in the non-configured or inactive state) or if it is missing in the bus. Similarly, the user can only load a project on Active CPU, if the other CPU is in the inactive state or is not present on the bus.

6.5.4. Load of Changes in Offline and Online Mode

Project modifications can be loaded offline or online.

Offline loads require CPU in stop mode where the modification must be loaded. On the other hand, online charges allow the CPU to continue running user application while the modification is loaded.

Some types of modifications require offline load, i.e. cannot be uploaded online on the CPU where the MasterTool is connected. See Offline Load of Changes with Interruption of Process Control section for more details.

6.5.4.1.   Modifications that Require Offline Load with Interruption of the Process Control

The following modifications in a project cannot be loaded in a redundant system without interrupting the process control:

- Add or remove devices in the device tree, for example:
  - I/O modules;
  - Instances of communication protocols such as MODBUS, DNP3, etc.
- Modification of device parameters in the device tree, for example:
  - IP addresses and other parameters of Ethernet interfaces;
  - Parameters of I/O modules;
  - Instantiated protocols parameters, mappings, etc.
- Changes in the tasks settings.

6.5.4.2.   Modifications that allow Online Load

In principle, the modifications not referred in section Modifications that Require Offline Load with Interruption of the Process Control, allow online load.

Even so, the main modifications that allow online load on the CPU where the MasterTool is connected are listed below. The modifications mentioned below apply to variables, POUs and GVLs, redundant or not.

- Add program type POUs, since these POUs do not need to be associated with any task.
- Remove program type POUs, as long as these POUs are not associated with any task.
- Add or remove function or function block type POUs.
- Modify the code of any POU (program, function or function block).
- Add or remove symbolic variables in any POU (program, function or function block, being them redundant or not).
- Add or remove instances of function block in program or function block type POUs.
- Add or remove GVLs.
- Add or remove symbolic variables or instances of function block in GVLs.

There are two methods of online load, described below, one for compiling and sending the Initialization Application done manually and the other automatically after sending the online load. For more information, the MasterTool IEC XE User Manual MT8500 (MU299048) should be consulted.

6.5.5.   Load of Changes in Online Mode

In the section Load of Changes in Offline and Online Mode, were described modifications that require offline load and those that allow online load.

An online load must be made by connecting the MasterTool to NET channel 1 of the Active CPU using its unique IP address (UCPA IP or UCPB IP). It is necessary to change the redundancy state of the Standby CPU to Inactive before sending the Online modification to the Active CPU, otherwise the Mastertool will block the operation. After sending the modifications, the application is automatically created in the non-volatile memory of the CPU, and there is no risk of losing the modifications in case the CPU is switched off or in the event of a power failure.

After finished online loading process, the CPU can exit inactive state, for example, by using the command *"Switch to Standby"* in the display menu. This will cause the CPU transition to the non-configured state. The CPU will remain in the non-configured state until the process of automatic synchronization of projects finishes. After that, the CPU goes to the initializing state and then returns to the Standby state or goes back to the inactive state if a failure occurs.

6.5.6.   Offline Load of Changes with Interruption of Process Control

In this section, it is defined the procedure for performing an offline loading that interrupts the process control. Such a situation is acceptable in certain types of cases and during process shutdown.

An offline load of this type must be made by connecting the MasterTool to Active CPU NET 1 channel, using the unique address of the CPU in the active state (CPUx IP address). Before starting an offline loading on the Active CPU, the user must pass the Standby CPU to the Inactive State. If this is not done, the MasterTool will not allow this change.

To run offline load, it is necessary that Standby CPU goes to the inactive state, preventing an accidental program load.

At the time that offline load starts, the Active CPU goes to the non Configured state.

When the offline load finishes, the user can restart program execution on the CPU where the application has been loaded (switch to RUN mode). After a few seconds, this CPU returns to the active state.

After this CPU returns to the active state, the user can take the other CPU of inactive state, for example, by using the command *"Switch to Standby"* in display menu. This will cause the CPU transition to the non-configured state. The CPU will remain in the non configured state until the process of automatic synchronization of projects finishes. After that, the CPU goes to the initializing state and then returns to the Standby state.

6.6. Redundant RTUs Maintenance

6.6.1. MasterTool Warning Messages

When the MasterTool is making a connection or is already connected to a CPU with redundant project, some special warning messages may occur as described in the next subsections.

6.6.1.1. Blocking Before Commands that can Stop the Active CPU

Some commands, such as the following, can stop a CPU:

- Offline load after Online / Login
- Debug / Stop
- Debug / New Breakpoint
- Online / Reset (hot, cold, origin)

These commands will only be triggered if the Active CPU is operating with redundancy disarmed, i.e. another CPU must be in the Inactive State (or out of the bus). Otherwise, a blocking message appears:

"The Login cannot be performed, because the current project is different from the project in use on the CPU. If you really want to perform this operation, configure another CPU to the inactive state and then rerun this command".

6.6.1.2. Blocking of Operations in non-Active CPU

In some circumstances, certain operations are locked in non-active CPU. If another CPU is in the active state, which is usual in the case of redundancy, certain operations are blocked on non-active CPU:

- Offline and online load
- Switch application state (RUN/STOP)
- Debug / Stop
- Debug / New Breakpoint

When there is an attempt of executing a command of this type, the MasterTool sends the following message:

"This operation can only be performed on another CPU, because it is controlling the process."

On the other hand, it is possible to login on non-active CPU to monitor or force non-redundant variables.

6.6.2. Interaction with the Redundancy through the HX3040 CPU Graphics Display

The redundancy state and menu available operations can be accessed on the HX3040 CPU display.

6.6.2.1. CPU Redundancy State

The CPU redundancy state, described in Redundant CPU States, is seen in the three characters that start from the second row of the main screen, as shown in the Graphic Display section. The screen is presented after the initialization of the CPU, and a few seconds after any navigation (without pushing the HX3040 CPU button).

6.6.2.2. Screens Below the Redundancy Menu

There is a menu called *REDUNDANCY*, below which there are some screens. The description and access to screens of redundancy are available in section Informative Menu and of CPU's Configuration.

6.6.3.   Redundancy Diagnostics Structure

The redundancy diagnostics area corresponds to the *DG_HX3040_01*symbolic variable, allocated automatically by MasterTool on GVL *System_Diagnostics*.

- This area is divided into six other data structures:
  - *RedDgnLoc:* contains local CPU redundancy diagnostics (where the MasterTool is connected), as for example, the CPU redundancy state. This section is described in Redundancy Diagnostics.
  - *RedDgnRem:* is a copy of RedDgnLoc from another CPU, received via sync channels. In this way, the CPU has access to remote CPU diagnostics. This section is described in Redundancy Diagnostics.
  - *RedCmdLoc:* contains redundancy commands generated on this CPU (local), for example, from a SCADA system, or generated in POUs of this CPU (ex: UserPrg or NonSkippedPrg). This section is described in Redundancy Commands.
  - *RedCmdRem:* this is a copy of RedCmdLoc from another CPU (remote), received via sync channels. This section is described in Redundancy Commands.
  - *RedUsrLoc:* CPU 128-byte memory area, which will be transferred to the RedUsrRem data structure of the remote CPU. Is used for the user to exchange information between CPUA and CPUB. This section is described in User Information Exchanged among CPUA and CPUB.
  - *RedUsrRem:* CPU 128-byte memory area, which corresponds to the RedUsrLoc data structure of the remote CPU. Is used for the user to exchange information between CPUA and CPUB. This section is described in User Information Exchanged among CPUA and CPUB.

It is important to note that the redundancy diagnostics structures of remote CPU are updated only when data synchronization occurs successfully. Therefore, while a new synchronization does not occur, the diagnostics will remain with the frozen value, corresponding to the last successful data exchange.

In addition, the remote CPU structures are read-only, that is, values written in these structures will be overwritten without being considered, the next data synchronization. Therefore, you cannot use the *RedCmdRem* structure to execute a command on remote CPU. The structure used to execute commands, to be written, should always be *RedCmdLoc*.

6.6.3.1.   Redundancy Diagnostics

The diagnoses of redundancy can have multiple uses. Among them:

- Can be used to verify the existence of a problem that needs to be remedied.
- Can be used to query the status of the system, as well as actions to be taken in case of failures or variations of diagnostic values. The main events related to redundancy system diagnoses are presented as events in system logs. Referring to the historical sequence of such events can find out, for example, the cause of a switchover.

> **ATTENTION**
>
> The diagnosis *DG_HX3040_01.RedDgnLoc.sGeneral_Diag.bExchangeSync* (defined next) should be tested to see if the *RedDgnRem* data structure has been read successfully from remote CPU in the last cycle of MainTask. If the value of this diagnosis is FALSE, this means that the *RedDgnRem* data structure was not read successfully from remote CPU and therefore the *RedDgnRem* values may be invalid or outdated.

As RedDgnRem is a copy of RedDgnLoc from another CPU, both structures have the same format. These are still divided into four substructures:

- *RedundancyDiagnostics:* general diagnostics of redundancy.
- *SyncLinkDiags[1]:* A sync channel diagnostics.
- *SyncLinkDiags[2]:* B sync channel diagnostics.
- *SyncLinkStatistics:* common statistics for sync channels A and B, for count of successes and failures of synchronization services.

The substructure *"RedundancyDiagnostics"* has the following fields for general diagnostics of redundancy:

.RedundancyDiagnostics.*	Type	Description
bConfigDone	BOOL	**TRUE –** The configuration process, run on non-configured state, has finished. **FALSE –** The configuration process, run on non-configured state, still not ended or was not run.
bConfigError	BOOL	**TRUE –** The configuration process, run on non-configured, finished with errors. This is a system error, not usually expected. Contact Altus support for reporting it. Notify also the diagnostic value of ConfigErrorCode to Altus support. **FALSE –** The configuration process successfully occurred or was not performed.
bTooManyRedAreas	BOOL	**TRUE –** The number of redundant areas has exceeded the maximum number allowed. This is a system error, not usually expected. Contact Altus support for reporting it. **FALSE –** The number of redundant areas are within the expected.
bTemporaryBufferTooSmall	BOOL	**TRUE –** Intermediate data structure with insufficient size. This is a system error, not usually expected. Contact Altus support for reporting it. **FALSE –** The size of the data structure is within the expected.
bIncompatibleApplication	BOOL	**TRUE –** The application is not compatible between the two PLCs. Was performed a new application download to one of the CPU with one of the following changes: Modification of redundant data area; Modification of redundant symbolic variables. While this diagnosis is connected, one of the PLCs will be in inactive state until the same application is present on both PLCs. This implies reload the former application in both PLCs or update the two PLCs with the new application. For more information about how to proceed, consult Programs Load in a Redundant CPU. **FALSE –** The current application on both CPUs is compatible, i.e. the same.
bIncompatibleFirmware	BOOL	**TRUE –** This CPU was switched to the non-configured state as its firmware version is incompatible with the Active CPU firmware version. **FALSE –** The Active CPU firmware version is compatible with the firmware version of the non-active CPU.
bExchangeSync	BOOL	**TRUE –** The Diagnostics and Commands Exchange synchronization service was executed successfully in this cycle of MainTask. **FALSE –** The RedDgnRem structure has obsolete or invalid values, because it was not read from another CPU (remote) in this cycle of MainTask.
bRedDataSync	BOOL	**TRUE –** The Redundant Data Synchronization service was successfully executed in this cycle of MainTask. **FALSE –** The Redundant Data Synchronization service has not run successfully in this cycle of MainTask

.RedundancyDiagnostics.*	Type	Description
bApplicationProjectDiff	BOOL	**TRUE –** The application project of this CPU is different from that one present on another CPU. Active CPU. **FALSE –** The application project of this CPU is equal to that of another CPU.
bProjectArchiveDiff	BOOL	**TRUE –** The Project archive of this CPU is different from that one present on another CPU. **FALSE –** The Project archive of this CPU is equal to that of another CPU.
bOnlineChangeApply	BOOL	**TRUE –** An online change in the application was held and this has not yet been synchronized with the Standby CPU. **FALSE –** No online changes were carried out in the application or these have already been synchronized with the Standby CPU.
bBusError	BOOL	**TRUE –** Detected a failure in the bus access. **FALSE –** There is no fault in the bus access.
bBusIdle	BOOL	**TRUE –** There's no bus activity detected by the CPU (working in passive mode). **FALSE –** There is bus activity detected by the CPU.
eCPU_ID	ENUM (BYTE)	This diagnostic reports on the identification of this CPU: - 0 = non-redundant - 2 = CPUA - 3 = CPUB This is a copy of the CPU ID, as described in section Identification of a HX3040 CPU.
eRedState	ENUM (BYTE)	Reports on the state of this CPU redundancy: - non-Configured = 0 - Initializing = 2 - Stand-by = 3 - Active = 4 - Inactive = 5
ePreviousRedState	ENUM (BYTE)	Value that RedState diagnosis had before the final transition of states.
eStateChangeReason	ENUM (BYTE)	Numeric code that indicates the reason why there were redundancy state change, this information is available in table 224.
eAppState	ENUM (BYTE)	Application Mode: - Unknown = 0 - Run = 1 - Stop = 2 - Breakpoint = 3
wRedStateDuration	WORD	Measure how long (milliseconds) the current redundancy state was assumed. This time stop increasing as it reaches 65535 ms.
dwApplicationCRC	DWORD	32-bit CRC of the Application Project, used to detect differences between the Application Projects of both CPUs.
dwArchiveCRC	DWORD	32-bit CRC of the Project Archive, used to detect differences between the Project Archives of both CPUs.
dwFirmwareVersion	DWORD	CPU firmware version, used to check compatibility between firmware of the two CPUs.

altus

.RedundancyDiagnostics.*	Type	Description
dwIECTimer	dwIECTimer	Synchronization of the *IEC Timer* is necessary for *bumpless* operation of some function blocks such as TON and TOF. Through this diagnostic the *IEC Timer* of the Active CPU is received and updated in the Non-Active CPU, provided the Diagnostic and Command Exchange service has been successfully executed. Its count starts at 0 and increments to 4294967295. After count overflow, it restarts with a value of 0.
wCycleCounter	WORD	16-bit counter used as auxiliary sequence information in the *Redundancy Event Logs*. In the Active CPU, it is incremented at each MainTask cycle. On the Non-Active CPU, it receives a copy from the Active CPU, provided that the Diagnostic and Command Exchange service has been successfully executed. Its count starts at 0 and increments up to 65535. After count overflow, it restarts with a value of 0.

Table 221: General Diagnostics of Redundancy

Notes:

Visualization of Diagnostics Structures: The diagnostic structures added to the project can be viewed by accessing the *"Library Manager"* item in the MasterTool Xtorm window treeview. With this, the user can view all the data types defined in the structure.

The substructure *"SyncLinkDiags"* is an array of two elements and has the following fields for diagnostics of both sync channels (array 1 and 2):

.SyncLinkDiags[1..2].*	Type	Description
bGeneralFailure	BOOL	**TRUE –** The sync channel has some type of failure. The next 3 diagnostics indicate the specific failure. **FALSE –** The sync channel is in correct operation.
bInternalFailure	BOOL	**TRUE –** The fault detected has its cause into this CPU. Such failures are handled differently. **FALSE –** No internal failure.
bLinkDownFailure	BOOL	**TRUE –** There is no link between the CPUs. **FALSE –** The link is operational.
bTimeoutFailure	BOOL	**TRUE –** This fault is reported if a synchronization service has not completed successfully until a time-out specified, and have not been found faults of type bInternalFailure or bLinkDownFailure that would justify this. **FALSE –** No time-out.

Table 222: Specific diagnoses of sync links

The substructure *"SyncLinkStatistics"* contains statistics of failures and successes of the services. Statistics relating to local and remote CPUs can be reinitialized using the commands:

```
//Local CPU
DG_HX3040_01.RedCmdLoc.bResetNETStatisticsLocal := TRUE;
//Remote CPU
DG_HX3040_01.RedCmdLoc.bResetNETStatisticsRemote := TRUE;
```

altus

.SyncLinkStatistics.*	Type	Description
wSuccessExchDgCmdSync	WORD	Count of hits of Diagnostics and Commands Exchange service. Its count starts with the value 0, increments up to 65535, and then reboots again with the value 0 when the limit value is exceeded.
wFailedExchDgCmdSync	WORD	Failures count of Diagnostics and Commands Exchange service. Its count starts with the value 0, increments up to 65535, and then reboots again with the value 0 when the limit value is exceeded.
wSuccessRedDataSync	WORD	Count of hits of the Redundant Data Synchronization service. Its count starts with the value 0, increments up to 65535, and then reboots again with the value 0 when the limit value is exceeded.
wFailedRedDataSync	WORD	Failures count of the Redundant Data Synchronization service. Its count starts with the value 0, increments up to 65535, and then reboots again with the value 0 when the limit value is exceeded

Table 223: Interface Specific Diagnostics

The table below shows all messages to be presented to the states and changes of redundancy, accessible to the user by means of diagnosis "DG_HX3040_01.RedDgnLoc.RedundancyDiagnostics.eStateChangeReason.". Messages presented in the description are identical to the messages that will be displayed in the Log. For more information about Log, see the Redundancy Event Logs. For more information about the redundancy states see Transition between Redundancy States.

State String	State Value	Log Message	Description
INIT_APP	0	Redundancy initialization started.	Redundancy process was initiated.
INIT_DONE	1	Redundancy initialization done.	Redundancy process completed.
INIT_REDUNDANT _DATA_OVER _MAXIMUM	2	Error: Redundant data is too big.	Area destined to redundancy exceeds the maximum threshold.
INIT_CLUCONF_ERROR	3	CPU is not inserted in the right position for a redundant setup.	The CPU is not inserted in the correct position for a redundant configuration.
NCF_LOCAL_CPU _USER_CMD	100	User state change requested via local diagnostic command bit.	Not configured for other states, state exchange command via diagnostic local bit
NCF_REMOTE_CPU _USER_CMD	101	User state change requested via remote diagnostic command bit.	Not configured for other states, state exchange command via diagnostic remote bit.
NCF_LCD_MENU _USER_CMD	102	User state change requested via LCD menu.	Not configured for other states, state exchange command via LCD menu.
NCF_AUTO_ CONFIGURATION	103	Auto configuration.	Not configured for other states, auto configuration of redundancy.
STR_STANDBY	200	Other CPU is controlling the system.	Initializing to Standby, the other CPU is controlling the system.
STR_ACTIVE	201	CPU started controlling the system.	Initializing to Active, CPU started controlling the Active system.
STR_CONFIG_ERROR	202	The Application has a configuration error.	Initializing to Non-Configured, the application has configuration error.

altus

State String	State Value	Log Message	Description
STR_LINK_WITH_INTERNAL_FAILURE	203	Redundant link with an internal failure.	Initializing to Inactive, internal redundancy channel failed.
STR_NO_COMM_OTHER_CPU_AND_BUS_IDLE	204	This CPU is unable to communicate with the other CPU via redundant link and the BUS is idle.	Initializing to Active, this CPU is unable to communicate with the other CPU via redundancy channel and the bus is stopped.
STR_NO_COMM_OTHER_CPU_AND_BUS_BUSY	205	This CPU is unable to communicate with the other CPU via redundant link and the BUS is busy.	Initializing to Inactive, this CPU is unable to communicate with the other CPU via redundancy channel and the bus is in use.
STR_INCOMPATIBLE_FIRMWARE	206	The firmware version of this CPU is not compatible with the other one.	Initializing to Inactive, the firmware version of this CPU is not compatible with the other one.
STR_APPLICATION_DIFF	207	The Application between the CPUs is different.	Initializing to Non-Configured, the application in the CPUs are different.
STR_ARCHIVE_DIFF	208	The Project Archive between the CPUs is different.	Initializing to Non-Configured, the CPUs Project Archive are different.
STR_ONLINE_CHANGE_APPLY	209	There is an online change in the Application of the Active CPU.	Initializing to Non-Configured, an online load of the application was performed on the Active CPU.
STR_DATA_EXCHANGE_FAILED	210	Redundant data synchronization failed.	Initializing to Inactive, redundant data sync failed.
STR_APP_STATE_DIFF_WITH_OTHER_ACT	211	The application state of this CPU is different from the other Active CPU.	Initializing to Non-Configured, the application state of this CPU is different from another Active CPU.
STR_APP_STATE_DIFF_WITH_OTHER_NON_ACT	212	The application state of this CPU is different from the other Non-Active CPU.	Initializing to Active, the application state of this CPU is different from another Inactive CPU.
STR_BUS_ERROR	213	Bus error detected.	Initializing to Inactive, bus error detected.
STR_OTHER_CPU_ACT_WITH_BRKP	214	Other CPU is Active and with Application stopped in a Breakpoint.	Initializing to Inactive, the other CPU is in Active and with the application stop on Breakpoint.
SBY_LINK_WITH_INTERNAL_FAILURE	301	Redundant link with an internal failure.	Standby to Active, redundancy channel with internal failure.
SBY_APPLICATION_DIFF	302	The Application between the CPUs is different.	Standby to Non-Configured, the application in the CPUs are different.
SBY_ARCHIVE_DIFF	303	The Project Archive between the CPUs is different.	Standby to Non-Configured, the CPUs Project Archive are different.
SBY_ONLINE_CHANGE_APPLY	304	There is an online change in the Application of the Active CPU.	Standby to Non-Configured, an online load of the application was performed on the Active CPU.
SBY_DATA_EXCHANGE_FAILED	305	Redundant data synchronization failed.	Standby to Inactive, redundant data sync failed.
SBY_VITAL_FAILURE	306	The other CPU wasn't Active	Standby to Inactive, the other CPU wasn't active.
SBY_OTHER_PLC_NON_ACTIVE	307		Standby to Active, the other CPU wasn't active.
SBY_DIAGNOSTICS_FAILED	308	Exchange of Redundancy diagnostics failed	Standby to Inactive, the other CPU wasn't active.
SBY_BUS_ERROR	309	Bus error detected	Standby to Inactive, bus error detected: the other CPU is active.

State String	State Value	Log Message	Description
SBY_BUS_IDLE	310	BUS in idle state	Standby to Active, the other CPU wasn't active.
SBY_APP_STATE_DIFF _WITH_OTHER_ACT	311	The application state of this CPU is different from the other Active CPU	Standby to Inactive, the application state of this CPU is different from another Active CPU.
SBY_LOCAL_CPU _USER_CMD	312	User state change requested via local diagnostic command bit	Other States to Standby, State Exchange Command via local diagnosis bit.
SBY_REMOTE_CPU _USER_CMD	313	User state change requested via remote diagnostic command bit	Other States to Standby, State Exchange Command via remote diagnosis bit.
SBY_LCD_MENU_USER _CMD	314	User state change requested via LCD menu	Other States to Standby, State Exchange Command via LCD menu.
SBY_OTHER_CPU_ACT _WITH_BRKP	315	Other CPU is Active and with Application stopped in a Breakpoint	Standby, the other CPU is in active and with the application stopped on Breakpoint.
INA_LOCAL_CPU_USER _CMD	401	User state change requested via local diagnostic command bit	Other States to Inactive, State Exchange Command via local diagnosis bit.
INA_REMOTE_CPU _USER_CMD	402	User state change requested via remote diagnostic command bit	Other States to Inactive, State Exchange Command via remote diagnosis bit.
INA_LCD_MENU_USER _CMD	403	User state change requested via LCD menu	Other States to Inactive, State Exchange Command via LCD menu.
ACT_LINK_WITH _INTERNAL_FAILURE	501	Redundancy internal link failure for one link	Redundancy internal channel error.
ACT_BOTH_LINKS _WITH_INTERNAL _FAILURE	502	Redundancy internal link failure for both links	There was error in both redundancy internal channels.
ACT_CONFLICT _ACTIVE	503	Both CPUs are in active redundancy state	Active to Standby, both CPUs are active.
ACT_VITAL_FAILURE	504	Both CPUs are in active redundancy state and Application state is equal	Active to Inactive, both CPUs are in active and with the application in the same state.
ACT_CONFLICT _ACTIVE_APP_STATE _EQUAL	505		Active to Standby, both CPUs are active and with the application in the same state.
ACT_CONFLICT _ACTIVE_APP_STATE _DIFF	506	Both CPUs are in active redundancy state and Application state is different	
ACT_BUS_ERROR	507	Bus error detected	Active to Inactive, bus error detected, the other CPU is in Standby.
ACT_LOCAL_CPU _USER_CMD	508	User state change requested via local diagnostic command bit	Other States to Inactive, State Exchange Command via local diagnosis bit.
ACT_REMOTE_CPU _USER_CMD	509	User state change requested via remote diagnostic command bit	Other States to Inactive, State Exchange Command via remote diagnosis bit.
ACT_LCD_MENU _USER_CMD	510	User state change requested via LCD menu	Other States to Inactive, State Exchange Command via LCD menu.
ACT_APP_DELETE	511	Application of this CPU was deleted	The application of this CPU was deleted

altus

State String	State Value	Log Message	Description
ACT_APP_RESET	512	Application of this CPU was reset	The application of this CPU was reset.
ACT_APP_EXCEPTION	513	Application of this CPU entered in exception state	Application of this CPU entered in exception state.
NON_ACT_OTHER_CPU _APP_DELETED_BY _USER	600	Application of the other CPU was deleted	Application of the other CPU was deleted.
NON_ACT_OTHER_CPU _APP_RESET_BY_USER	601	Application of the other CPU was reset	Application of the other CPU was reset
NON_ACT_OTHER_CPU _APP_BRKP_BY_USER	602	Application of the other CPU is stopped by breakpoint	Application of other CPU is at breakpoint.

Table 224: Reasons of Redundancy States Change

6.6.3.2. Redundancy Commands

The command fields of the RedCmdLoc and RedCmdRem structures always have a suffix that can be Local or Remote. For example, there are command fields StandbyLocal and StandbyRemote, which have an effect equivalent to the command "Switch to Standby" in the CPU display.

A command with Local suffix generated in RedCmdLoc will be executed on the CPU itself (local).

On the other hand, a command with Remote RedCmdLoc-generated suffix will run on another CPU (remote). This works as follows:

- The remote CPU, in each cycle of MainTask, receives a copy of RedCmdLoc from local CPU via sync channels, and this copy is called RedCmdRem on remote CPU.
- The remote CPU only performs RedCmdRem commands that have the suffix Remote.

Example 1: If the local CPU is in the Active state, and the user wants to switch it to Standby State, the bit *DG_HX3040_01.RedCmdLoc.* must be set to 1 in the local CPU.

Example 2: If the remote CPU is in the active state, and the user wants to switch it to Standby State, the bit *DG_HX3040_01.RedCmdLoc* must be set to 1 in the local CPU

> **ATTENTION**
>
> If the diagnosis *DG_HX3040_01.RedDgnLoc.RedundancyDiagnostics.bExchangeSync* is indicating failure in Diagnostics and Commands Exchange service, a command with Remote suffix cannot be passed to the remote CPU, and therefore will not run.

To trigger a command, the corresponding bit in RedCmdLoc must be set. This can be done by a SCADA system, making a written via MasterTool, or even calling the bit inside a POU as UserPrg or NonSkippedPrg.

The user does not have to worry about turning off the command bit, which will be done automatically by the redundancy manager:

- In the case of commands executed on the local CPU (RedCmdLoc-command with Local suffix), the bit is turned off as soon as the command is realized and executed.
- In the case of commands executed on remote CPU (RedCmdRem-command with Remote suffix):
  - On remote CPU, the command is executed when the redundancy manager notices a raise trigger in control bit.

The fields of structures RedCmdLoc and RedCmdRem are defined next:

altus

.RedCmdLoc.* .RedCmdRem.*	Type	Description
bAutoConfigLocal	BOOL	**TRUE –** This diagnostic reports a request for automatic configuration, needed to leave the non-Configured state in some situations. **FALSE –** Automatic configuration request disabled.
bStandbyLocal	BOOL	**TRUE –** This command produces an action equivalent to the " Switch to Standby " request in the Redundancy menu available on the CPU display. **FALSE –** There is no pending request.
bMenuStandbyLocal	BOOL	**TRUE –** Turns on by a cycle of MainTask, indicating that the user has requested a "Switch to Standby" command through the Menu of the CPU display. **FALSE –** There is no pending request.
bInactiveLocal	BOOL	**TRUE –** This command produces an action equivalent to the request of "Switch to Inactive" in the Redundancy menu, available on the CPU display. **FALSE –** There is no pending request.
bMenuInactiveLocal	BOOL	**TRUE –** Turns on by a cycle of MainTask, indicating that the user has requested a "Switch to Inactive" command through the Menu of the CPU display. **FALSE –** There is no pending request.
bStandbyRemote	BOOL	**TRUE –** This command produces an action equivalent to the " Switch to Standby " request in the Redundancy menu available on the Remote CPU display. **FALSE –** There is no pending request.
bInactiveRemote	BOOL	**TRUE –** This command produces an action equivalent to "Switch to Inactive" request in Redundancy menu, available on the remote CPU display. **FALSE –** There is no pending request.
bResetNETStatisticsLocal	BOOL	**TRUE –** This command resets the statistics for A/B redundancy sync channels (see SyncLinkStatistics substructure in RedDgnLoc and RedDgnRem). Such statistics are failures and successes counters in synchronization services. **FALSE –** The reset command of the statistics of A/B sync channels on local CPU was not triggered.
bResetNETStatisticsRemote	BOOL	**TRUE –** Produces an action equivalent to the ResetNETStatisticsLocal command, but this time in the remote CPU. **FALSE –** The reset command of the statistics of A/B sync channels on remote CPU was not triggered.

Table 225: Redundancy Commands

6.6.3.3. User Information Exchanged among CPUA and CPUB

The Diagnostics and Commands Exchange synchronization service, exchange the following data structures between the two CPUs in each cycle of MainTask, using A/B sync channels:

- Redundancy Diagnostics (RedDgnLoc and RedDgnRem), already discussed in the section Redundancy Diagnostics Structure.
- Redundancy Commands (RedCmdLoc and RedCmdRem), already discussed in the section Redundancy Commands.
- User information exchanged between CPUA and CPUB (RedUsrLoc and RedUsrRem), which will be discussed in this section.

altus

The RedUsrLoc and RedUsrRem structures are just a 128-byte array, whose use can be freely set by the user. They allow the user to download in each cycle, 128 bytes of information from CPUA to CPUB, and 128 bytes from CPUB to CPUA.

RedUsrRem is a copy of RedUsrLoc from another CPU, received via internal sync channels. A CPU writes information in RedUsrLoc, which will be received in another CPU in RedUsrRem.

6.6.4. Redundancy Event Logs

The MasterTool allows observing various logs to a Hadron Xtorm CPU, among which is the redundancy event logs. These messages, specific to the redundancy, log in System Log.

Relevant modifications are recorded in the fields of the diagnostic data structure and redundancy commands, which are as follows:

- RedDgnLoc
- RedDgnRem
- RedCmdLoc
- RedCmdRem

In the case of diagnostic structures, only the following fields do not generate event logs:

- wRedStateDuration
- wCycleCounter
- dwIECTimer
- SyncLinkStatistics

Each line shown in the log has the following columns:

- *Time Stamp:* date and time of the event, with a resolution of milliseconds.
- *Severity:* information, warning, error or exception.
- *Description:* text that describes the event.
- *Component:* component that generated the event, which in the case of redundancy event log will be the *"Redundancy Management"*.

The text in the column "Description" contains information related to the event. For example, in the case of Redundancy State exchange from Standby to Active, an entry is generated in the log indicating that there was a state exchange (Standby to Active).

To access this screen, the user must double-click on the device (HX3040) in the device tree, and then select the tab *"Log"*. There is a filter that allows the user to select only the component *"Redundancy Management"*, in order to display only the redundancy events.

> **ATTENTION**
>
> Some diagnoses may point possible failures during initialization of redundant system and in the first cycles of tasks. However, in a correct functioning of the system, these diagnoses will indicate the absence of errors soon after system boot.

7.   Maintenance

> **ATTENTION**
>
> For a correct and safe maintenance, please check the Electrical Safety section.

7.1.   CPU Module Diagnostics

One feature of the Hadron Xtorm Series is the generation of diagnostics related to abnormal behaviors, whether they are failures, errors or operation modes. It allows the operator to easily identify and solve eventual problems in the system.

The Hadron Xtorm CPUs offers different ways to visualize the system diagnostics:

- One Touch Diag
- Diagnostics via LED
- Diagnostics via WEB
- Diagnostics via Variables
- Diagnostics via Function Blocks

The first one is an innovating feature of Hadron Xtorm Series, which allows a quick access to the abnormal conditions of the application. The second is purely visual, generated through one bicolor LED (DL) (red / blue) placed on the frontal part of the module. The next feature is the graphic visualization in a WEB page of the rack and the respective configured modules. It allows the individual access to the operation state and active diagnostics. The diagnostics are also directly stored in the CPU symbolic variables and can be used by the user application (presentation in a supervisory system, for example). The last ones provide specific conditions of the system functioning.

These diagnostics aim to point eventual problems regarding the installation or configuration of the system, as well as communication network issues. The user must see the Maintenance section, whenever necessary.

7.1.1.   One Touch Diag

The One Touch Diag (OTD), or single touch, is an exclusive feature brought by Hadron Xtorm Series for the Remote Terminal Unity. Through this new concept, the user can check the diagnostics of any module straight on the CPU graphic display with a single touch on the module Diagnostic Switch. This is a powerful diagnostic tool which can be used off-line (with no need of supervisory or programming software), that makes easier to find and solve eventual problems.

The diagnostics key is placed on the CPU upper part, in an easy access place and, besides giving active diagnostics, allows the access to the navigation menu, described in the Informative Menu and of CPU's Configuration section.

The figure below shows the CPU switch placement:
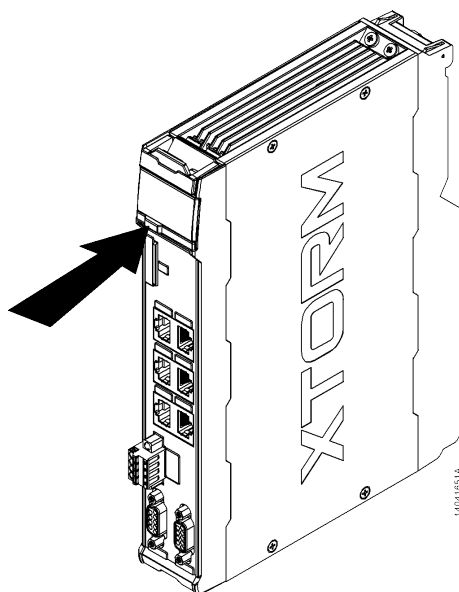
Figure 221: Diagnostics Switch

With only a short touch, the CPU starts to show the bus diagnostics (when active, otherwise shows the "NO DIAG" message). Initially, the Tag is visualized (configured in the module properties in the MasterTool Xtorm software, following the IEC 61131-3 standard), in other words, the name attributed to the CPU, and after that all diagnostics are shown, through CPU display messages. This process is executed twice on the display. Everything occurs automatically as the user only has to execute the first short touch and the CPU is responsible to show the diagnostics. The diagnostics of other modules present on the bus are also shown on the CPU graphic display by a short press in the diagnostic module button, in the same presentation model of diagnostics.

The figure below shows the process starting with the short touch, with the conditions and the CPU times presented in smaller rectangles. It is important to stress the diagnostics may have more than one screen, in other words, the specified time in the block diagram below is valid for one of them.
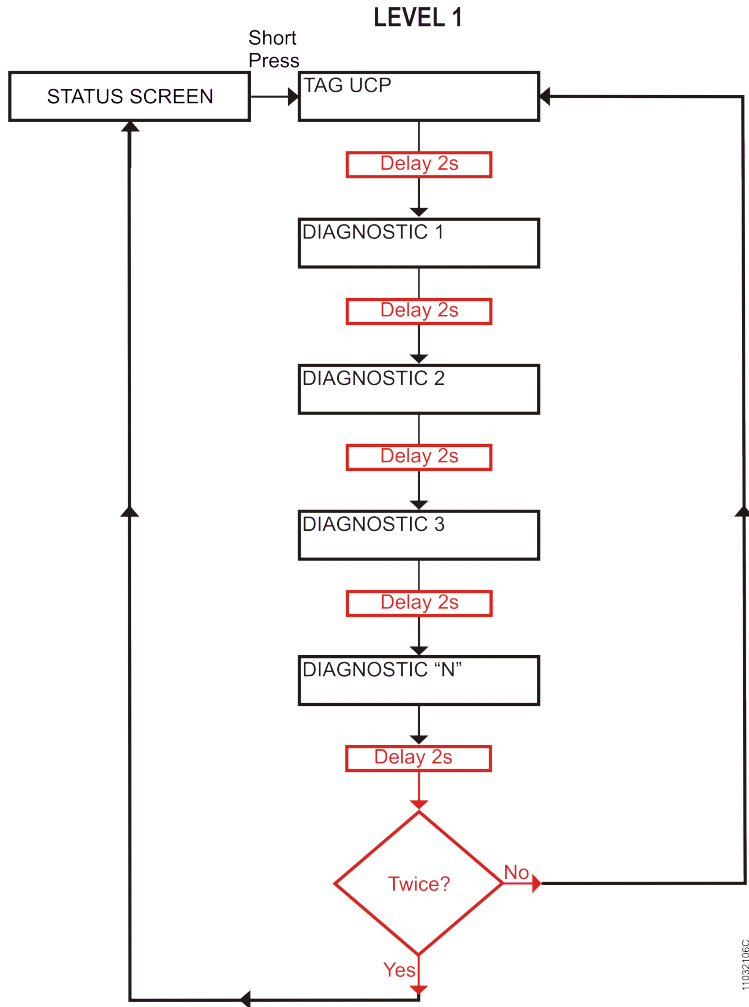
Figure 222: CPU Diagnostics View

Before all visualization process be concluded, it is just to give a short touch on the diagnostic switch, at any moment, or press the diagnostic switch from any I/O module connected to the bus.

In case a long touch is executed, the CPU goes to navigation menu, which is described in the Informative Menu and of CPU's Configuration section.

The table below shows the difference between the short touch time, the long touch time and stuck button.

Touch type	Minimum time	Maximum time	Indication condition
No touch	-	59,99 ms	-
Short touch	60 ms	0,99 s	Pressing and releasing the button within the set period
Long touch	1 s	20 s	Pressing and releasing the button within the set period
Locked Switch	20,01 s	(∞)	Pressing the button for more than 20 seconds

Table 226: One Touch Time

The messages presented on the Hadron Xtorm CPU graphic display, correspondent to the diagnostics, are described in the Diagnostics via Variables section, on table 228 e nas tabelas de Diagnósticos Detalhados logo abaixo.

If any situation of stuck button occur in one of the I/O modules, the diagnostic button of this module will stop of indicate the diagnostics on CPU graphic display when is pressed. In this case, the CPU will indicate that there is a module with active diagnostics. To remove this diagnostic from the CPU, a hot swap must be done in the module where the diagnostic is active.

> **ATTENTION**
>
> *One Touch Diag (OTD):* This option is only available to the user when the module is in operational mode.

7.1.2. Diagnostics via LED

Hadron Xtorm Series CPUs contain a LED for diagnostics indication (LED DL). The table below illustrates the meaning of each LED state.

DL (Color)	Description	Causes	Priority
Off	Module off or display failure.	No power supply. Hardware problem.	-
On (Blue)	Application in execution (Run Mode).	-	4 (Lower)
Blinking 2x (Blue)	CPU or Bus modules with diagnostic.	At least one bus module, including the CPU, presents an active diagnostic.	2
Blinking 3x (Blue)	Data forcing.	Some memory area is being forced by the user through MasterTool Xtorm.	3
On (Red)	Application stopped (Stop Mode).	-	4 (Lower)
Blinking 1x (Red)	Software watchdog.	User application watchdog.	1
Blinking 4x (Red)	Configuration or hardware error in the bus.	The bus is damaged or is not properly configured.	0 (Higher)

Table 227: Description of the Diagnostic LEDs States

Note:

Software Watchdog: In order to remove the watchdog indication reset the application or turn the CPU and on again. This watchdog occurs when the user application execution time is higher than the configured watchdog time.

7.1.3. Diagnostics via WEB

Besides the previously presented features, the Hadron Xtorm Series brings to the user an innovating access tool to the system diagnostics and operation states, through a WEB page.

altus

The utilization, besides being dynamic, is very intuitive and facilitates the user operations. The use of a supervisory system can be replaced when it is restricted to system status verification.

To access the desired CPU WEB page, it is just to use a standard navigator (Internet Explorer 7 or superior, Mozilla Firefox 3.0 or superior and Google Chrome 8 or superior) and type, on the address bar, the CPU IP address (Ex.: http://192.168.1.1). First, the CPU information is presented, according to the figure below:
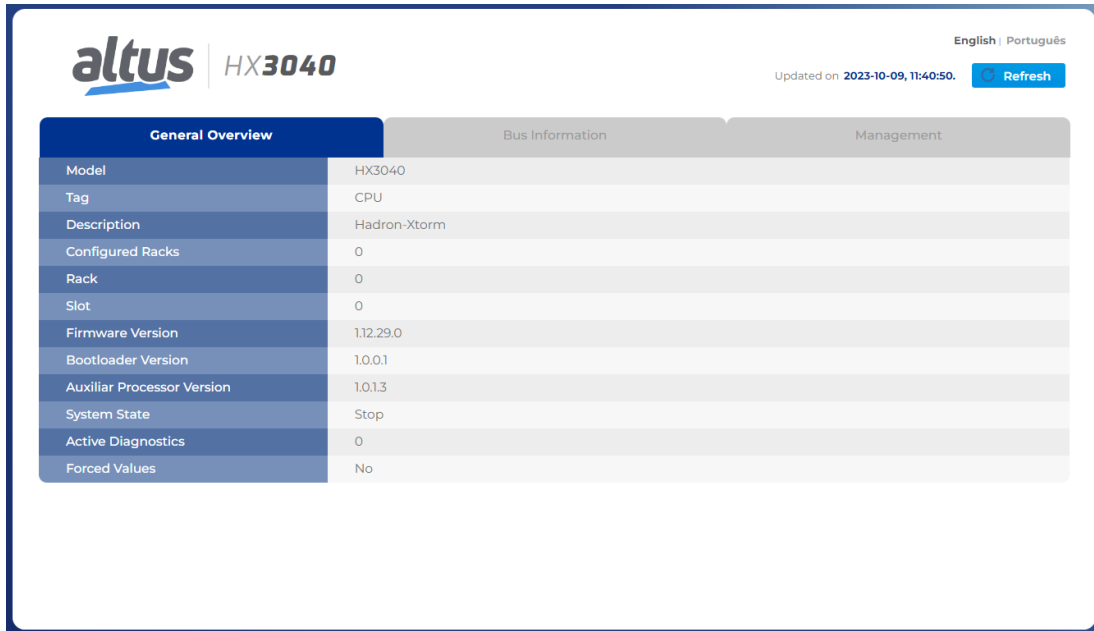
Figure 223: Initial Screen

There is also the *Bus Information* tab, which can be visualized through the Rack or the present module list (option on the screen right side). While there is no application on the CPU, this page will display a configuration with the largest available rack and a standard power supply, connected with the CPU. When the Rack visualization is used, the modules that have diagnostics blink and assume the red color, as shown on Figure 224. Otherwise a list with the system connected modules, Tags and active diagnostics number is presented:
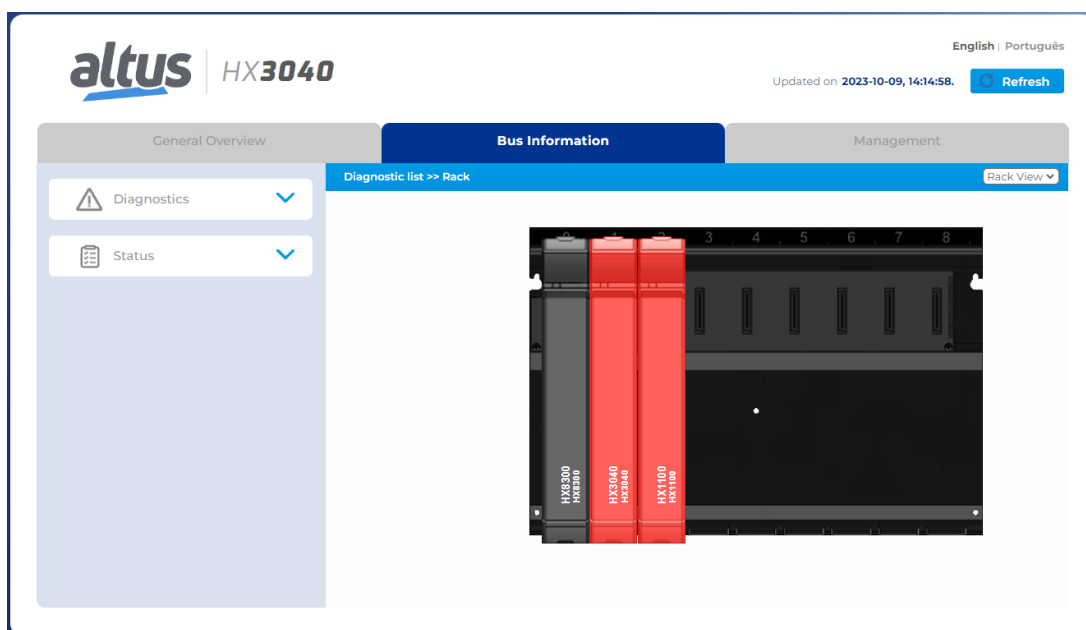
Figure 224: System Information

When the module with diagnostics is pressed, the module active(s) diagnostic(s) are shown, as illustrated in the figure below:

> **ATTENTION**
>
> When a CPU is restarted and the application goes to exception in the system's startup, the diagnostics will not be valid. It is necessary to fix the problem which generates the application's exception so that the diagnostics are updated.
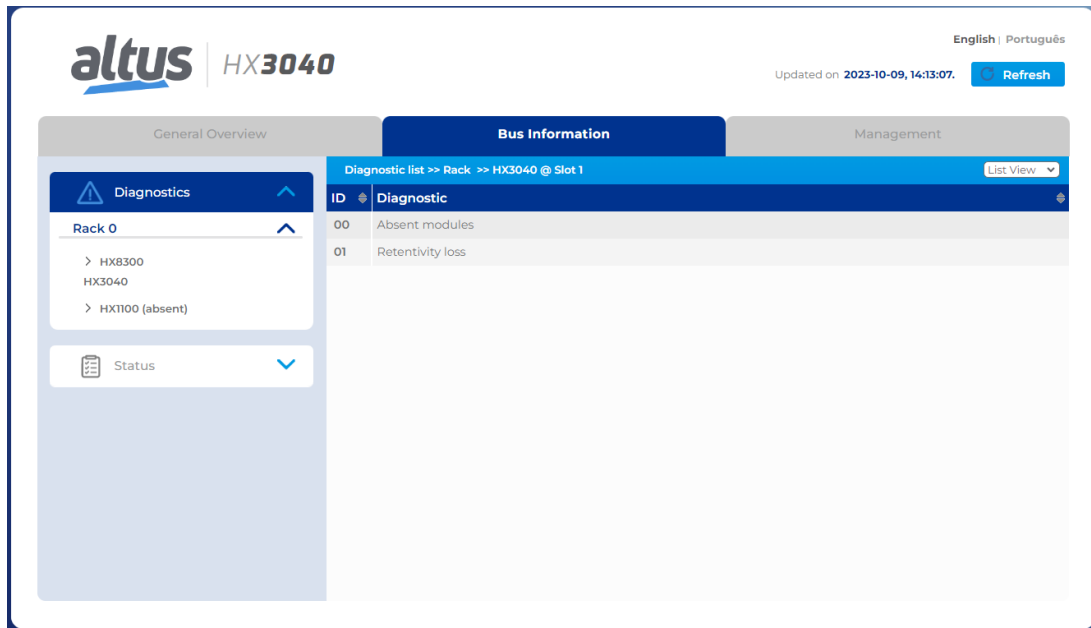
Figure 225: System Diagnostics

In case the *Status* section is selected, the state of all detailed diagnostics is shown on the screen, as illustrated in the figure below:
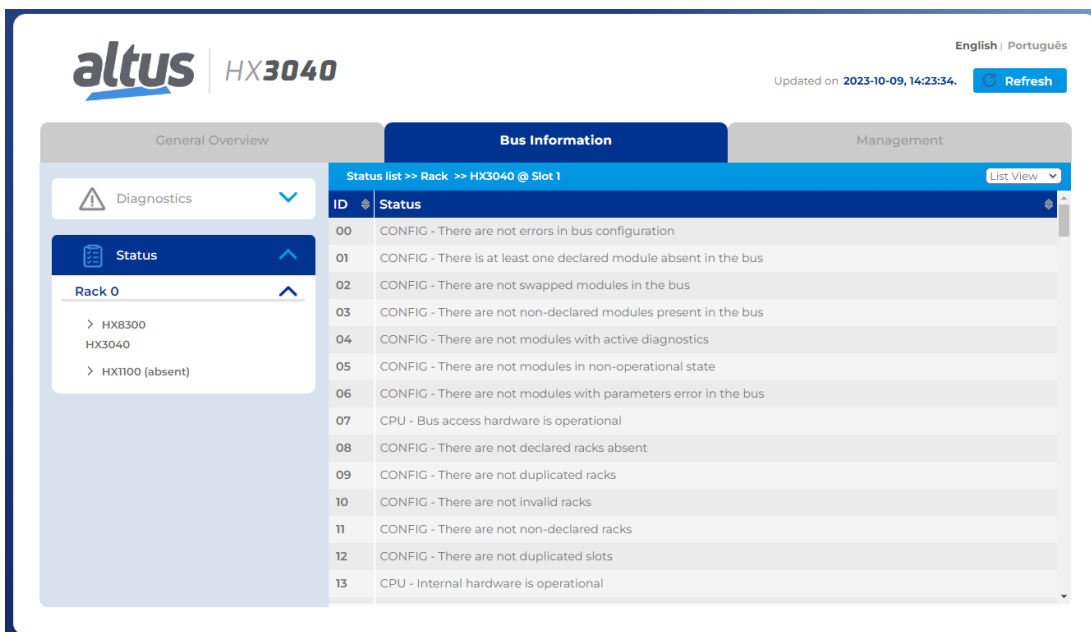
Figure 226: System Status

The user can choose to visualize two language options: Portuguese and English. Simply change in the upper right part of the screen to the desired language.

7.1.4. Diagnostics via Variables

The Hadron Xtorm Series CPU has a set of global symbolic variables where a range of diagnostic information related to hardware and software is available. These data structures are automatically created by the Xtorm MasterTool.

7.1.4.1. Summarized Diagnostics

The table below shows the meaning of each CPU summarized diagnostic variable:

Diagnostics Message	DG_modulename .tSummarized.*	Type	Description
NO DIAG	-	BOOL	No active diagnostic
CONFIG. MISMATCH	bConfigMismatch	BOOL	**TRUE** – Bus configuration problem (e.g. module inserted in the wrong position). **FALSE** – The bus is configured correctly.
ABSENT MODULES	bAbsentModules	BOOL	**TRUE** – One or more missing modules. **FALSE** – All declared modules are detected in the bus
SWAPPED MODULES	bSwappedModules	BOOL	**TRUE** – There are changed modules in the bus. **FALSE** – There are no changed modules in the bus
NON-DECLARED MODULES	bNonDeclaredModules	BOOL	**TRUE** – One or more bus modules are not declared in the configuration. **FALSE** – All modules are declared.
MODULES W/ DIAGNOSTICS	bModulesWithDiagnostic	BOOL	**TRUE** – One or more modules in the bus present an active diagnostic. **FALSE** – There are no active diagnostics in the bus modules.
MODULES W/ FATAL ERROR	bModuleFatalError	BOOL	**TRUE** – One or more modules in the bus are in fatal error. **FALSE** – All modules are working properly
MODULES W/ PARAM. ERROR	bModuleParameterError	BOOL	**TRUE** – One or more modules in the bus present parameterization error. **FALSE** – All modules are parameterized.
BUS ERROR	bWHSBBusError	BOOL	**TRUE** – The master indicates a failure in the WHSB bus. **FALSE** – The WHSB bus is working properly.
ABSENT RACK	bAbsentRacks	BOOL	**TRUE** – One or more declared racks are absent. **FALSE** – No absent racks.
DUPLICATED RACK	bDuplicatedRacks	BOOL	**TRUE** – Racks with duplicated identification number. **FALSE** – No racks with duplicated identification number.
INVALID RACK	bInvalidRacks	BOOL	**TRUE** – Racks with invalid identification number. **FALSE** – No racks with an invalid identification number.

altus

Diagnostics Message	DG_modulename .tSummarized.*	Type	Description
NON-DECLARED RACK	bNonDeclaredRacks	BOOL	**TRUE –** Racks with a non-declared identification number. **FALSE –** No racks with a non-declared identification number
DUPLICATED SLOT	bDuplicatedSlots	BOOL	**TRUE –** Duplicated slot address. **FALSE –** No duplicated slot address.
-	bReserved_13..15	BOOL	Reserved.
HARDWARE FAILURE	bHardwareFailure	BOOL	**TRUE –** CPU hardware failure. **FALSE –** The hardware is working properly.
SOFTWARE EXCEPTION	bSoftwareException	BOOL	**TRUE –** One or more exceptions generated by the software. **FALSE –** No exceptions generated in the software.
-	bReserved_18	BOOL	Reserved.
ERRO CARTAO DE MEMORIA	bMemoryCardError	BOOL	**TRUE –** The memory card is inserted in the CPU, but is not working properly. **FALSE –** The memory card is working properly.
-	bReserved_20..23	BOOL	Reserved.
COM 1 CONF. ERROR	bCOM1ConfigError	BOOL	**TRUE –** An error occurred during, or after, the COM 1 serial interface configuration. **FALSE –** The configuration of the COM 1 serial interface is correct.
COM 2 CONF. ERROR	bCOM2ConfigError	BOOL	**TRUE –** An error occurred during, or after, the COM 2 serial interface configuration. **FALSE –** The configuration of the COM 2 serial interface is correct.
NET 1 CONF. ERROR	bNET1ConfigError	BOOL	**TRUE –** An error occurred during, or after, the NET 1 Ethernet interface configuration. **FALSE –** The configuration of the NET 1 Ethernet interface is correct.
NET 2 CONF. ERROR	bNET2ConfigError	BOOL	**TRUE –** An error occurred during, or after, the NET 2 Ethernet interface configuration. **FALSE –** The configuration of the NET 2 Ethernet interface is correct.
NET 3 CONF. ERROR	bNET3ConfigError	BOOL	**TRUE –** An error occurred during, or after, the NET 3 Ethernet interface configuration. **FALSE –** The configuration of the NET 3 Ethernet interface is correct.
NET 4 CONF. ERROR	bNET4ConfigError	BOOL	**TRUE –** An error occurred during, or after, the NET 4 Ethernet interface configuration. **FALSE –** The configuration of the NET 4 Ethernet interface is correct.
NET 5 CONF. ERROR	bNET5ConfigError	BOOL	**TRUE –** An error occurred during, or after, the NET 5 Ethernet interface configuration. **FALSE –** The configuration of the NET 5 Ethernet interface is correct.
NET 6 CONF. ERROR	bNET6ConfigError	BOOL	**TRUE –** An error occurred during, or after, the NET 6 Ethernet interface configuration. **FALSE –** The configuration of the NET 6 Ethernet interface is correct.

altus

Diagnostics Message	DG_modulename .tSummarized.*	Type	Description
INVALID DATE/TIME	bInvalidDateTime	BOOL	**TRUE** – The date or time is invalid. **FALSE** – The date and time are correct.
RUNTIME RESET	bRuntimeReset	BOOL	**TRUE** – The RTS (Runtime System) has been restarted at least once. This diagnostic is only cleared on system restart. **FALSE** – The RTS (Runtime System) is operating normally.
OTD SWITCH ERROR	bOTDSwitchError	BOOL	**TRUE** – The button was blocked for more than 20 s at least once while the CPU was energized. This diagnostic is only cleared on system restart.
RETENTIVITY LOSS	bRetentivityLost	BOOL	**TRUE** – Invalid data in the retentive memory during start up.
CPU WRONG POSITION	bWrongCPUSlot	BOOL	**TRUE** – Incorrect CPU position (different position than the one configured by the programmer). **FALSE** – Correct CPU position
-	bReserved_37..39	BOOL	Reserved.

Table 228: CPU Summarized Diagnostics

Notes:

No Diag: There is no active diagnostic in the CPU.

Configuration Mismatch: The incompatible configuration diagnostic is generated if one or more present modules (physically) do not correspond to the declared one. The changed modules diagnostic will turn on in case there is a change of two modules between themselves (it is a complementary information to the incompatible configuration).The modules inserted in the bus as well as the declared ones in the project do not enable this diagnostic bit. This condition is valid only in case of absent or different modules.

Absent Modules: This diagnostic is true if there are one or more absent modules.

Swapped Modules: If only two modules are changed between themselves in the bus, then changed diagnostic can be identified. Otherwise, the problem is treated as "Configuration Mismatch".

Non-declared Modules: This diagnostic is true if one or more bus modules are not declared.

Modules with Diagnostics: This diagnostic is true if one or more bus modules present active diagnostics.

Modules with Fatal Error: In case the modules with fatal error diagnostic is true, it must be verified which is the problematic module in the bus and send it to Altus Technical Assistance, as it has hardware failure.

Module with Parameterization Error: In case the parameterization error diagnostic is true, the user must check if the bus modules are correctly configured and if the firmware /MasterTool Xtorm version are correct.

Bus Error: This error interrupts the modules bus access. Probable causes: hot swap configuration orhardware problem in bus communication lines. In this case, contact Altus Technical Assistance.

Absent Rack: If one or more declared racks are missing then the diagnosis of Absent Rack will be true.

Duplicated Rack: If there is any rack with the identification number duplicated then the diagnosis of Duplicated Rack will be true.

Invalid Rack: If there is any rack with invalid identification number then the diagnosis of Invalid Rack will be true.

Non-Declared Rack: If there is any rack with undeclared identification number then the diagnosis of Non-Declared Rack will be true.

Duplicated Slot: If there is any duplicated address position then the diagnosis of Duplicated Slot will be true.

Hardware Failure: In case the Hardware Failure diagnostic is true, the CPU must be sent to Altus Technical Assistance, as it has problems in the RTC, auxiliary processor, or other hardware resources.

Software Exception: In case the software exception diagnostic is true, the user must verify his application to guarantee it is not accessing the memory wrongly. If the problem remains, the Altus Technical Support sector must be consulted. The software exception codes are described next in the CPU detailed diagnostics table.

Retentivity Loss: This diagnostic indicates that there was loss of non-volatile data (retain/persistent variables and event queue). It is turned on only when the loss is caused by hardware problems (hardware failure, or hot swap of power supply or CPU). Cold reset commands and reset origin triggered by MasterTool Xtorm tool does not cause the indication of this diagnosis.

altus

7.1.4.2. Detailed Diagnostics

The following tables show the detailed diagnostics of the Hadron Xtorm CPU, for your reference it is important to check the notes below:

- **Diagnostic Structures Visualization:** The Diagnostics Structures added to the project can be viewed by accessing the item *"Library Manager"* in MasterTool Xtorm treeview. This makes it possible to visualize all the data types defined in the structure.
- **Counters:** All counters of the CPU diagnostics return to zero when the limit value is exceeded.

DG_HX3040.tDetailed.* Variable	Type	Description
Target. dwCPUModel	DWORD	HX3040 = 0x3040.
Target. abyCPUVersion	BYTE ARRAY(4)	Firmware version.
Target. abyBootloaderVersion	BYTE ARRAY(4)	Bootloader version.
Target. abyAuxprocVersion	BYTE ARRAY(4)	Auxiliary processor version.

Table 229: Target Group Detailed Diagnostics

DG_HX3040.tDetailed.* Variable	Type	Description
Hardware. bAuxprocFailure	BIT	Faulty communication between the auxiliary processor and the main processor.
Hardware. bRTCFailure	BIT	The main processor is not able to communicate with the RTC (CPU clock).
Hardware. bThermometerFailure	BIT	Communication failure between the thermometer and the main processor.
Hardware. bLCDFailure	BIT	Faulty communication between the LCD display and the main processor.

Table 230: Hardware Group Detailed Diagnostics

DG_HX3040.tDetailed.* Variable	Type	Description
Exception. wExceptionCode	WORD	Exception Code generated by the RTS, the meaning of the codes can be seen in the table 232.
Exception. byProcessorLoad	BYTE	Level, in percent (%), of processor load.

Table 231: Exception Group Detailed Diagnostics

Code	Description	Code	Description
0x0000	There is no exception code.	**0x0051**	Access violation
0x0010	Watchdog time of the expired IEC task (Software Watchdog).	**0x0052**	Privileged instruction.
0x0012	I/O configuration error.	**0x0053**	Page failure
0x0013	Check-up errors after program download.	**0x0054**	Stack overflow.
0x0014	Fieldbus error.	**0x0055**	Invalid disposition
0x0015	I/O updating error.	**0x0056**	Invalid maneuver.
0x0016	Cycle time (execution) exceeded	**0x0057**	Protected page.
0x0017	Program online updating too long.	**0x0058**	Double failure.
0x0018	Unsolved external references.	**0x0059**	Invalid OpCode.
0x0019	Download rejected	**0x0100**	Data type misalignment.
0x001A	Project unloaded, as the retentive variables cannot be reallocated.	**0x0101**	Arrays limit exceeded.
0x001B	Project unloaded and deleted.	**0x0102**	Division by zero
0x001C	Out of memory stack.	**0x0103**	Overflow.
0x001D	Corrupted retentive memory, cannot be mapped.	**0x0104**	Cannot be continued
0x001E	Project can be loaded but it causes a break later on.	**0x0105**	Watchdog in the processor load of all IEC task detected.
0x0021	Target of startup application does not match to the current target.	**0x0150**	FPU: Not specified error.
0x0022	Scheduled tasks erro. Downloaded file Check-up error.	**0x0151** **0x0153**	FPU: Abnormal operand FPU: Inexact result.
0x0024	Mismatch between the retentive identity and the current boot project program identity	**0x0154**	FPU: Invalid operation.
0x0025	EC task configuration failure	**0x0155**	FPU: Overflow.
0x0026	Application is running with the wrong target.	**0x0156**	FPU: Stack verification.
0x0050	Illegal instruction.	**0x0157**	FPU: Underflow.

Table 232: RTS Exception Codes

DG_HX3040.tDetailed.* Variable	Type	Description
RetainInfo. byCPUInitStatus	BYTE	CPU Initialization Status: 01: Hot start 02: Warm Start 03: Cold Start Note: These variables are reset on every power-up.
RetainInfo. wCPUColdStartCounter	WORD	Cold Start Counter: Increments when the CPU starts with loss of retentivity. (0 to 65535).
RetainInfo. wCPUWarmStartCounter	WORD	Warm Start Counter: Increments when the CPU starts normally with valid retain data. (0 to 65535).

altus

DG_HX3040.tDetailed.* Variable	Type	Description
RetainInfo. wCPUHotStartCounter	WORD	Counter of disturbances less than the time to support failures in the CPU power supply. (0 to 65535)
RetainInfo. wRTSResetCounter	WORD	RTS (Runtime System) reset counter. (0 a 65535)

Table 233: RetainInfo Group Detailed Diagnostics

DG_HX3040.tDetailed.* Variable	Type	Description
Reset. bBrownOutReset	BIT	The CPU has been restarted due to a power failure during the last power-up.
Reset. bWatchdogReset	BIT	The UCP has been restarted due to the active watchdog during the last power-up.

Table 234: Reset Group Detailed Diagnostics

Note:

Brownout Reset: The brownout reset diagnostic is only true when the power supply exceeds the minimum limit required in its technical features, remaining in low voltage, without suffering any interruption. The CPU identifies the voltage break and indicates the power supply failure diagnostic. When the voltage is reestablished, the CPU is restarted automatically and indicates the brownout reset diagnostic.

DG_HX3040.tDetailed.* Variable	Type	Description
Thermometer. bOverTemperatureAlarm	BIT	Alarm generated due to the internal temperature being at or above 85°C.
Thermometer. bUnderTemperatureAlarm	BIT	Alarm generated due to the internal temperature being at or below 0°C.
Thermometer. diTemperature	DINT	Temperature read on the internal sensor of the CPU, measured in Celsius degrees, with its resolution given in 1°C.

Table 235: Thermometer Group Detailed Diagnostics

DG_HX3040.tDetailed.* Variable	Type	Description
Serial.COM[1-2]. byProtocol	ENUM (BYTE)	Selected protocol in COM[1-2]: NO_PROTOCOL (0): No protocol; MODBUS_RTU_MASTER (1): MODBUS RTU Master; MODBUS_RTU_SLAVE (2): MODBUS RTU Slave; OTHER_PROTOCOL (3): Other protocol;
Serial.COM[1-2]. dwRXBytes	DWORD	Counter of characters received from COM[1-2]. (0 to 4294967295)
Serial.COM[1-2]. dwTXBytes	DWORD	Counter of characters transmitted from COM[1-2]. (0 to 4294967295)

altus

DG_HX3040.tDetailed.* Variable	Type	Description
Serial.COM[1-2]. **wRXPendingBytes**	WORD	Number of characters left in the reading buffer in COM[1-2]. (0 to 65535)
Serial.COM[1-2]. **wTXPendingBytes**	WORD	Number of characters left in the transmission buffer in COM[1-2]. (0 to 65535)
Serial.COM[1-2]. **wBreakErrorCounter**	WORD	These counters are restarted in the following conditions:
Serial.COM[1-2]. **wParityErrorCounter**	WORD	Energizing; COM[1-2] serial port configuration;
Serial.COM[1-2]. **wFrameErrorCounter**	WORD	Removal of RX and TX queues;
Serial.COM[1-2]. **wRXOverrunCounter**	WORD	

Table 236: Serial COM 1 and COM 2 Group Detailed Diagnostics

Note:

Parity Error Counter: When the CPU is configured with parity *No Parity*, the parity error counter is not incremented if it receives a different parity. In this case, a frame error will be indicated. The maximum value of each counter is 65535.

DG_HX3040.tDetailed.* Variable	Type	Description
Ethernet.NET[1-6]. **bLinkDown**	BIT	Indicates the state of the link in the interface.
Ethernet.NET[1-6]. **byOperatingMode**	ENUM(BYTE)	Interface operation mode: STANDALONE (0): Normal mode. NIC_TEAMING (1): NIC Teaming with other interface mode. SWITCH (2): Switch mode.
Ethernet.NET[1-6]. **byOperatingState**	ENUM(BYTE)	Interface state: ERROR_UNCONFIGURED (0): Non-configured interface. DISABLED (1): Disabled interface. CONFIGURED_AND_RUNNING (2): Interface configured and operational.
Ethernet.NET[1-6]. **wProtocol**	WORD	Selected protocol in NET[1-6]: 00: Without protocol.
Ethernet.NET[1-6]. **wProtocol.** **bMODBUS_RTU_ETH_Client**	BIT	MODBUS RTU Client via TCP.
Ethernet.NET[1-6]. **wProtocol.** **bMODBUS_ETH_Client**	BIT	MODBUS TCP Client.
Ethernet.NET[1-6]. **wProtocol.** **bMODBUS_RTU_ETH_Server**	BIT	MODBUS RTU Server via TCP.
Ethernet.NET[1-6]. **wProtocol.** **bMODBUS_ETH_Server**	BIT	MODBUS TCP Server.

altus

DG_HX3040.tDetailed.* Variable	Type	Description
Ethernet.NET[1-6]. wProtocol. bDNP3_Client	BIT	DNP3 Client.
Ethernet.NET[1-6]. wProtocol. bDNP3_Server	BIT	DNP3 Server.
Ethernet.NET[1-6]. wProtocol. bIEC61850_Server	BIT	IEC 61850 Server.
Ethernet.NET[1-6]. szIP	STRING(15)	Port IP Address.
Ethernet.NET[1-6]. szMask	STRING(15)	Port Subnet Mask.
Ethernet.NET[1-6]. szGateway	STRING(15)	Port Gateway Address.
Ethernet.NET[1-6]. szMAC	STRING(17)	Port MAC Address.
Ethernet.NET[1-6]. abyIP	BYTE ARRAY(4)	Port IP Address.
Ethernet.NET[1-6]. abyMask	BYTE ARRAY(4)	Port Subnet Mask.
Ethernet.NET[1-6]. abyGateway	BYTE ARRAY(4)	Port Gateway Address.
Ethernet.NET[1-6]. abyMAC	BYTE ARRAY(6)	Port MAC Address.
Ethernet.NET[1-6]. NICTeaming. bStandbyState	BIT	Interface in stand-by state.
Ethernet.NET[1-6]. NICTeaming. bActiveState	BIT	Interface in active state.
Ethernet.NET[1-6]. NICTeaming. bLinkDown	BIT	Interface without link.
Ethernet.NET[1-6]. NICTeaming. bInterMsgTimeout	BIT	Time-out waiting for NIC Teaming pair packages.
Ethernet.NET[1-6]. NICTeaming. bGeneralRxMsgTimeout	BIT	Time-out waiting for network packages.
Ethernet.NET[1-6]. dwTransmittedBytes	DWORD	Counter of bytes sent though the interface. (0 to 4294967295)
Ethernet.NET[1-6]. dwTransmittedPackets	DWORD	Counter of packages sent though the interface (0 to 4294967295)
Ethernet.NET[1-6]. dwTransmittedDropErrors	DWORD	Counter of connection losses in the transmission though the interface. (0 to 4294967295)

altus

DG_HX3040.tDetailed.* Variable	Type	Description
Ethernet.NET[1-6]. **dwTransmittedCollisionErrors**	DWORD	Counter of collision errors in the transmission through the interface. (0 to 4294967295)
Ethernet.NET[1-6]. **dwReceivedBytes**	DWORD	Counter of bytes received through the port. (0 to 4294967295)
Ethernet.NET[1-6]. **dwReceivedPackets**	DWORD	Counter of received packages through the port. (0 to 4294967295)
Ethernet.NET[1-6]. **dwReceivedDropErrors**	DWORD	Counter of connection losses in the reception through the interface. (0 to 4294967295)
Ethernet.NET[1-6]. **dwReceivedFrameErrors**	DWORD	Counter of frame errors in the reception through the interface. (0 to 4294967295)

Table 237: Ethernet NET 1 .. NET 6 Group Detailed Diagnostics

DG_HX3040.tDetailed.* Variable	Type	Description
UserFiles. **byMounted**	BYTE	Indicates if the memory used for recording the user files is able to receive data.
UserFiles. **dwFreeSpacekB**	DWORD	Free memory space for user files (Kbytes).
UserFiles. **dwTotalSizekB**	DWORD	Storage capacity of the memory of user files (Kbytes).

Table 238: UserFiles Group Detailed Diagnostics

DG_HX3040.tDetailed.* Variable	Type	Description
UserLogs. **byMounted**	BYTE	Status of memory in which the user logs are inserted.
UserLogs. **wFreeSpacekB**	WORD	Free memory space of user logs (Kbytes).
UserLogs. **wTotalSizekB**	WORD	Storage capacity of the memory of user logs (Kbytes).

Table 239: UserLogs Group Detailed Diagnostics

DG_HX3040.tDetailed.* Variable	Type	Description
MemoryCard. **byMounted**	ENUM(BYTE)	Memory Card Status: NOT_MOUNTED (0): Memory card not mounted. MOUNTED (1): Memory card inserted and mounted.
MemoryCard. **bMemcardtoCPUEnabled**	BIT	Protection level of the memory card: Data reading of the memory card by the authorized CPU.

DG_HX3040.tDetailed.* Variable	Type	Description
MemoryCard. bCPUtoMemcardEnabled	BIT	Data writing in the memory card by the authorized CPU.
MemoryCard. dwFreeSpacekB	DWORD	Free memory card space (Kbytes).
MemoryCard. dwTotalSizekB	DWORD	Memory card storage capacity (Kbytes).

Table 240: MemoryCard Group Detailed Diagnostics

DG_HX3040.tDetailed.* Variable	Type	Description
WHSB. byHotSwapAndStartupStatus	ENUM(BYTE)	Informs the abnormal situation in the bus which caused the application to stop, for each *Hot Swap Mode*. See table 242 for details of the possibilities.
WHSB. adwRackIOErrorStatus	DWORD ARRAY(32)	Identification of errors in individual I/O modules. For more information on this diagnostic see the note below.
WHSB. adwModulePresenceStatus	DWORD ARRAY(32)	Status of the presence of declared I/O modules on buses, individually. For more information about this diagnostic see the note below.
WHSB. byWHSBBusErrors	BYTE	WHSB bus failure counter. This counter is reset on power-up (0 to 255).

Table 241: WHSB Group Detailed Diagnostics

Notes:

Error diagnosis of the bus modules: Each DWORD in this diagnostic array represents a rack, whose positions are represented by the bits of these DWORDs. Thus, Bit-0 of DWORD-0 is equivalent to the zero position of the rack with address zero. Each of these Bits is the result of a logical OR operation between the diagnostics of incompatible configuration (*bConfigMismatch*), absent modules (*bAbsentModules*), swapped modules (*bSwappedModules*), modules with fatal error (*bModuleFatalError*) and the operational state of the module of a certain position.

Module Presence Status: Each DWORD in this diagnostic array represents a rack, whose positions are represented by the bits in these DWORDs. Thus, Bit-0 of DWORD-0 is equivalent to the zero position of the rack with address zero. So, if the module is present, this bit will be true. It is important to point out that this diagnosis is valid for all modules, except power supplies and CPUs, that is, they do not present on the bus in their respective positions (bit remains in false).

Situations that cause the application to stop: The codes for possible situations that cause the application to stop can be seen below:

Code	Enumerable	Description
00	INITIALIZING	This state is presented while the other states are not ready.
01	RESET_WATCHDOG	Application in Stop Mode due the reset by hardware watchdog or by a Runtime restart, when the configuration "Start User Application after Reset by Watchdog" is disabled.
02	ABSENT_MODULES_HOT_SWAP_ DISABLED	Application in Stop Mode due to activation of the Absent Modules diagnostic, when hot swapping is configured as disabled or disabled only for declared modules.

Code	Enumerable	Description
03	CFG_MISMATCH_HOT_SWAP_ DISABLED	Application in Stop Mode due to activation of the Incompatible Configuration diagnostic, when hot swapping is configured as disabled or disabled only for declared modules.
04	ABSENT_MODULES_HOT_SWAP_ STARTUP_CONSISTENCY	Application in Stop Mode due to activation of the Absent Modules diagnostic, when hot swapping is configured with startup consistency or with startup consistency for declared modules only.
05	CFG_MISMATCH_HOT_SWAP_ STARTUP_CONSISTENCY	Application in Stop Mode due to activation of Incompatible Configuration diagnostics, when hot swapping with startup consistency or with startup consistency only for declared modules is configured.
06	APPL_STOP_ALLOWED_TO_RUN	Application in Stop Mode and all consistencies performed successfully. Application can be set to Run Mode.
07	APPL_STOP_MODULES_NOT_READY	Application in Stop Mode and all consistencies performed successfully, but the I/O modules are not able for the system to start. It is not possible to set the Application to Run Mode.
08	APPL_STOP_MODULES_GETTING_ READY_TO_RUN	Application in Stop Mode and all consistencies performed successfully. The I/O modules are being prepared for system startup. It is not possible to set the Application to Run Mode.
09	NORMAL_OPERATING_STATE	Application in Run Mode.
10	MODULE_CONSISTENCY_OK	Internal use.
11	APPL_STOP_DUE_TO_EXCEPTION	Application in Stop Mode due to an exception in the CPU.
12	DUPLICATED_SLOT_HOT_SWAP_ DISABLED	Application in Stop Mode due to activation of the Duplicate Slots diagnostic, when hot swapping is configured as disabled or disabled for declared modules only.
13	DUPLICATED_SLOT_HOT_SWAP_ STARTUP_CONSISTENCY	Application in Stop Mode due to activation of the Duplicate Slots diagnostic, when hot swapping is configured with startup consistency or with startup consistency for declared modules only.
14	DUPLICATED_SLOT_HOT_SWAP_ ENABLED	Application in Stop Mode due to activation of the Duplicate Slots diagnostic, when hot swapping is configured as enabled with no startup consistency.
15	NON_DECLARED_MODULE_HOT_ SWAP_STARTUP_CONSISTENCY	Application in Stop Mode due to activation of the Non-Declared Modules diagnostic, when hot swapping is configured as enabled with startup consistency.
16	NON_DECLARED_MODULE_HOT_ SWAP_DISABLED	Application in Stop Mode due to activation of the Non-Declared Modules diagnostic, when hot swapping is configured as disabled.

Table 242: Codes of situations that cause application stopping

altus

DG_HX3040.tDetailed.* Variable	Type	Description
Application. byCPUState	ENUM(BYTE)	Informs the CPU operation state: RUN (1): The application is in execution (Run Mode). STOP (3): The application is stopped (Stop Mode).
Application. bForcedIOs	BIT	There are one or more forced I/O points.

Table 243: Application Group Detailed Diagnostics

DG_HX3040.tDetailed.* Variable	Type	Description
TimeManager. eActiveTimeSource	ENUM(BYTE)	Indicates which synchronism source is active: NONE (0): No sync source is active. IRIGB (1): Time synchronization via IRIGB is active. SNTP (2): Time synchronization via SNTP is active. DNP (3): Time synchronization via DNP3 is active. PTP (4): Time synchronization via PTP is active.
TimeManager.SNTP. bServiceEnabled	BIT	SNTP service enabled.
TimeManager.SNTP. byActiveTimeServer	ENUM(BYTE)	Indicates which server is active: NO_TIME_SERVER (0): No active server. PRIMARY_TIME_SERVER (1): Active primary server. SECONDARY_TIME_SERVER (2): Active Secondary server.
TimeManager.SNTP. wPrimaryServerDownCount	WORD	Counter of times in which the primary server is unavailable. (0 to 65535)
TimeManager.SNTP. wSecondaryServerDownCount	WORD	Counter of times in which the secondary server is unavailable. (0 to 65535)
TimeManager.SNTP. dwRTCTimeUpdatedCount	DWORD	Counter of times the RTC was updated by the SNTP service. (0 to 4294967295)
TimeManager.SNTP. byLastUpdateSuccessful	ENUM(BYTE)	Last update status: NOT_UPDATED (0): Not updated. FAILED (1): Last update failed. UPDATE_SUCCESSFUL (2): Last update was successful.
TimeManager.SNTP. byLastUpdateTimeServer	ENUM(BYTE)	Indicates which server was used in the last update: NO_TIME_SERVER (0): None update. PRIMARY_TIME_SERVER (1): Primary Server. SECONDARY_TIME_SERVER (2): Secondary Server.

altus

DG_HX3040.tDetailed.* Variable	Type	Description
TimeManager.SNTP. sLastUpdateTime	EXTENDED_DATE_ AND_TIME	Date and time of the last sync time via SNTP.
TimeManager.SNTP. sLastUpdateTime. byDayOfMonth	BYTE	
TimeManager.SNTP. sLastUpdateTime. byMonth	BYTE	
TimeManager.SNTP. sLastUpdateTime. wYear	WORD	
TimeManager.SNTP. sLastUpdateTime. byHours	BYTE	
TimeManager.SNTP. sLastUpdateTime. byMinutes	BYTE	
TimeManager.SNTP. sLastUpdateTime. bySeconds	BYTE	
TimeManager.SNTP. sLastUpdateTime. byMilliseconds	WORD	
TimeManager.IRIGB. bServiceEnabled	BIT	IRIGB service enabled.
TimeManager.IRIGB. dwTimeUpdatedCount	DWORD	Counter of times the time was updated by IRIGB service. (0 to 4294967295)
TimeManager.IRIGB. byLastUpdateSuccessful	ENUM(BYTE)	Indicates which server was used in the last update: NO_TIME_SERVER (0): None update. PRIMARY_TIME_SERVER (1): Primary Server. SECONDARY_TIME_SERVER (2): Secondary Server.
TimeManager.IRIGB. sLastUpdateTime	EXTENDED_DATE_ AND_TIME	Date and time of the last sync time via IRIGB.
TimeManager.IRIGB. sLastUpdateTime. byDayOfMonth	BYTE	
TimeManager.IRIGB. sLastUpdateTime. byMonth	BYTE	
TimeManager.IRIGB. sLastUpdateTime. wYear	WORD	
TimeManager.IRIGB. sLastUpdateTime. byHours	BYTE	

DG_HX3040.tDetailed.* Variable	Type	Description
TimeManager.IRIGB. sLastUpdateTime. byMinutes	BYTE	
TimeManager.IRIGB. sLastUpdateTime. bySeconds	BYTE	
TimeManager.IRIGB. sLastUpdateTime. byMilliseconds	WORD	

Table 244: TimeManager Group Detailed Diagnostics

Note:

Active Sync Source: This diagnostic may take a few seconds to update after the system's active synchronization source is switched (which also occurs at system startup).

DG_HX3040.tDetailed.* Variable	Type	Description
Events. wUsage	WORD	Usage sampling of the event queue.
Events. bEventQueueOverFlow	BIT	Bit that indicates an OverFlow in the event queue. See ClearRtuDiagnostic section.
Events. bEventQueueCleared	BIT	Bit that indicates a clearance OverFlow in the event queue. See ClearEventQueue section.
Events. byReserved_00	BYTE	Reserved.
Events. wReserved_01	WORD	Reserved.
Events. dwReserved_02	DWORD	Reserved.

Table 245: Events Group Detailed Diagnostics

DG_HX3040.tDetailed.* Variable	Type	Description
Rack. dwAbsentRacks	DWORD	Each bit represents a rack identification number, if some bit is TRUE, it means this rack is absent
Rack. dwDuplicatedRacks	DWORD	Each bit represents a rack identification number, if some bit is TRUE, it means that more than one rack is set with the same identification number.
Rack. dwNonDeclaredRacks	DWORD	Each bit represents a rack identification number, if some bit is TRUE, it means there is a rack set with a non-declared identification number.

Table 246: Rack Group Detailed Diagnostics

altus

7.1.5. Diagnostics via Function Blocks

The function blocks allow the visualization of some parameters which cannot be accessed otherwise. The function regarding advanced diagnostics is in the *NextoStandard* library and is described below.

7.1.5.1. GetTaskInfo

This function returns the task information of a specific application.

Figure 227: GetTaskInfo Function

Below, the parameters that must be sent to the function for it to return the application information are described.

Input parameter	Type	Description
psAppName	POINTER TO STRING	Application name.
psTaskName	POINTER TO STRING	Task name.
pstTaskInfo	POINTER TO stTask-Info	Pointer to receive the application information.

Table 247: GetTaskInfo Input Parameters

The data returned by the function, through the pointer informed in the input parameters are described on table below.

Returned Parameters	Size	Description
dwCurScanTime	DWORD	Task cycle time (execution) with 1 μs resolution.
dwMinScanTime	DWORD	Task cycle minimum time with 1 μs resolution.
dwMaxScanTime	DWORD	Task cycle maximum time 1 μs resolution.
dwAvgScanTime	DWORD	Task cycle average time with 1 μs resolution.
dwLimitMaxScan	DWORD	Task cycle maximum time before watchdog occurrence.
dwIECCycleCount	DWORD	IEC cycle counter.

Table 248: GetTaskInfo Output Parameters

Possible ERRORCODE:

- NoError: success execution;
- TaskNotPresent: the desired task does not exist.

Example of utilization in ST language:

```
PROGRAM UserPrg
VAR
sAppName : STRING;
psAppName : POINTER TO STRING;
sTaskName : STRING;
psTaskName : POINTER TO STRING;
pstTaskInfo : POINTER TO NextoStandard.stTaskInfo;
TaskInfo :NextoStandard. stTaskInfo;
Info : NextoStandard.ERRORCODE;
END_VAR
//INPUTS:
sAppName := 'Application';    //Variable receives the application name.
psAppName := ADR(sAppName); //Pointer with application name.
sTaskName := 'MainTask';    //Variable receives task name.
psTaskName := ADR(sTaskName); //Pointer with task name.
pstTaskInfo := ADR(TaskInfo); //Pointer that receives task info.
//FUNCTION:
//Function call.
Info := GetTaskInfo (psAppName, psTaskName, pstTaskInfo);
//Variable Info receives possible function errors.
```

7.1.6. Graphic Display

The graphic display available in this product has an important tool for the process control, as through it is possible to recognize possible error conditions, active components or diagnostics presence. Besides, all diagnostics including the I/O modules are presented to the user through the graphic display. For further information regarding the diagnostic key utilization and its visualization see One Touch Diag section.

On figure below, it is possible to observe the available characters in this product graphic display and, next, its respective meanings.
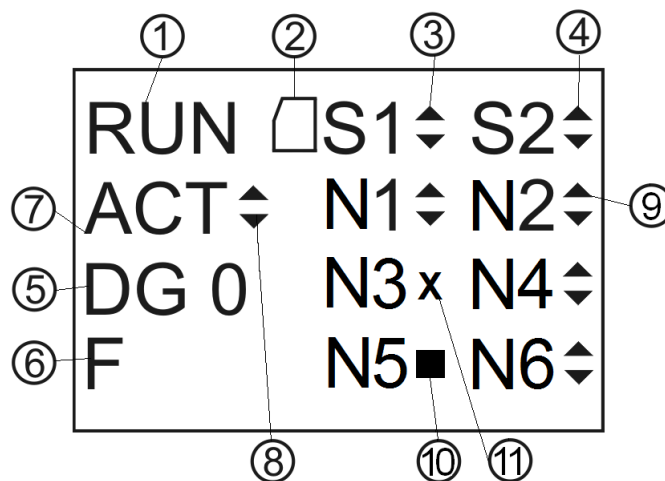

Figure 228: CPU HX3040 Status Screen

Legend:

- 1. Indication of the CPU status operation. In case the CPU application is running, the state is RUN. In case the CPU application is stopped, the state is STOP and, when is stopped in an application depuration mark, the state is BRKP. For further details, see CPU Operating States section.
- 2. Memory Card presence indication. Further details regarding its installation see Memory Card Installation section.
- 3. COM 1 traffic indication. The up arrow (▲) indicates data transmission and the down arrow (▼) indicates data reception. For further information regarding the COM 1 interface see Serial Interfaces section.
- 4. COM 2 traffic indication. The up arrow (▲) indicates data transmission and the down arrow (▼) indicates data reception. For further information regarding the COM 2 interface see Serial Interfaces section.
- 5. Indication of the CPU active diagnostics quantity. In case the number shown is different than 0 (zero), there are active diagnostics in the CPU. For further details regarding their visualization on the CPU graphic display, through diagnostic key, see One Touch Diag section.
- 6. Forced variables in the CPU indication. In case the "F" character is shown in the graphic display, a variable is being forced by the user, whether symbolic, direct representation or AT. For further information regarding variable forcing see Run Mode section.
- 7. Identification of the CPU redundancy state (message only valid in HX3040 in redundant mode). If the CPU is the active PLC, the ACT information will be presented. The other possible states are NCF (Not configured), STR (Starting), INA (Inactive) and SBY (Stand-by). For any CPU, BRKP indicates BreakPoint.
- 8. Indication that the project synchronization is being executed. The up arrow (▲) indicates project data transmission and the down arrow (▼) indicates project data reception. For further information about the project synchronization see Project Synchronization section.
- 9. Indication of traffic on the Ethernet (N) (NET 1 to NET 6) port. The up arrow (▲) indicates transmitting data and the down arrow (▼) indicates receiving data. For more information about Ethernet interfaces, see Ethernet Interfaces Configuration section.
- 10. Indication that the interface is part of a NIC Teaming pair and that it is in a standby state (stop). Remember that NIC Teaming pairs will always be N1/N2, N3/N4 and N5/N6. For more information about the configuration modes of Ethernet interfaces, see Ethernet Interfaces Configuration section.
- 11. Indication of network link failure (x) on the Ethernet port (N)(NET 1 to NET 6). For more information about Ethernet interfaces, see Ethernet Interfaces Configuration section.

Besides the characters described above, Hadron Xtorm CPUs can present some messages on the graphic display, correspondent to a process, which is being executed at the moment.

The table below presents the messages and their respective descriptions:

Message	Description
FORMATTING...	Indicates that the CPU is formatting the memory card.
FORMATTING ERROR	Indicates that an error occurred during the memory card formatting by the CPU.
WRONG FORMAT	Indicates that the format of the memory card is incorrect.
INCORRECT PASSWORD	Indicates that the password entered does not match the configured password.
TRANSFERRING...	Indicates that the project is being transferred.
TRANSFERRING ERROR	Indicates that an error occurred while transferring the project, caused by a problem with the memory card or by removing it during transfer.
COMPLETE TRANSFERRING	Indicates that the transfer was completed successfully.
TRANSFERRING TIMEOUT	Indicates that a time-out occurred (communication time expired) during the project transference.
INVALID CPU TYPE	Indicates that the CPU model is different from the one configured in the project within the memory card.
INVALID CPU VERSION	Indicates that the CPU version is different from the one configured in the project within the memory card.
APPLICATION CORRUPTED	Indicates that the application within the memory card is corrupted.
APPLICATION NOT FOUND	Indicates there is no application in the memory card to be transferred to the CPU.

altus

Message	Description
CRC NOT FOUND	Indicates that the application CRC does not exist.
MCF NOT FOUND	Indicates there is no MCF file in the memory card.
NO TAG	There is no configured tag for the CPU in the MasterTool Xtorm
NO DESC	There is no configured description for the CPU in the MasterTool Xtorm.
MSG. ERROR	Indicates that there is (are) error (s) on diagnostics message (s) of the requested module (s).
MISSING SIGNATURE	Unexpected problem with the product. Get in contact with Altus Technical Support
APP. ERROR RESTARTING	Application error. Runtime is restarting the application.
APP. NOT LOADED	Indicates that the runtime will not load the application.
LOADING APP.	Indicates that the runtime will load the application.
WRONG SLOT	Indicates that the CPU is in an incorrect position in the rack
FATAL ERROR	Indicates serious problems in the CPU startup (e.g. CPU partitions were not properly mounted). Contact Altus customer support.
HW-SW MISMATCH	Indicates that Hardware and software mismatch due an unexpected problem with the product. Contact Altus customer support.
UPDATING FIRMWARE	Indicates the firmware is being updated in the CPU.
RECEIVING FIRMWARE	Indicates the updating file is being transferred to the CPU.
UPDATED	Shows the firmware version updated in the CPU.
UPDATE ERROR	Indicates that an error occurred during the CPU firmware updating due communication failure or configuration problems.
REBOOTING SYSTEM...	Indicates the CPU is being restarted for the updating to have effect.

Table 249: HX3040 Graphic Display Further Messages

7.2. System Log

The *System Log* is an available feature in the MasterTool IEC XE programmer. It is an important tool for process control, as it makes it possible to find events on CPU that may indicate error conditions, presence of active components or active diagnostics. Such events can be viewed in chronological order with a resolution of milliseconds, with a storage capacity of up to one thousand log entries stored in the CPU internal memory, that can't be removed.

In order to access these Logs, just go to the *Device Tree* and double-click on *Device*, then go to the *Log* tab, where hundreds of operations can be seen, such as: task max cycles, user access, online change, application download and upload, application synchronization between CPUs, firmware update between another events and actions.

In order to view the *Logs*, just need to be connected to a CPU (selected Active Path) and click on ⟳ . When this button is pressed the Logs are displayed and updated instantly. When the button is not being pressed the Logs will be hold in the screen, it means, these button has two stages, one hold the logs state being updated and in the state the updating is disabled. To no longer show the Logs, press ✕ .

It is possible to filter the Logs in 4 different types: warning(s), error(s), exception(s) and information.

Another way to filter the messages displayed to the user is to select the component desired to view.

The Log tab's *Time Stamp* is shown by MasterTool after information provided by the device (CPU). MasterTool can display the Time Stamp in local time (computer's time) or UTC, if *UTC time* checkbox is marked.

> **ATTENTION**
>
> If the device's time or time zone parameter are incorrect, the Time Stamp shown in MasterTool also won't be correct.

For further information about the System Logs please check the MasterTool IEC XE User Manual – MU299609 and the RTC Clock and Time Synchronization subsection of this manual.

> **ATTENTION**
>
> The system logs of the CPUs, starting in firmware version 1.4.0.33 (Nexto) and 1.14.36.0 (Xtorm), are reloaded in the cases of a restart of the CPU or a reboot of the Runtime System, that is, it will be possible to check the older logs when one of these situations occurs.

7.3.   I/O Modules Diagnostics

The tag access Functions, Modules, and I/O points description/diagnostics are closely linked. There are three main components that are widely referenced in this section, they are:

1. CPU graphic display
2. Module display to be accessed
3. Module diagnostic button to be accessed

7.3.1.   Access to diagnostic mode

Regardless of what is being shown on the CPU display, after a short press in the module diagnostic button will be shown on the CPU display the tag and active diagnoses of the respective module. These data will be shown on the CPU display in the order indicated in the figure below.
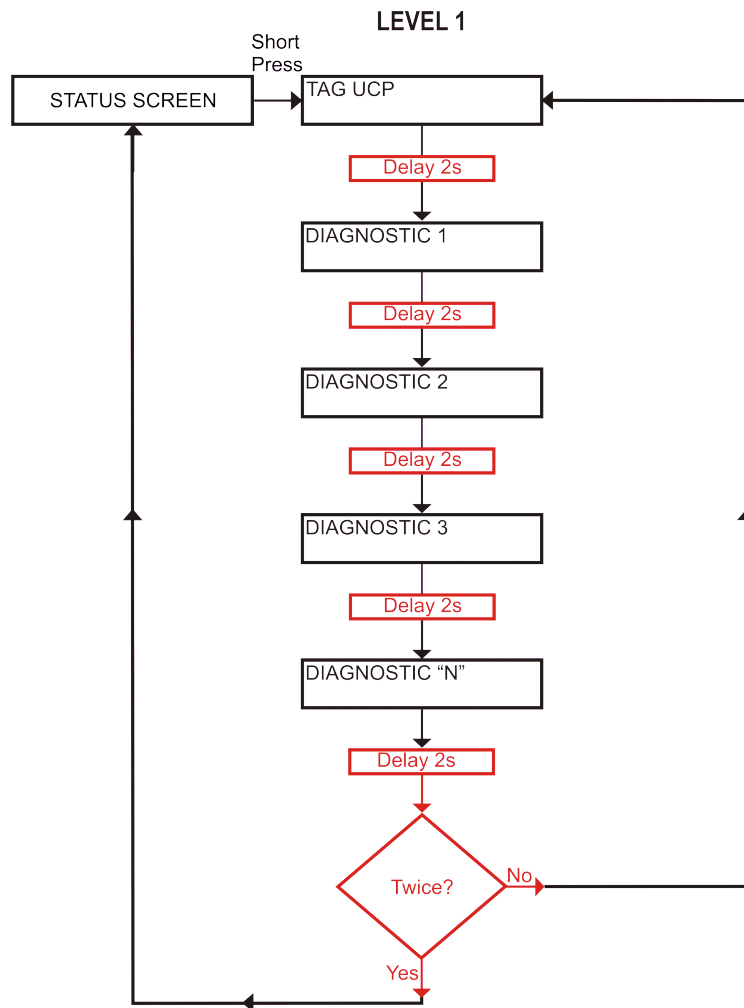

Figure 229: CPU Diagnostics View

As illustrated in the figure above, both the tag and the list of all active diagnostic related to the module will be shown twice on the CPU display. After that the respective module will exit the diagnostic mode and the CPU display will indicate information regarding the CPU.

It is possible to identify that a particular module is in diagnostic mode when the segments shown in the figure below (module display) blink. The number of segments depends on the number of I/O points of the module.

Figure 230: Module Display

7.3.2. Accessing I/O points

After entering the diagnostic mode, the next short pressing in the module diagnostics button selects the first I/O point, this time the display will clear indication of individual active diagnostics (explained above) and will indicate the selected I/O point. For the selection of the next I/O point execute a new short pressing in the respective module diagnostic button. When the last I/O point is selected a new short press on the button will terminate the diagnostic mode.

By accessing an I/O point, the CPU will display the respective I/O point tag and all its active diagnostic in the same manner as indicated in the flowchart of figure 229.

For modules that do not have I/O points, a new short press on the diagnostic button, after entering the diagnostic mode, will finalize the operation.

7.3.3. Accessing the module and I/O points description

In addition to the tag, modules and I/O points can have a description. The description is used when there is a need to add extra information beyond the tags for a given module or I/O point, for example: "Temperature reading module" for describing a module or "Main pump drive" to describe a particular output point. The limit for the tag, the tag description or module name is 255 characters.

To change the name and description of each module inserted in the application, click the right button on the module, under *"Properties"* in the *"Common"* tab, change the name or description (limited to 255 characters).

To change the tag, go to the tab *"Bus I/O Mapping"* of each module, and double-click the column table to add the point tag. To change the description of the tag, do the same in the column intended for description tag, and enter the desired information.

To access the description, just perform a long press on the diagnostic button when the module is indicating the tag or active diagnostics of the module. In this case the CPU will display the module description. Likewise, when selecting an I/O point and a long press is performed in the diagnostics button, the description of the I/O point appears on the CPU display.

ATTENTION

It is recommended the use of alphanumeric characters for the tag and module name (uppercase or lowercase and the text does not start with numbers). In the field "description" use alphanumeric characters (uppercase or lowercase), blank and period (".").
The use of characters other than those mentioned above is not recommended

---

ATTENTION

When using *ETD - Electronic Tag on Display* function or *Web Server* to display the tag of I/O points, the tag name is truncated in the first 24 characters after the string *"Application."* of the tag name. Ex. For the tag "Application.UserPrg.MyTest.ON", only the string "UserPrg.MyTest.ON" will be displayed on the graphic display.

> **ATTENTION**
>
> When using *ETD - Electronic Tag on Display* function or *Web Server* to view the description of the tag of I/O points, the field will be truncated after the first 48 characters of the tag description.

> **ATTENTION**
>
> When using *ETD - Electronic Tag on Display* function or *Web Server* to view the module name, the field will be truncated after the first 24 characters of the module name description.

7.3.4. Short and long press

The table below indicates the times.

Touch type	Minimum time	Maximum time	Indication condition
No touch	-	59,99 ms	-
Short touch	60 ms	0,99 s	Pressing and releasing the button within the set period
Long touch	1 s	20 s	Pressing and releasing the button within the set period
Locked Switch	20,01 s	(∞)	Pressing the button for more than 20 seconds

Table 250: One Touch Time

7.4. Not Loading the Application at Startup

If necessary, you can choose not to load an existing application on the CPU during its startup. For that, energize the CPU via diagnostics button, by pressing it for at least two seconds. During the CPU startup the display shows a message warning that the application will not be loaded and will go into Stop Mode. In case of a login, MasterTool Xtorm software indicates that there is no application on the CPU. For reloading the application, reboot the CPU or download a new application.

7.5. Power Supply Failure

The Hadron Xtorm Series power supply has a failure detection system, according to the levels defined in its technical characteristics (see Power Module Technical Characteristics).

1. If the power supply is energized with voltage lower than the minimum required limit, a power failure diagnosis will be generated, which will be recognized by the CPU and the message *"POWER FAILURE"* will appear on its display. When the power supply is within the established limits, the CPU recognizes it and automatically restarts with the user application. The diagnostic will still be active to show the user that the last initialization suffered a power supply failure.
2. If the power supply has a voltage drop below the required minimum threshold and returns to a higher value within 10 ms, the power supply failure will not be recognized by the CPU and no diagnostics will be generated, since the system remains intact during this time.

POWER
FAILURE

Figure 231: Power Supply Failure Message

7.6. Common Problems

If, at power on the CPU, it does not work, the following items must be verified:

- Is the room temperature within the device supported range?

- Is the rack power supply being fed with the correct voltage?
- Is the power supply placed on the far left in the rack (front view)? Is it followed by the Hadron Xtorm Series CPU? In case the architecture is built with redundant sources, the far left module in the rack is the power supply, followed to its right by another one. In the next slot on the right, the Hadron Xtorm Series CPU is placed
- Are there network devices, as hubs, switches or routers powered, interconnected, configured and properly working?
- Is the Ethernet network cable properly connected to the NET 1 to NET 6 ports of the Hadron Xtorm CPU and to the network device?
- Is the Hadron Xtorm Series CPU on, in execution mode (Run) and with no diagnostics related to hardware?

If the Hadron Xtorm CPU indicates execution mode (Run) but it does not respond to the requested communications, whether through MasterTool Xtorm or protocols, check the following items:

- Is the CPU Ethernet parameters configuration correct?
- Is the respective communication protocol correctly configured in the CPU?
- Are the variables, which enable the MODBUS/DNP3 relations properly, enabled?

If no problem has been identified, consult the Altus Technical Support

7.7. Troubleshooting

The table below shows the symptoms of some problems with their possible causes and solutions. If the problem persists, consult the Altus Technical Support.

Symptom	Possible Cause	Solution
Does not power on	Lack of power supply or incorrectly powered.	Verify if the CPU is connected properly in the rack. Power off and take off all modules from the bus, but the power supply and the CPU. Power on the bus and verify the power supply functioning, the external and the one in the rack. Verify if the supply voltage gets to the Hadron Xtorm power supply contacts and if is correctly polarized.
Does not communicate	Bad contact or bad configuration.	Verify every communication cable connection. Verify the serial and Ethernet interfaces configuration in the MasterTool Xtorm software.
Does not recognize the memory card	Bad connection or not mounted.	Verify if the memory card is properly connected in the compartment. Verify if the memory card was put in the right side, as indicated on the CPU frontal panel. Verify if the memory card wasn't unmounted through MS button, placed on the frontal panel, visualizing the indication on the CPU graphic display.

Table 251: Troubleshooting

7.8. Preventive Maintenance

- It must be verified, each year, if the interconnection cables are connected firmly, without dust accumulation, mainly the protection devices.
- In environments subjected to excessive contamination, the equipment must be periodically cleaned from dust, debris, etc.
- The TVS diodes used for transient protection caused by atmospheric discharges must be periodically inspected, as they might be damaged or destroyed in case the absorbed energy is above limit. In many cases, the failure may not be visual. In critical applications, is recommendable the periodic replacement of the TVS diodes, even if they do not show visual signals of failure.
- Bus tightness and cleanness every six months.

8. Electric Panel Design

8.1. Mechanical Design

8.1.1. Dimensions

The dimension of each Hadron Xtorm Series module can be found in the Technical Characteristics document of each module. The documents are listed in table 1.

The dimensions of the main modules are shown below, in mm.

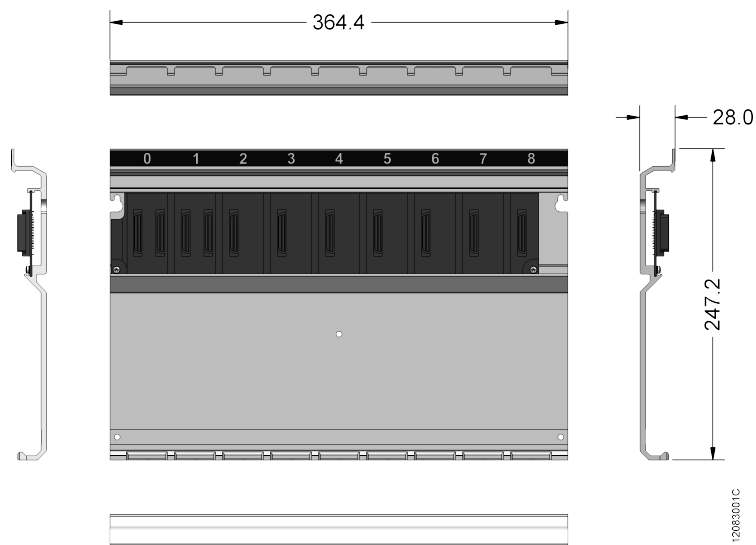#### 8.1.1.1. Blackplane Rack

8.1.1.1.1.   9-Slot Backplane Rack

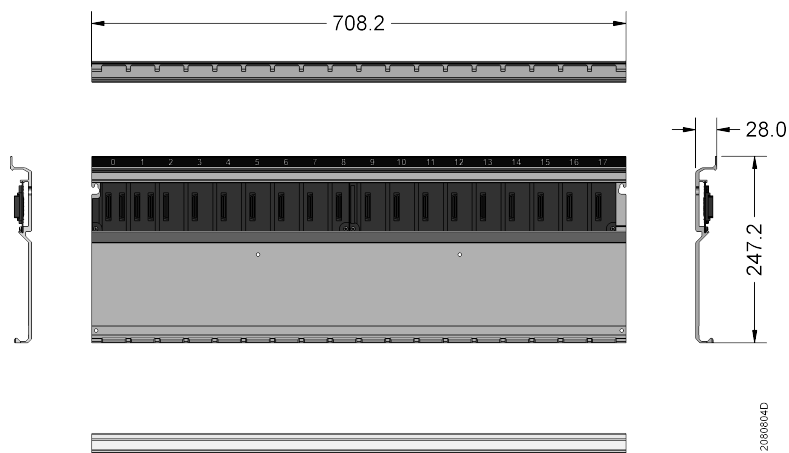Figure 232: 9-Slot Backplane Rack

8.1.1.1.2.   18-Slot Backplane Rack

Figure 233: 18-Slot Backplane Rack

altus

8.1.1.2. Modules

This module size is used by all other Hadron Xtorm Series modules. The figure illustrates the HX3040 CPU.
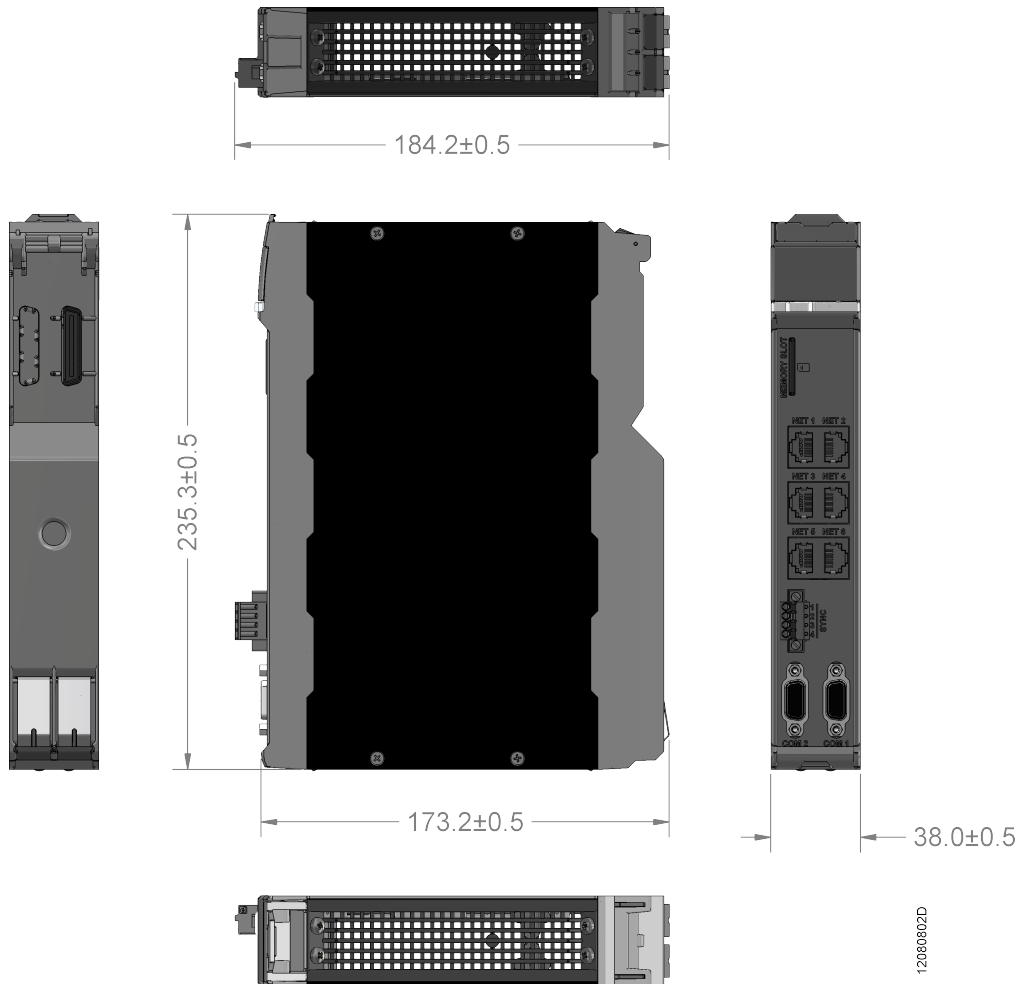
Figure 234: Module

8.1.2. Depth of Rack-mounted Module

The depth of the Hadron Xtorm Series module and rack set can be obtained by adding 28 mm to the depth of the module. In the example of the figure below, a standard module with 184.2 mm depth was used. When considering the rack, the depth of the set is 212.2 +/- 0.5 mm.
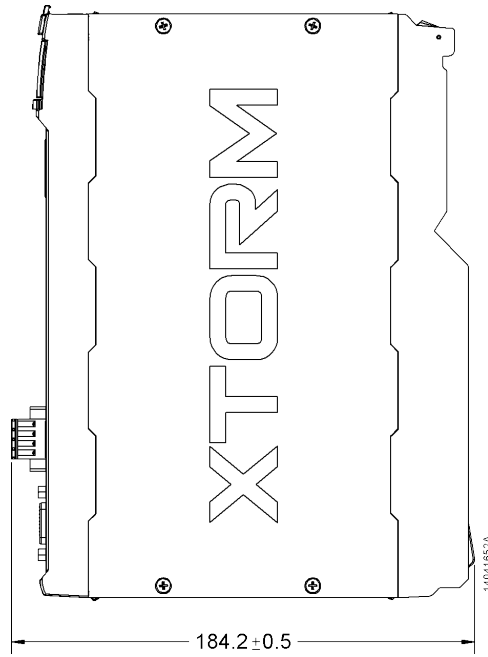
Dimension in mm.

Figure 235: Depth of Rack-mounted Module

8.1.3. Spacing between modules and other equipment in the panel

The RTU requires free spaces around it. This is necessary to enable the correct handling of the equipment. Furthermore, such spacing should be respected in order to allow the air flow through the CPU, in convection type, maintaining the equipment temperature.

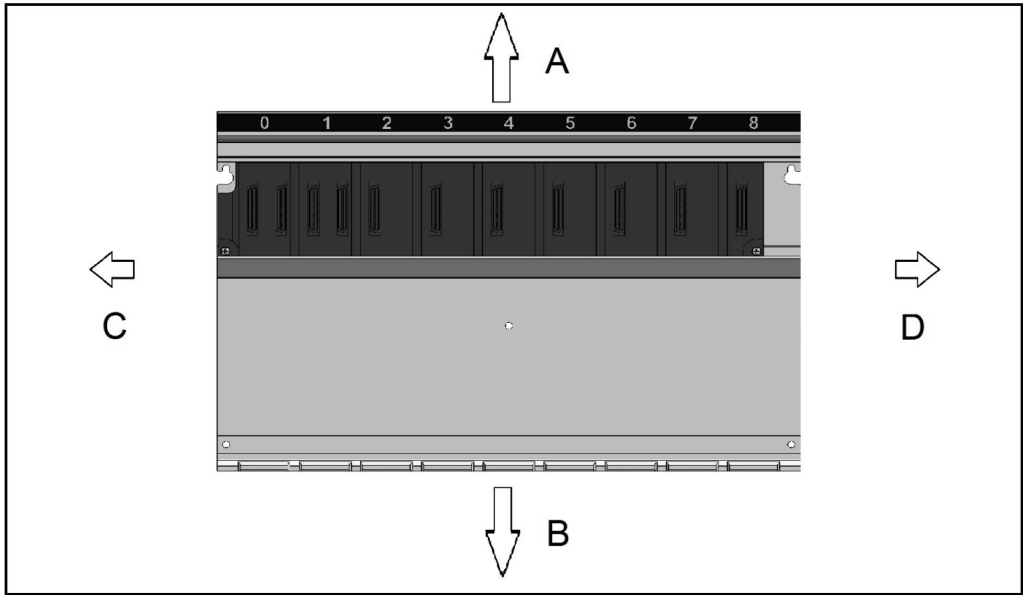The figure below and the following table indicate the spacing required by Hadron Xtorm Series modules.

Figure 236: Free space around the RTU

Dimension A	Dimension B	Dimension C	Dimension D
10 cm	10 cm	4 cm	4 cm

Table 252: Dimensions of free space around the RTU

8.1.4. Gutter Sizing

For the sizing of the gutter, in addition to the area occupied by the wires, observe the internal heating in the gutter, caused by the heat dissipated by the wires, which can lead to a reduction in the gutter area occupancy.

Use the following rule: gutter area >= sum of the wiring area / 0.4

Wiring area = $(3,14 * radius^2)$

It is considered as "wiring area" the total area, including the insulation.

8.1.5. Horizontal/ Vertical Mounting

Hadron Xtorm Series allows the use of RTU in horizontal orientation. Vertical mounting is not allowed in the rack.

8.2. Thermal Design

Altus equipment is designed to work at an ambient temperature of up to 60°C (except where specified). Therefore, this should be the maximum internal temperature of the cabinet. The following precautions should be observed in panel design:

- Size cabinets with enough internal volume for good air circulation.
- Provide forced ventilation or air exchangers with the external environment, if necessary, in order to avoid over temperature. In critical cases, it is recommended the use of refrigeration, to keep the equipment operating within the levels of operating temperature.
- Distribute homogeneously heat sources inside the cabinet.
- Consider the dissipation in the cables leading higher currents to avoid internal overheating into the gutters.

> **ATTENTION**
>
> For maximum dissipation of each module of the Nexto series, see the Technical Characteristics document for the specific module.

A method to calculate the internal temperature of the panel depending on their power dissipation is explained below.

8.2.1. Heat dissipation in an electrical panel

Each electrical panel dissipates, through its surface, an amount of heat for a given difference in internal and external temperature. To calculate the heat dissipation in cases where the temperature difference between the inside and the outside of the panel reaches 50 °C, the following values should be considered:

- Effective dissipation surface of the panel; calculated according to DIN-VED 0660 chapter 500, as indicated by the type of installation.
- The dissipation constant for steel sheet painted in W/m^2 °C.
- Panel ventilation conditions (installation location).
- Degree of inner panel occupation (inside air flow impedance).

For the values mentioned above, only the value of the panel surface can be calculated exactly.

Calculation of the effective surfaces of dissipation A (m^2) for the panel:

The calculation of surface *"A"* is done as directed by DIN-VDE, depending on the type of installation of the panel:

Installation type according to DIN-VDE 0660/500	Formula for calculation of A (m^2)
All side free panel	A = 1,8 * H * (L + P) + 1,4 * L * P
Pane with the rear surface obstructed	A = 1,4 * L * (H + P) + 1,8 * P * H
Panel with a lateral surface blocked	A = 1,4 * L * (H + L) + 1,8 * L * H
Panel with a lateral and rear surface obstructed	A = 1,4 * H * (L + P) + 1,4 * L * P
Panel with both sides obstructed	A = 1,8 * L * H + 1,4 * L * P + P * H
Panel with the two sides and the rear surface obstructed	A = 1,4 * L * (H + P) + P * H
Panel with the two sides and the rear and upper surface obstructed	A = 1,4 * L * H + 0,7 * L * P + P * H

Table 253: Calculation of Effective Surface of Dissipation

L = width (m), H = height (m), P = depth (m)

In the case of panels built with painted steel sheet, for the still air around them, the heat dissipation constant can be considered 5,5 W m^2 °C.

The power dissipated by a panel can then be calculated through equation Qs = k * A * (internal temperature – external temperature), or obtained from figure below.
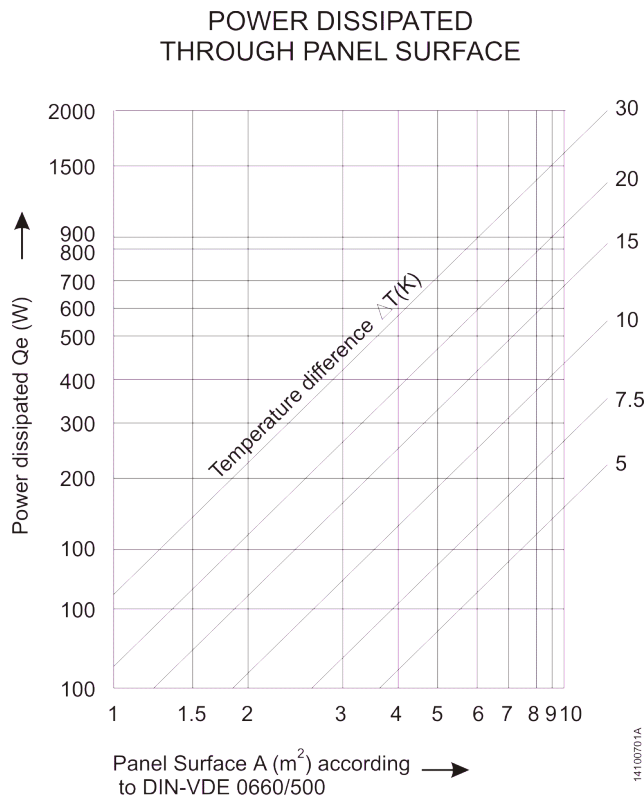

Figure 237: Dissipated Power x Surface x Temperature Diff.

This value can, however, be tripled in case of air circulation outside of the panel.

Air circulation in a panel is obstructed by the installation of equipment in its interior, leading to the formation of localized heating points. In this situation, the desired air circulation can be obtained by installing internal fans to the panel, increasing the air flow inside.

The forced circulation through fans inside the panel also brings an improvement in own convection and a tendency to temperature stabilization along the panel. Without the forced air circulation, a heat point at the top of the panel, due to convection, is observed.

Examples:

For a free panel from all sides, with effective area of 3,96 m^2, installed power of 350 W and external ambient temperature of 30 °C, calculate the internal average temperature.

Qs = k * A * (Ti – Te)

350 = 5,5 * 3,96 * (Ti – 30)

Ti = 46 °C

To the same panel, calculate the internal temperature for an installed power of 1000 W.

Qs = k * A * (Ti – Te)

1000 = 5,5 * 3,96 * (Ti – 30)

Ti = 76 °C

In this case, the temperature has exceeded the limit of the equipment operation (60 °C), and must be provided another way to remove excess heat. The limit of installed power for the internal temperature of 60 degrees is:

Qs = k * A * (Ti – Te)

Qs = 5,5 * 3,96 * (60 – 30)

Qs = 653 W, 653 W limit, the remaining 347 W (1000 W - 653 W) must be removed, for example, through an air conditioning equipment.

> **ATTENTION**
>
> In previous calculations, note that the internal temperature is always a medium temperature, and if there is no forced air circulation inside the panel, the temperature at the top of the panel will be greater than the base, and there may be localized heat points. Proper safety margin should be given in each case.

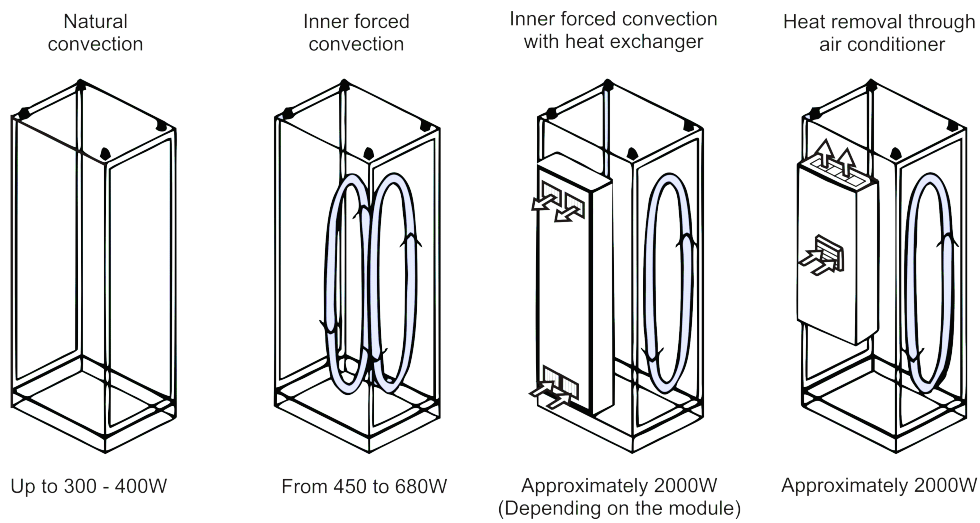| Natural convection	Inner forced convection	Inner forced convection with heat exchanger	Heat removal through air conditioner
Up to 300 - 400W	From 450 to 680W	Approximately 2000W (Depending on the module)	Approximately 2000W

Figure 238: Examples of Heat Movement – Closed Installation

A much greater heat dissipation, comparing with previously obtained, can be reached if is allowed the air exchange with the outside. Ventilation is normally performed with the ventilation louvers on the sides, on the door or on the back cover. This will, of course, reduce the degree of protection (IP) of the panel.
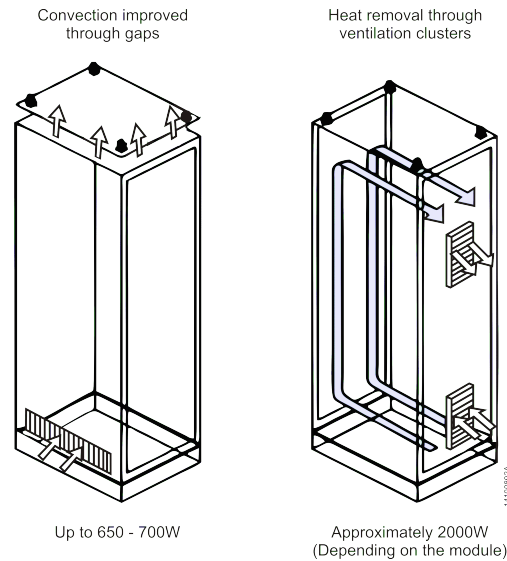
Figure 239: Example of Heat Movement – Open Installation

8.3. Electrical Design

8.3.1. General Information

Remote terminal units are manufactured in view of global standards that establish acceptable levels of environmental and noise conditions normally found in control applications of generation, transmission and distribution systems of electric energy, such as hydroelectric power plants (HPPs), and power substations.

It is essential that the installation of these products follow rules of convenient project, established by installation standards. Problems caused by electromagnetic interference (EMI), such as communication and program execution failures, noise in analog variables and even loss of program, can be caused by bad electrical design or installation.

The electrical design of the Altus RTUs must comply with the IEEE 518/1977, "Guide for Installation of Electrical Equipment to Minimize Electrical Noise Inputs to Controllers External Sources". The most important points are covered next.

8.3.2. Cabinet Power

The power supply of the control system must include main circuit breaker. We recommend the use of terminals for general power supply of mounting panel with integrated fuses, as well as the use of an outlet providing 127 or 220 Vac, for the programming terminal. The outlet must include ground pin to allow proper grounding of programming terminal. All the outlets of the cabinets must have clear indication of its voltages.

8.3.3. Distribution of Cables in the Cabinet

The correct distribution of cables in the cabinet and the correct grounding of the equipment ensure electromagnetic compatibility (EMC) of the installation.

It is important the proper distribution of the electric panel power supply, via distribution bars or connecting terminals.

From these general distribution points, a specific cable is used to each supply point. Local branches should be avoided in modules supplies, thus decreasing the cables pathways leading high-current.

For the best performance of your equipment, it is necessary to separate the circuits as to its type, in order to reduce electromagnetic interference, as follows:

1. AC power circuits and AC and DC loads drives
2. Low current input and output digital circuits (less than or equal to 1 A)
3. Analog circuits and communication

These circuits should be distributed preferentially in separate gutters or avoiding parallel cables disposal. A minimum distance of 150 mm is recommended between all I/O signals and power supplies greater than 500 V.

8.3.4. Cabinet Lighting

An internal lighting in cabinet, triggered by switch, to facilitate their operation is essential.

It is recommended the use of incandescent lamps for lighting, since fluorescent lamps can generate undesirable interference. If these are used, the following precautions must be taken in order to reduce the interference:
1. Put a grounded metallic fabric between the lamp and the cabinet, to reduce the noise emission;
2. Use shielded cables for lamp supply;
3. Protect the switch in metal casing and place filter in the supply network preferably next to lamp.

8.3.5. Grounding

A general terminal board or a ground bar in the cabinet is necessary for power supplies and modules grounding. This bar must be linked to a protective earth with low resistance.

8.3.6. Electromagnetic interference

The electromagnetic interference (EMI) is responsible for the vast majority of problems encountered on installed equipment.

These problems can significantly be reduced if the following precautions are taken in the design phase of the panel:
1. Distribute and arrange the cables in the gutters, avoiding mixing power cables with signal cables;
2. Inactive metal parts must be grounded in the cabinet;
3. In case of elements which cause noise emission, it is recommended to use shields;
4. Filter the panel power input.

8.3.7. Shielding

Generating sources of strong electromagnetic interference (transformers, motors, cables with high current or voltage) located inside the cabinet, should be covered by grounded metal sheets when located less than 50 cm from RTU electronic parts. Cables that go beyond the armored parts should be shielded or filtered.

The shielded cables inside the cabinet must be grounded in accordance with the instructions of each equipment.

8.3.8. Noise Suppressors

It is extremely important the connection of appropriate-sized noise suppressors directly in all inductive loads (relays, contactors, solenoids, etc.) triggered or not by RTU. The inductive loads activation generates electrical noise that can exceed the limits established by the regulations. The noises, if not mitigated in its origin, can reach the RTU, affecting its operation.

The protection circuits must be mounted close to the load. As a rule, should not be spaced more than 0.5 m. In the case of resistive loads (incandescent lamps, signaling LEDs, heating resistances, etc.), it is not necessary the use of these devices.

8.3.8.1. RC Circuit

The RC protection circuit (Resistor in series with a Capacitor) can be mounted in parallel with the contact or in parallel with the load. The assembly in parallel with contacts is recommended for loads supplied into DC voltage. The assembly in parallel with the load is recommended for loads supplied with continuous or alternating voltages. RC circuits are most effective when used at voltages above 100 V.

To select the values of R and C, it is recommended a resistor of 0.5 to 1 ohm for each 1 V (voltage), and a capacitor of 0.5 to 1 μF for each 1 A (current). For example, in case of a 220 V/1 A load, a 220 ohm resistor and 1 μF capacitor (the model of the capacitor must be appropriate to the type and value of the voltage of the load) can be used.
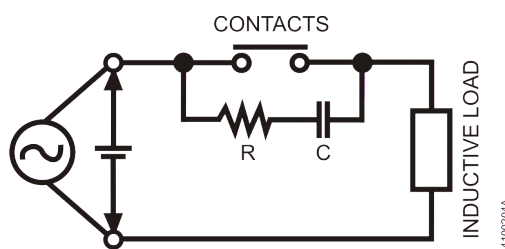

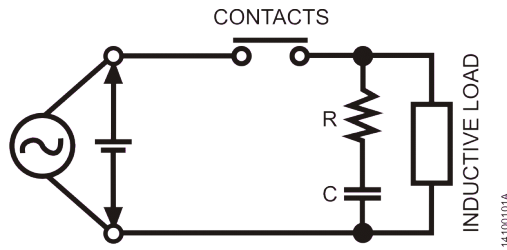Figure 240: Parallel RC Circuit with the Contacts

Figure 241: Parallel RC Circuit with the Load

> **ATTENTION**
>
> For 110/125 Vdc voltages, must be used the RC protection circuit in parallel with the load, under penalty of severe wear of the contacts. It is suggested the following values for R and C: 33 ohms and 470 μF, respectively.

8.3.8.2. Circuit with Diode

This is the most efficient way to limit the inductive circuit voltage at the time of disconnection. However, can bring problems as it increases the time to disarm if the load is, for example, a contactor or solenoid.
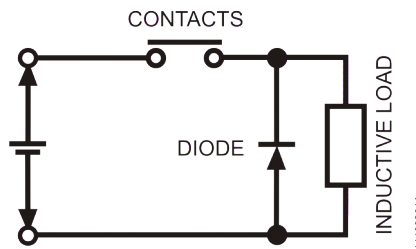
Figure 242: Diode Circuit

The circuit can only be used for DC voltages. Their recommended reverse voltage should be 10 times larger than the power supply and the current at least equal to the load.

8.3.8.3. Circuit with Diode and Zener

The circuit with diode and zener is suitable when the time to disconnect the circuit with diode is excessive. As well as the protection circuit with diode, it should only be used in DC voltages. The zener voltage should be slightly higher than the peak voltage of load.
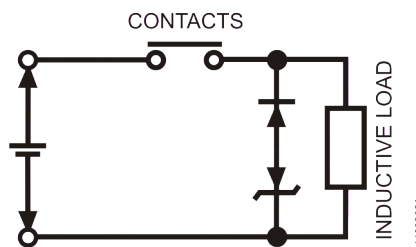

Figure 243: Zener Diode Circuit

8.3.8.4. Circuit with Varistor

The varistor circuit limits the inductive circuit voltage similarly to a zener diode. Your driving tension is generally greater than a zener diode and is bi-directional, allowing its use in DC or AC circuits, where it's used. Should be selected as the maximum voltage source, energy stored in the load and desired service life.

> **ATTENTION**
>
> It is imperative the complete reading of the Technical Characteristics of the products used prior to installation or use thereof. For choosing the type of suppressor to be used, it should be checked the type of load (DC or AC) and the voltage levels supported by the module chosen for the project.
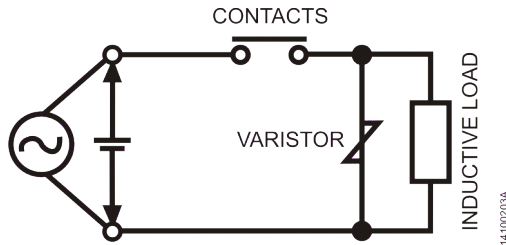

Figure 244: Varistor Circuit

8.3.8.5. Circuit with Capacitor

The circuit with capacitor is highly effective to suppress arcs generated during the opening of the contacts, but can cause wear of contact depending on the charge and discharge current of the capacitor. For selection of capacitor use the same rule of the RC circuit.

> **ATTENTION**
>
> This circuit is not recommended in most applications. Should be chosen only when the previous circuits show themselves inadequate.
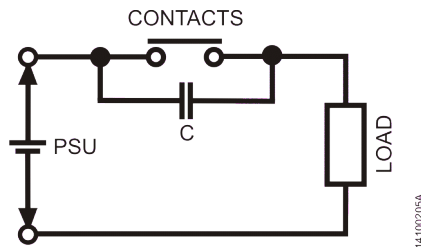

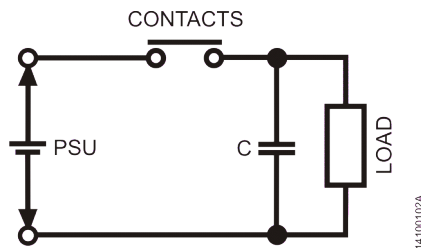Figure 245: Circuit with Capacitor in Parallel with Contacts

Figure 246: Circuit with Capacitor in Parallel with the Load

8.3.9. Distribution of Power Supply out of the Cabinet

In applications where the Cabinet is distant from the machine or the system to be controlled, although it is in the same building, the following procedures are recommended:

1. The conduct of cables from cabinet to the machine must be done in metallic conductors.

altus

2. The grounding of these ducts must be done every 20 meters.
3. Separate the wires into two groups for distribution in ducts:

   ▪ Digital signal cables up to 60 V, shielded cables leading analog signals and shielded cables with power supplies up to 230 V;

   ▪ Cables with voltage exceeding 230 V.

8.3.10. Lightning Protection

In outdoor applications, namely, in which the cables or communication lines of the CPU with the field signals come on out of the installation or triage open paths, one should consider the possible damage caused by lightning.

We recommend the use of varistors or arrestors (with inert gases) in these cables, for protecting the system against over voltages on these lines. Some shields are also required, as shown in the figure below.
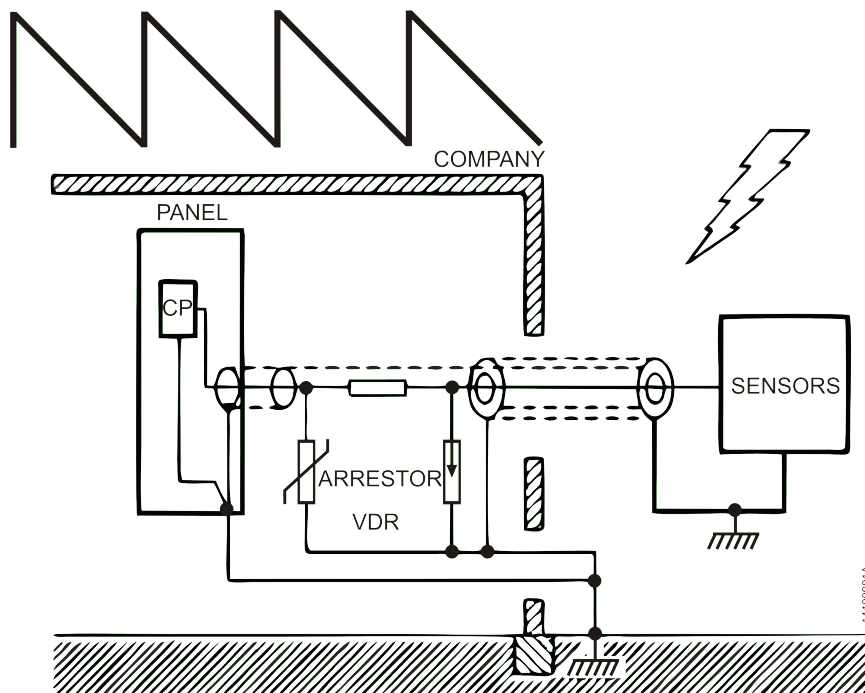

Figure 247: Lightning Protection

It is recommended that you install these protective devices at industry entrance or right next to the cabinet.

The figure above shows the correct way of installing lightning protection to a generic system. Each system has its own details of installation, so it is recommended to consider each case individually to define the best protection way.

In cases regarded as critical, consult the Altus support service.