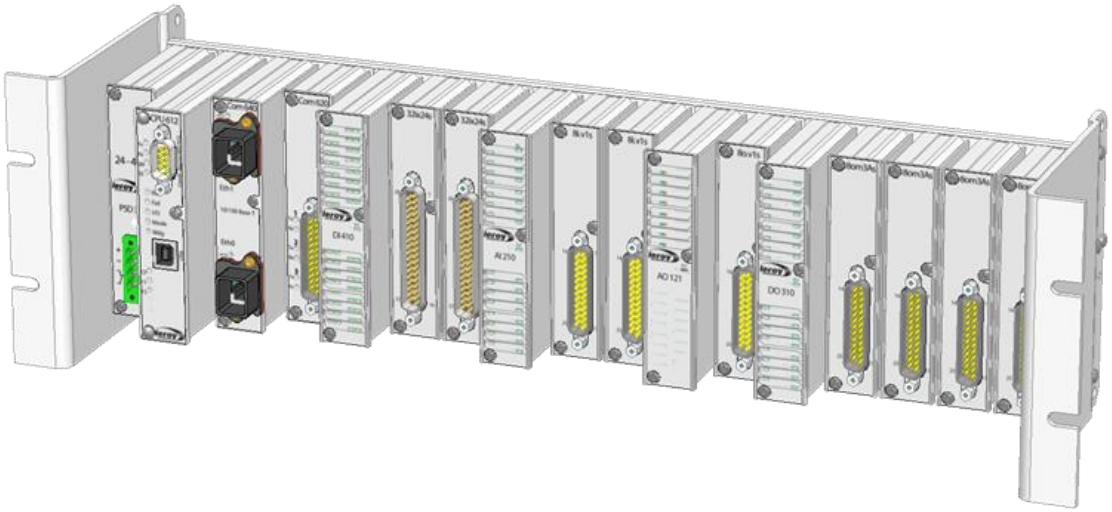
LT range

Modular Plate-form



Programming manual
STRATON
for CPU6xx central units

P DOC LT 006 E - V05

LEROY AUTOMATION 250 rue Max Planck 31670 LABEGE – France

This page is intentionally left blank

Overview

Congratulations on the purchase of your LT PLC.

CPU6xx is a central process unit programmable with Straton software. It contains a Straton Kernel. Straton is an IEC 61131-3 workbench developped by COPADATA.

The LT hardware implementation is explained in another manual available on our website.

In the rest of the document, the references "LT" and "LT Straton" denote a CPU6xx equipped with a Straton kernel.

Prerequisite

The developpement of STRATON applications requires the knowledge of programming in IEC61131-3 languages.

The actual implementation of the LT requires skills in electricity and industrial automation.

The equipment required is a development PC running Windows Seven or later, with an USB port or an Ethernet card.

Version

This documentation describes the features in the LT Straton BSP V2.3

Property

Contact

✉ Leroy Automation
250 rue Max Planck
31670 LABEGE
France

☎ +33 562 240 550

http://www.leroy-automation.com

Technical support:

☎ +33 562 240 546

mailto:support@leroy-autom.com

This page is intentionally left blank

Contents

This page is intentionally left blank

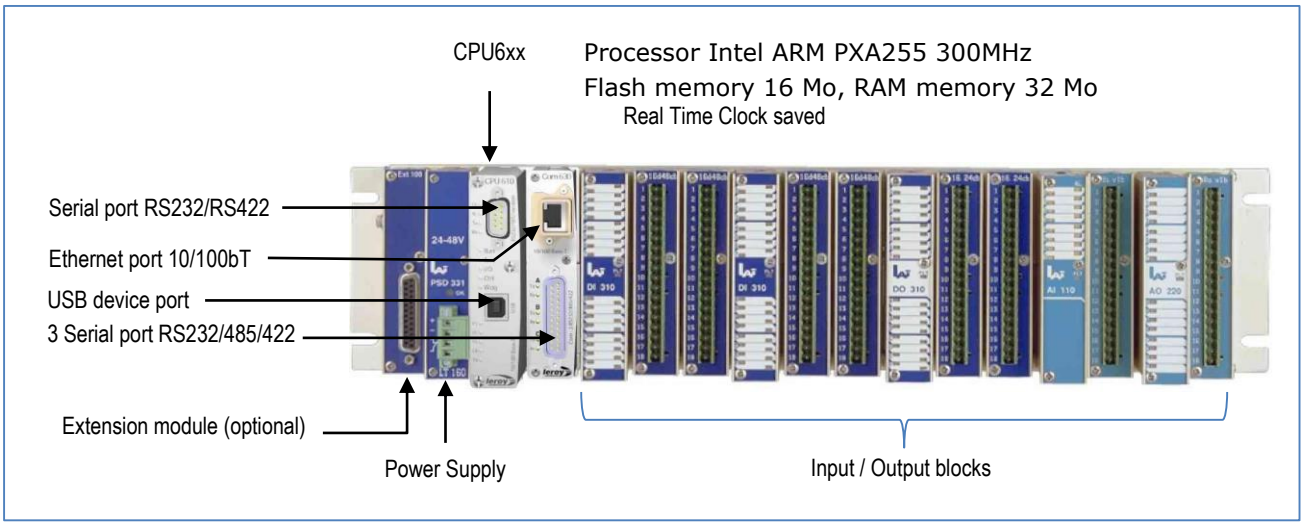
Chapter

1

General Overview

CPU6xx hardware

Straton operates on CPU6xx central process units. More information in the hardware user manual.



Embedded software

The CPU6xx contains a Linux 2.6.12 Operating System. The LT Straton Board Support Package (BSP) is a specific Linux distribution.
The Linux kernel and file system is the main element. It is the only interface between the system and hardware. Its essential functions are the task manager, memory management, and devices monitoring.
The libraries are the interface of applications launched automatically at startup.

After powering up the system, the first software running is U-boot: it performs the initialization of components on the CPU board (micro processor, clock, RAM and Flash component Ethernet ...), then performs the launch of Linux in RAM memory, and at the startup end of Straton virtual machine.
The CPU6xx is naturally programmable with the Straton workbench V8.1 or higher.

This page is intentionally left blank

Chapter

2

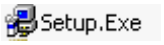
Quick start

Overview

This chapter describes all the operations necessary to implement and test a basic program in less than 20 minutes. We detail in this chapter the following steps:

- Installing the USB driver
- Installing Straton Workbench
- Creating a new project
- Target settings modification of an existing project
- I/O wiring
- Build, download and debug

Installing the USB driver



This driver enables the workshop to communicate with the USB port: this driver is running under Windows 7,8,10 (32 bits or 64 bits).

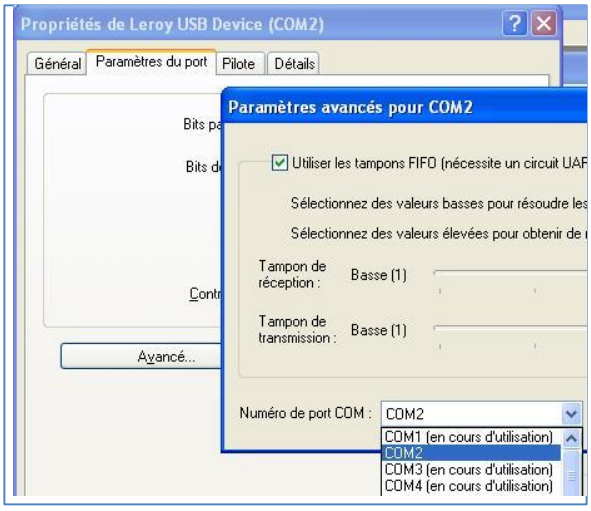
The file to run a setup file is shipped on CD ROM in the folder "Driver PC" file "LEROY USB DRIVER Setup.exe"

Installation procedure:

- IMPORTANT: The LT should not be connected with USB to the PC, during installation of the driver
- Run the file « LEROY USB DRIVER Setup.exe »
- Connect the USB port of your PC to the LT: a "beep" must be heard
- Identify the USB port com for LT in your PC: open the « configuration panel», then run the application « System », tab « Hardware », « device manager », « Ports (Com and LPT) » : note the number x of com : « Leroy USB Device (COMx)»
-

Note : the number of USB com port for the LT is assigned automatically by Windows.

Straton workbench requires to have a number strictly inferior to COM5. So, if it is superior or equal to COM5, you have to release a port of the first 4th serial ports, and in the advanced settings of the port « Leroy USB device », change your com port number and select the one that is not used, inferior to COM5.



Installing Straton

Installating of Straton workbench

Insert the Straton CD Rom in your PC, and then run the installation of the workbench. The recognition of your Straton USB donggle can be checked with the tool "License Manager": it can be run from the Windows start menu and then the menu "Straton/Licence".

Integration Leroy Automation LT Straton libraries to the workbench

Startup Straton Library Manager with the new link from start menu.

Open the menu "Tools/Import",

Select both two libraries files, present on the Leroy Automation CD-ROM "LT200_Straton_LNX_xx":

- "LT200_IO.XL5"
- "LT200_FUNCTION.XL5"

Creating a new project

Open the menu « File / New Project », then select an « Empty Project », fill in the name of your new project, and submit.

Open I/Os 

In the button bar click the icon for I/O wiring: the result is the opening of the editor of I/O Boards configuration, then dubble click on each line in order to add I/O boards:

Select in dropdown the CPU 6xx board.

Close the I/O wiring editor.

Build 

In the menu bar, click the icon « Build Project ». The result is the compilation of your project, with the following message appears in the Build window:
« 0 errors detected ».

Console configuration link PC ↔ LT

Open the menu « Tools/Communication Parameters » and select in the list if your choice is defined or define, depending on you wanted to establish the communication in USB or Ethernet:

- for USB communication : enter the USB port opened on your PC « com3 »
- for Ethernet communication : enter the IP address of the LT and the 502 TCP port :
  « xxx. xxx. xxx.xxx:502 » with xxx between 0 and 255
  for example : « 192.168.1.190:502 »

Download and Debug 

Click the « On Line » button in the main toolbar. Different cases can appear:

- the program inside the LT is different from the workbench program : it appears the window "Bad Version!", select stop and download the new version.
- there is another program running in LT : stop the application, and download.
  After downloading, the following message should appear: "Run".

To return to edit mode for your project, click a new time on the "On Line" button.

Chapter

3

CPU and I/O boards

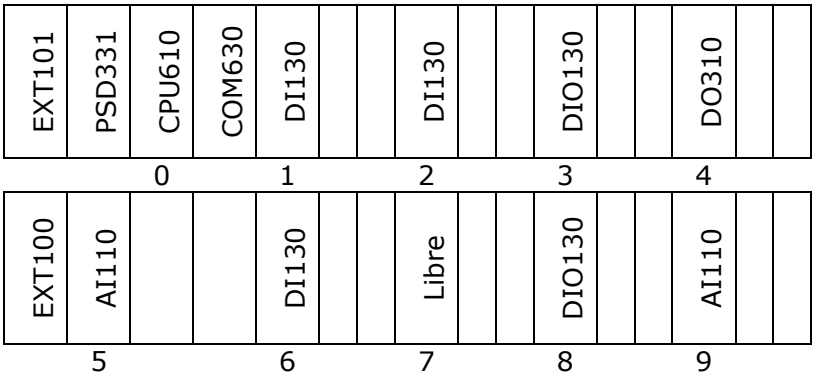
Overview

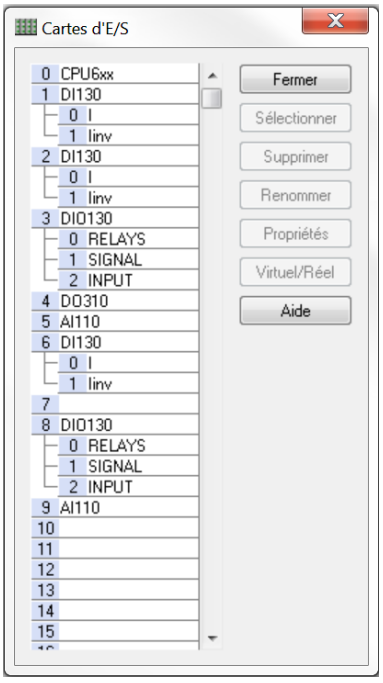
This chapter describes the board configuration of the LT.
We detail in this chapter the settings of CPU and IO boards:

- Hardware configuration matérielle
- CPU 6xx board : WdG parameters and Ethernet
- Digital I/O boards : DI310, DI410, DI312, DIO210, DO310, DI130, DIO130
- Analog I/O boards : AI110, AI210, AO121, AIO320
- I/O boards status

Open I/Os

Click on the menu « **Project** » / « **I/O wiring** » or on the corresponding button: the wiring editor appears.
16 boards maximum can be added. Each board will be identified with an « Device Index».
Device Index n° 0: reserved to the CPU 6xx.
Note: CPU communications ports are managed in Straton with the Fieldbus configuration tool.
Device Index n° 1 to 15: reserved to I/O boards.
The device index corresponds to the physical position of these boards on the main rack and extensions racks (2 maximum).
Example: hardware configuration on 2 racks, and associated software configuration:





CPU6xx board

Board parameters

- **Key** (readonly) : Internal identification code of the board. Value = 1
- **WDGLevel** : watchdog behavior if the Straton cycle time is over the WDGTimer
  - 0 (default): no watchdog.
  - 1 : led Fail set to True and IO boards set in local watchdog
  - 2 : led Fail set to True and Straton program set in mode « step by step »
  - 3 : LT reboot

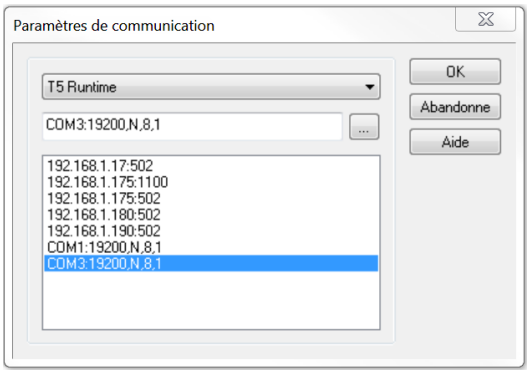
Wdg Once triggered, it is required to reboot the LT hardware (power off and power on).

- **WDGTimer** : watchdog value in ms ; minimal value : 100ms.
  - 0 (default): no watchdog.
  - >0 : Time maximum allocated for the cycle

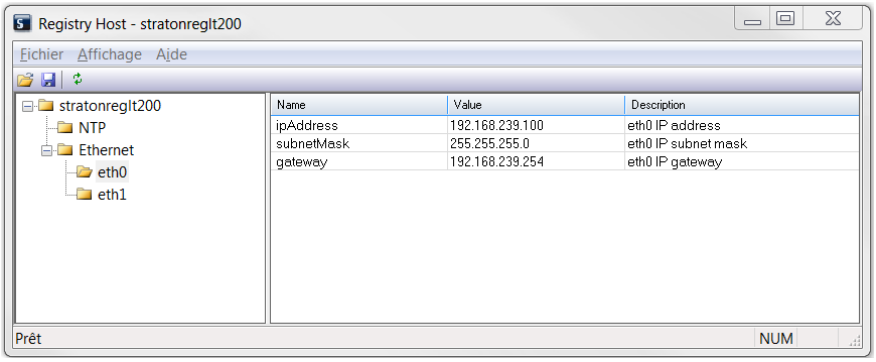
Ethernet parameters

You must set the Ethernet parameters with the "T5 registry" tool:

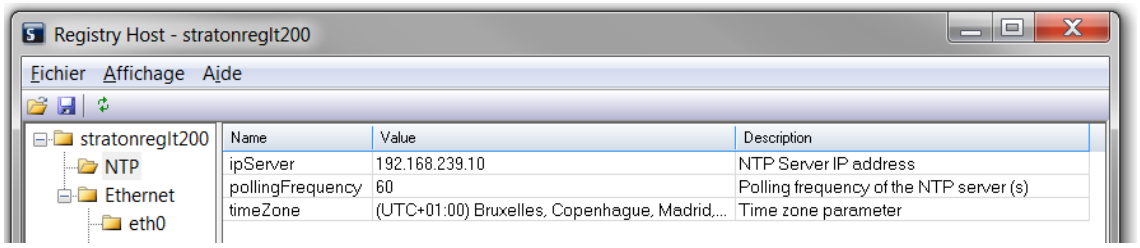
- open the menu "Tools/Runtime parameters/Monitor",
- and then select the communication settings in order to connect you to the LT:
- Example : « COM3 :19200,N,8,1 » then click "OK"



The "Registry host" tool reads the current parameters in the CPU.



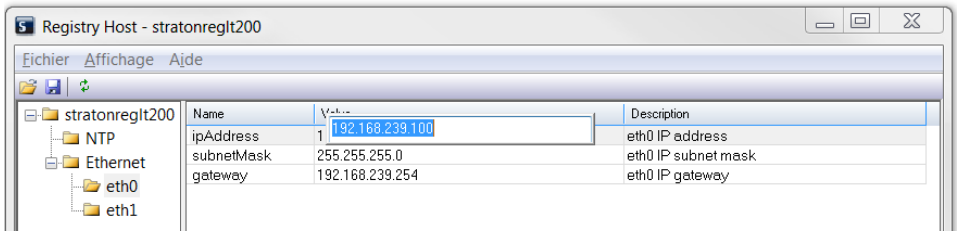
The NTP setting group allows the time setting of the CPU with the Network Time Protocol.



NTP Settings: By default the fields are empty, and the NTP function is not active.

- ipServer: IP address of the NTP server on a TCP / IP network that will provide the UTC date and time.
- pollingFrequency: polling frequency of the NTP server.
- timeZone: installation time zone of the LT.

Ethernet Group



Parameters **eth0**: Ethernet parameters for the first Ethernet port of the LT.

- **IP address** of LT on a TCP/IP network

By default IP address is « NULL ». In this case, the LT ignores the other parameters and needs a BOOTP or DHCP server : if it exists on the network, it will send an IP address to the LT.

Format : xxx.xxx.xxx.xxx with xxx [0..255]

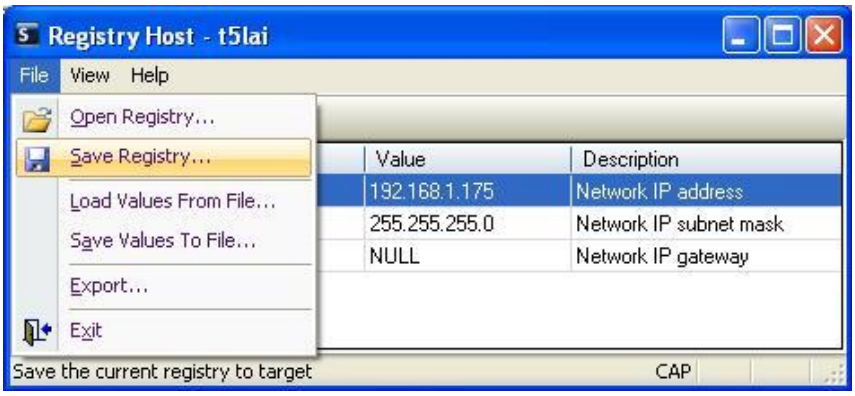
- **SubNetMask** : subnet mask used to show the breakdown of the IP address into sub-network address and device address on the sub-network. By default, this address is « NULL ».

  Format : xxx.xxx.xxx.xxx with xxx [0..255]

- **Gateway**: IP address of the gateway on the network. If the LT wishes to communicate outside the network to which it belongs, it must address this gateway. By default, this address is « NULL ».

  Format : xxx.xxx.xxx.xxx with xxx [0..255]

When you have finished all your settings, you have to save the new settings trough the menu "File" and "Save Registry":



The new Ethernet settings will be considered only on LT restart.

Leds F1 & F2

Two functions allow you to manage as you want both CPU leds called F1 and F2.

Function	**LEDF1**
Action	Monitor the led named F1
Parameters	(Bool) : FALSE=OFF ; TRUE=ON
Returned Value	(SINT) : function status [0..FFh] Status = 0 : function succeeded. Status <> 0 : function failed.
Example	*Status := LEDF1(TRUE);* (*the F1 led lights up*)

Function	**LEDF2**
Action	Monitor the led named F2
Parameters	(Bool) : FALSE=OFF ; TRUE=ON
Returned Value	(SINT) : function status [0..FFh] Status = 0 : function succeeded. Status <> 0 : function failed.
Example	*Status := LEDF2(TRUE);* (*the F2 led lights up *)

DI310 board: 32 digital inputs

The 32 green LED on the front panel indicate the status of the inputs.
Software notation : ***%IXi.j***

- "i" index is the logical position of the block on the base : 1 ≤ i ≤15.
- "j" index is the logical position of the input in the block : from 0 (Input 1) to 31 (Input 32)

DI410 board : 64 digital inputs

The 64 green LED on the front panel indicate the status of the inputs.
Software notation : ***%IXi.j***

- "i" index is the logical position of the block on the base : 1 ≤ i ≤15.
- "j" index is the logical position of the input in the block : from 0 (Input 1) to 63 (Input 64)

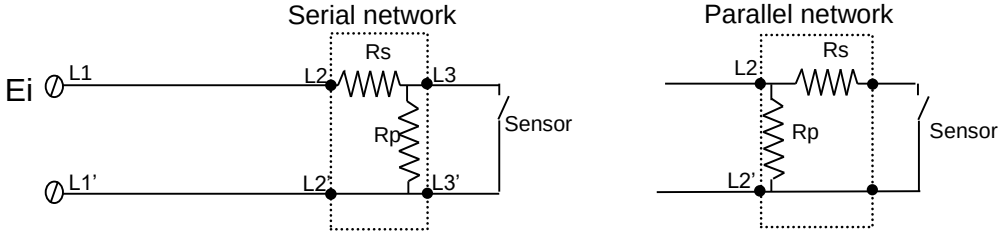
DI312 board: 32 wiring control digital inputs

The DI312 module can detect

- if the sensor is normally opened (NO),
- if the sensor is normally closed (NC),
- if the line between the PLC input (L1) and the Sensor (L2) is failed,
- if the line between the PLC input (L1') and the Sensor (L2') is failed.

« Failed » means Open circuit or Short circuit.

To detect the 4 states, it is necessary to place 2 resistors Rs and Rp near the sensor.
Two networks are possible (see below).

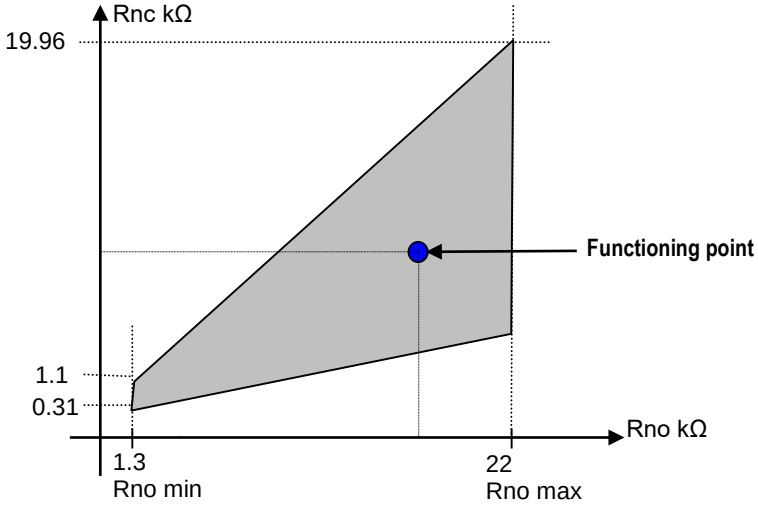


Sensor	Equivalent resistor	Serial network	Parallel network
opened	Rno =	Rs + Rp	Rp
closed	Rnc =	Rs	(Rs x Rp) / (Rs + Rp)

Rs and Rp values

Rs and Rp must be selected so that the point (Rno,Rnc) is included in the below grey polygon.

CAUTION : Line resistance = resistance between L1 and L2 + resistance between L1' and L2'.



Current in the sensor = I (mA)= 22 / (1 +Rno) with Rno in kΩ

Example

We want I = 3mA (reasonable value)
Rno= (22/I) - 1 = (22/3) – 1 = 6.333kΩ
Rnc can be between 1k and 5.6k .
We choose the serial network , Rs = Rnc = 3.3k (standard value)
Rp = Rno-Rs = 6.3 – 3.3 = 3k.
We select Rp = 3.3k (standard value) so I = 22/(1+6.6) = 2.9 mA

Board parameters

We must call the function DI312Config in the init of the program.

Function	**DI312CONFIG**
Example	*StatusDI312_3 := DI312CONFIG (3, 16#FFFFFF90, 2000, 1000, 100);* (* carte emplacement 3, contrôle sur les entrées n°0, 1, 2, 3, 5, 6, montage résistances en série de 1kΩ*)
Parameters	➢ BoardOrder (USINT) : board position [1..15] ➢ Mask (UDINT) : 32-bit mask for the wiring check of the 32 inputs. The wiring check is active on the input n if the bit of order n is 0. ➢ RCNO (UDINT) : only one value for all inputs ➢ RCNF (UDINT) : only one value for all inputs ➢ Rligne (UDINT) : only one value for all inputs
Value	(SINT) : function status [0..FFh] • Status = 0 : succes • Status <> 0 : error.

The DI312 boards consists of 2 sub-cards :

- INPUT: 32 inputs
- FAULT: 32 alarms

The 32 green LEDs on the front panel indicate the input status (green=ON).
The 32 red LEDs indicates the fault status (OK/red=KO)

Input status (Green LED)	Alarm (red LED)	Description
0 (off)	0 (off)	sensor normaly open
1 (on)	0 (off)	sensor normaly closed
0 (off)	1 (on)	Input not connected or short circuit to 0V
1 (on)	1 (on)	Short circuit to +V

IEC61131-3 software notation: ***%IXi.0.j and %IXi.1.j***

- "i" index is the logical position of the block on the base : 1 ≤ i ≤15.
- "j" index is the logical position of the input in the block : from 0 (Input 1) to 31 (Input 32)
- "0" is the sub-board of inputs
- "1" is the sub-board of alarms

DO310 board : 32 digital outputs

The 32 green LEDs on the front panel indicate the output status (green=ON).

Notation : %QXi.j avec :

- "i" index is the logical position of the block on the base : 1 ≤ i ≤15.
- "j" index is the logical position of the output in the block : from 0 (Input 1) to 31 (Input 32)

DIO210 board: 16 digital inputs + 8 digital outputs

This board containts 2 sub-boards

- 16 digital inputs
- 8 digital outputs

Notation:

- Inputs : %IXi.**0**.j with with j=[0..15] = logical position of the input in the block
- Outputs : %QXi.**1**.j with j=[0..7] = logical position of the output in the block

- "i" index is the logical position of the block on the base : 1 ≤ i ≤15.

Signalisation :
The 16 green LEDs on the front panel indicate the input status (green=ON).
The 8 red LEDs on the front panel indicate the output status (red=ON).

DI130 board: 16 secure digital inputs

The DI130 module consists of 2 sub_boards:

- Sub-board 0 : 16 Boolean inputs (signals)
- Sub-board 1 : 16 complementary Boolean inputs (complementary signals)

Each input is read trough 2 independent electronic circuits in order to detect an internal failure of one of both circuit.

Notation:

- I = [1..15] = logical position of the block on the base
- J = [0..15] = logical position of the input [1..16] in the block
- Inputs : %IXi.**0**.j : sub board 0

- Complementary Inputs : %IXi.**1**.j : sub-board 1

DIO130 board : secure digital inputs / outputs

The board is composed of 3 sub-boards:

- RELAYS : 16 Boolean outputs (remote controls)
- SIGNAL : 4 Boolean outputs (LEDs)
- INPUT : 8 Boolean inputs (remote signaling)

The board is divided in 4 control/command groups. Each group allows to monitor an electric device, like a switchgear, and then is equiped with:

- 2 relays : each relay x=[1..8] must be command by 2 boolean outputs : Rx and Rxs
- 2 inputs :
  - one for the close position (I1,I3,I5,I7): the corresponding led is red.
  - one for the trip position(I2,I4,I6,I8): the corresponding led is green.
- 1 led : signaling led (Sa, Sb, Sc, Sd)

Notation :

- i=[1..15] = the logical position of the block on the base
- Relays commands : **%QXi.0.j** with j index in the table below :

J	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Relay	R2	R4	R6	R8	R2s	R4s	R6s	R8s	R1	R3	R5	R7	R1s	R3s	R5s	R7s

- LED signals : **%QXi.1.j** with j index in the table below :

J	0	1	2	3
LED	Sa	Sb	Sc	Sd

- Inputs : **%IXi.2.j** with j index in the table below :

J	0	1	2	3	4	5	6	7
Input	I2	I4	I6	I8	I1	I3	I5	I7

AI110 board: 8 analog inputs

It is made up of 1 board: 8 analog inputs are available.
8 green LEDs on the front panel. A LED by input ; the LED is ON if the input is monitored.

Notation : **%IWi.j** avec :

- i=[1..15] = the logical position of the block on the base
- j=[0..7] = logical position of the input in the block

Conversion :

Current inputs	±21,1mA ➔ ±32767 points.
Voltage inputs	±10,25V ➔ ±32767 points.

AI210 board: 16 analog inputs

It is made up of 1 board: 16 analog inputs are available.
16 green LEDs on the front panel. A LED by input ; the LED is ON if the input is monitored.

Notation : **%IWi.j** avec :

- I=[1..15] = the logical position of the block on the base

- j=[0..15] = logical position of the input in the block

Conversion :

Current inputs	±21,1mA ➔ ±32767 points.
Voltage inputs	±10,25V ➔ ±32767 points.

AO121 board: 8 analog outputs

It is made up of 1 board: 8 analog outputs are available.
8 orange LEDs on the front panel. A LED by input ; the LED is ON if the input is monitored.

Notation: **%QWi.j** avec :

- I=[1..15] = the logical position of the block on the base
- j=[0..7] = logical position of the output in the block

Conversion:

Current outputs	0 point ➔ 4 mA 32767 points ➔ 20 mA
Voltage outputs	±32767 points ➔ ±10V

AIO320 board: 8 analog inputs + 4 analog outputs

It is made up of 2 boards:

- Inputs : 8 analog inputs
- Outputs : 4 analog outputs

LEDs on the front panel :

- A green LED by input. ON if the channel is monitored.
- A orange LED by output. ON if the channel is monitored.

Notation :

- I=[1..15] = the logical position of the block on the base

Sub-board 0 : Inputs : **%IWi.0.j**

- j=[0..7] = logical position of the input in the block

Sub-board 1 : Outputs : **%QWi.1.j**

- j=[0..3] = logical position of the output in the block

Conversion:

Current inputs	±20mA ➔ ±32767 points.
Voltage inputs	±10V ➔ ±32767 points.

Current outputs	0 point ➔ 4 mA 32767 points ➔ 20 mA
Voltage outputs	±32767 points ➔ ±10V

PT100	-50° ➔ -500 points
Voltage inputs	+350°C ➔ +3500 points

CPU and I/O Boards Status

The ***IOStatus(BoardOrder)*** function is used to read the status of IO boards :

Function	**IOStatus**
Action	Reads the status of I/O boards
Parameters	BoardOrder [0..15] : 0=CPU, 1=1st I/O board ... , 15=15th I/O board
Returned Value	(INT) : value read [0..FFFFh] Status = 0: board parameters set in the workbench but board inaccessible on the I/O bus Status = -1: Wrong BoardOrder or board parameters not set in the workbench
Example	*Status1 := IOStatus(1);* (* reads the status of the first I/O board *)

Meaning of CPU board status bits

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Signification	RESERVED										PRM	Wdg output	init	IO fault	Wdg LED	Cycle

BIT	Comment
0	set to 1 at end of each cycle
1	set to 1 if WDG LED is on
2	set to 0 if I/O fault
3	set to 1 if initializations completed (set to 1 at end of each cycle)
4	Set to 0 if Watchdog activated
5	set to 1 if PRM detected
6-15	Reserved

Meaning of input/output board status bits

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Signification	Position				Fault	Wdg	Alim	VCC	Identification of the block							

- **Position** (Bits 12 à 15) of the block on the base = [1 …15].
- **Fault** (Bit 11) see the table below.
- **Wdg** (Bit 10) : state of the WDG of the block
  - 1 = WDG inactive = LED Off,
  - 0 = WDG actived = LED ON.
- **Alim** (Bit 9) see the table below.
- **Vcc** (Bit 8) : state of yhe internal power supply of the block.
  - 1 = OK
  - 0 = defect
- **Identification Code of the block (Bits 0 toà 7) :**

Block	**Status Bit 11**	**Status Bit 9**	**Identification block**
DI310	1	Al Ext	03h ou 43h (to mask the bit 6)
DI312	NS	Al Ext(*)	14h
DI410	1	Al Ext	06h
DI130	Fault	Al Ext	59h
DIO130	Fault	Al Ext	58h
DO310	Surcharge	Al Ext	05h ou 45h (to mask the bit 6)

DIO210	Monostable	Vrel	16h
AI110	0	0	80h
AI210	0	0	81h
AO121	0	0	88h
AIO320	Monostable	0	83h

Avec :

- NS : irrelevant
- **AI Ext**= 1 if the external voltage at the terminal blocks is in the Valim ±20% range.
- **(*) AI ext DI312** : external power supply = 24V ± 10%
- **Surcharge** =0 if surcharge on a digital output
- **Monostable** = 1 if OK
  - CAUTION : The polarity is inverse on the DIO130 block.
- **VRel** =1 if the relay coils are correctly energized.
- **Fault**: set to 0 in the event of overload on a digital output channel

This page is intentionally left blank

Chapter

4

CPU functions

Overview

This chapter describes the specific functions of the CPU.

- IP management
- CPU Time
- Data storage
- Web pages

IP management

The CPU6xx can manage 1 ou 2 Ethernet channels. The setup can be operated
With the Registry Host (cf Chap3)
Or with the function **IPCONFIG** in the program executed only one time in the initialization:

Function	**IPCONFIG**
Action	Configure a Ethenet channel
Paramètres	• (STRING) Name of the port : 'eth0' ou 'eth1' • (STRING) IP address of the port • (STRING) Sub-network mask. • (STRING) Gateway address.
Valeur retournée	(INT) : status 1 : commande executed Another value : error.
Exemple	*ConfigEth0:=IPCONFIG('eth0','192.168.1.200','255.255.255.0','192.168.1.1');*

CPU Time

The CPU6xx has a Real Time Clock (RTC).
The RTC provides the date, the clock and the day in the week.
This information can be read with the STRATON functions : DTCurDate, DTCurTime, DTCurDateTime, DTDay, DTMonth, DTYear, DTSec, DTMin, DTHour, DTMs.

The RTC update can be done with:

- The NTP protocol automatically :
  The NTP configuration can be done with:
  - The RegistryHost (cf Chap3),
  - Or 2 functions « NTPSTOP » et « NTPSTART » (see below)
- The DNP3 protocol (cf Chap8), automatically
- The functions « DATE_WRITE » et « TIME_WRITE » in the program (see below)

Function	**NTPSTOP**
Action	Stop the NTP client
Parameters	-
Value	(INT) : status = 1 : commande executed = Another value : error.
Example	*Stop_ntp:= NTPSTOP () ;*

Function	**NTPSTART**
Action	Start the NTP client
Parameters	• (STRING) NTP Server IP adress. • (UDINT) synchronisation interval with the server (in seconde). • (UINT) Time zone code (cf Annex 1»).
Value	(INT) : status = 1 : commande executed = Another value : error
Example	*Start_ntp:= NTPSTART ('192.168.239.250',60, 105) ;*

Function	**DATE_WRITE**
Action	Date update
Parameters	(STRING) Date format = DD/MM/YYYY
Value	(BOOL) : status = TRUE : update OK. = FALSE : update KO.
Example	*Status_date:= DATE_WRITE('20/10/2014') ;*

Function	**TIME_WRITE**
Action	Time update
Parameters	(STRING) Time format = hh:mm:ss or hh:mm:ss.ms
Value	(BOOL) : status = TRUE : Update time OK. = FALSE : Update time KO.
Example	*Status_date:= TIME_WRITE('12 :34 :56.789') ;*

Data Storage

The CPU6xx has a flash memory. The available size for the user is 1Mo. The user can store and read data in binary files. The user must use the Straton File functions: F_ROPEN, F_WOPEN, F_AOPEN, F_CLOSE, F_EOF …

The use of file functions must be very controled in order to not saturate the CPU embedded system file.

The use of these functions should be cautious and moderate so as not to penalize the cycle time

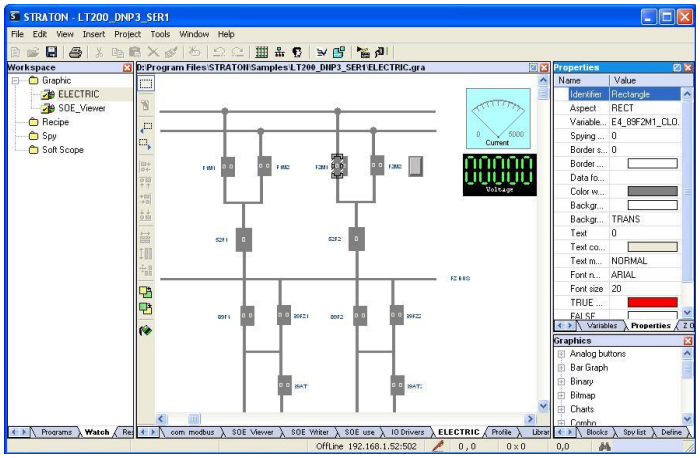
Reading customer files : ftp connexion

The CPU6xx has a FTP server. With a client software installed on your PC, you'll be able to connect you to this FTP server, and to download all files created with Straton functions.

FTP connection parameters must be from type: « anonymous »

Web pages

Web pages can be stored in the Flash memory.

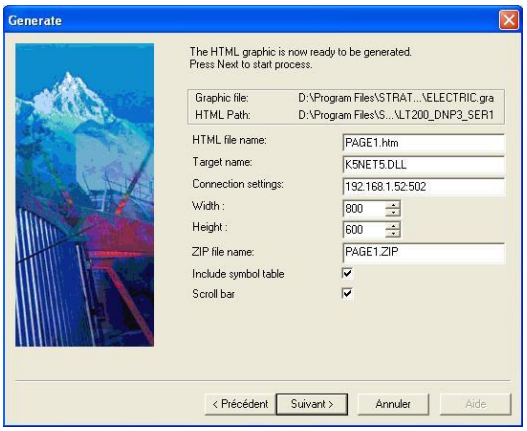


In your Straton project, to create a new graphic document, use the Insert New Graphic popup menu command in the workspace.

Straton workbench contains many graphic objects and features about those graphic pages: see the Straton Topics Help for more details.

In order to generate an embedded HTML web page:

- Select your graphic page in Straton workbench.
- click on menu "Tools/Generate HTML graphic"
- click on buton "next" and fill out the different fields :
  - HTML file name :
  - Target name : DLL used by Straton ActiveX : « K5NET5.DLL »
  - Connection settings : IP adress of the LT followed by « :502 » (TCP port)
  - Width : of the web page
  - Height : of the web page
  - ZIP file name: the name must be written in uppercase

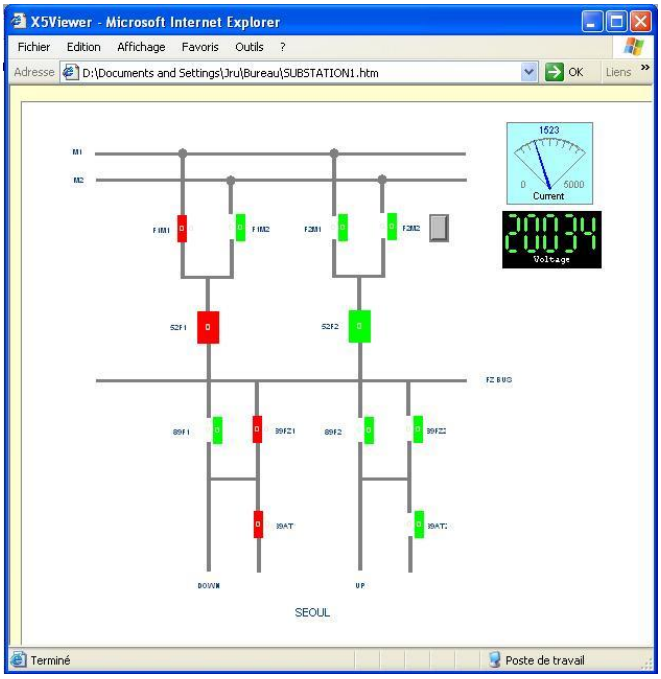


- then click on buton "End" : this action will automatically generate
  - generate the ZIP file and download it in the CPU.
  - Generate the local html page.

In order to watch the web page from your internet browser, you need the following programs and files installed on your PC :

- Internet Explorer or any other browser that is able to manage activeX
- Straton ActiveX (installed automatically with Straton workbench)
- The html file created in the previous step : for example « Page1.htm »

Then you just have to double click on the HTML file: your internet browser will display the HTML page and establish a connection to the LT, in order to refresh all the variables of the HTML page.



Chapitre

5

Serial and Ethernet communication

Transmission / reception of bytes

Presentation

The LT can manage 4 serial (RS232/RS485) and 2 Ethernet channels. In order to emit and receive bytes and strings on the serial and Ethernet port, the LT uses functions described in the STRATON workbench. The functions install and manage the FIFO.

After initialization of the communication port, the user can read and write bytes or strings in the FIFO. The LT Straton kernel takes care emitting bytes from FIFO to the line and receiving bytes from the line to the FIFO.

Serial communication : function "SERIO"

The SERIO function allows :

- To configure and open the selected port

  The format of the input parameter « **CONF** » is a string :

  'tts/port number, speed, parity, databits,stop bits, physical port'

  - tts/n — n=0 : com0, n=1 : com1, n=2 : com2, n=3 : com3
  - speed — 2400, 4800, 9600, 19200, 38400] bits/s.
  - Parity — N=none, E=Even; O=Odd
  - Data bits — 7 or 8
  - Stop bits — 1 or 2
  - Physical port — 0=RS232 , 1=RS485

  example : ***' tts/1:9600,N,8,1,0'*** :

- To emit string
- To check that strings has arrived
- To receive strings

TCP/UDP communication

The functions listed below are used to manage TCP and UDP sockets :

- tcpListen : to create a server socket
- tcpAccept : to accept a client connexion
- tcpConnect : to create a client socket and connect it to the server
- tcpIsConnected : to check a client connected socket.
- tcpClose : to close a socket
- tcpSend : to send strings.
- tcpReceive : to receive strings.
- tcpIsValid : to check if a socket is valid.

The use of these functions requires mastery of the communication principles of TCP / IP networks: misuse of these functions can degrade the operation of the LT.

Chapter

6

Modbus protocol

Overview

This chapter describes the modbus RTU and TCP implementation on the LT.
We detail in this chapter the following steps:

- Modbus protocol
- Modbus Slave protocol: RTU and TCP
- Modbus Master protocol: RTU and TCP

Modbus protocol

Modbus is a communication protocol that allow the exchange of data between several devices. It's a master / slave protocol. The hardware link on the CPU can be either a serial link (RS232, RS485), than an Ethernet link (10/100Mb).
This protocol is described in several downloadables documents : http://www.modbus.org/

The CPU6xx can handle simultaneously the following features:

- on each of its 4 serial links : master or slave modbus RTU
- on the Ethernet links : master and/or slave modbus/TCP

Data are bit and word (16 bits) type.

Functions modbus codes managed by the LT are:

- 1 : read coils : reading bits
- 2 : read bit inputs : reading input bits
- 3 : read holding registers : reading words
- 4 : read input registers : reading input words
- 5 : write 1 coil : writing one bit
- 6 : write 1 register : writing one word
- 15 : write N coils : writing N bits
- 16 : write N registers : writing N words

Modbus slave protocol

Modbus slave service must first have been initialized.

Insert a configuration :

Click on the button « Insert Configuration », then choose in the protocol list the « modbus Slave protocol ».

Insert a Slave / DataBlock :

Click on the button « Insert Master/Port », then choose the type of network :
If you have to add a modbus serial slave, you have to program as in the example below :

```
MySlave1 (TRUE,    'tts/1:9600,N,8,1,0',    1);
Q_myslave1 := MySlave1.Q;
```

With : « MySlave1 » is an instance of the function block « **MBSlaveRTU** ».
The parameters are:

- **IN** : (BOOL) Enabling command: the port is open when this input is TRUE.
- **CONF** : (STRING) Settings string chain :

'tts/port number, speed, parity, databits,stop bits, physical port'

tts/port	port=0 : com0, 1 : com1, 2 : com2, 3 : com3
speed	[2400, 4800, 9600, 19200, 38400] bits/s.
Parity	N=none, E=Even; O=Odd
Data bits	7 or 8
Stop bits	1 or 2
Physical port	0=RS232 , 1=RS485

example : ***' tts/1:9600,N,8,1,0'*** :

- **SLV** : (DINT) modbus slave number.

Insert Modbus data blocks

Double click on the Local Server item to setup the modbus slave number that will identify the runtime application. When the local server is selected, use the Edit / New data block menu command to insert modbus data blocks. The following kinds of block are available:

- input bits: bits read by external masters (function 2).
- coil bits: bits forced by by external masters (function 5 or 15).
- input registers: words read by external masters (function 4).
- holding registers: words forced by external masters (function 6 or 16).

Each data block is identified by a modbus base address and a number of items (bits or words).

Insert a variable in the data block

When a server data block is selected, use the *Edit / New variable* command to map a variable to an item of the data block. Each variable is identified by a valid symbol of a variable in the open project and an offset in the data block according to modbus addressing.

For exchanging boolean variables through modbus words, a hexadecimal mask is available in order to define to which bit of a word a variable is attached. For example, enter the mask "0001" to map a boolean variable to the less significant bit of a word.

For exchanging 32 bit variables (DINT, REAL...), you can select to map the variable on two consecutive words.

At any time you can sort the variables of each data block according to their offset using the *Edit / Sort symbols* menu command.

Modbus master protocol

The client side opens a new port (RS or ETHERNET) for each configured Modbus port. The number of client ports is not limited.

A modbus RTU master works with one COM port only. If there are several slaves (network 485 or 422), it is the slave number (in the structure Modbus Request) that differentiates.

A modbus/TCP master can have only a single TCP slave (single IP address provided to initialize the master).

All the modbus master settings are done through the open fieldbus configuration:

Insert a configuration :

Click on the button « Insert Configuration », then choose in the protocol list the « modbus Master protocol ».

Insert a Master/Port :

Click on the button « *Insert Master/Port* », then choose the type of network:

- for ETHERNET, you must enter the IP address of the slave

➢ for the « Ethernet » port : you must enter :
  - the IP address of the slave.
  - The number of the modbus/TCP port : 502
  - The name of the protocol : *TCP – Open MODBUS* (NOTA : the UDP – Open MODBUS and UDP – Modbus RTU are not supported by the LT).

- for a serial network, you must enter the configuration string chain as following,

'tts/port number, speed, parity, databits,stop bits, physical port'

  - tts/port — port= 0:com0; 1:com1; 2:com2; 3:com3
  - speed — [2400, 4800, 9600, 19200, 38400] bits/s.
  - Parity — N=none, E=Even; O=Odd
  - Data bits — 7 or 8
  - Stop bits — 1 or 2
  - Physical port — 0=RS232 , 1=RS485

example : ***' tts/1:9600,N,8,1,0'*** :

Insert a Slave / Data Block :

Click on the button « Insert Slave/Data Block », then you have to select:

- the « slave/unit » : modbus slave number
- the modbus function : « Read Coil Bits », « Read Input Bits », « Read Holding Registers », …
- the Modbus request call:
  - ➢ Periodic: the request is constant emission period,
  - ➢ On call: the request is issued by event (booleans triggers)
  - ➢ On change: the request is issued only if a variable change

Insert a variable in the data block

When a server data block is selected, use the *Edit / New variable* command to map a variable to an item of the data block. Each variable is identified by a valid symbol of a variable in the open project and an offset in the data block according to modbus addressing.

For exchanging boolean variables through modbus words, a hexadecimal mask is available in order to define to which bit of a word a variable is attached. For example, enter the mask "0001" to map a boolean variable to the less significant bit of a word. For exchanging 32 bit variables (DINT, REAL...), you can select to map the variable on two consecutive words.

At any time you can sort the variables of each data block according to their offset using the *Edit / Sort symbols* menu command.

This page is intentionally left blank

Chapter

7

T5 Binding protocol

Binding protocol

The T5 binding protocol is an event driven protocol on TCP-IP for exchanging real time between several LT. As the protocol is purely event-based, it ensures high performances and very low network trafic at runtime.

This section explains how to use the T5 event-based protocol for binding variables of runtime systems in order to build a distributed application.

Mechanism

The T5 protocol is based on the "publication-subscription" model. Each LT can produce (publish) variables on the network, and consume (underwriting) variables produced by other LTs. Each generated variable is identified by a number (identifier). This identifier is used to associate the source and destination variables in the respective LTs. The same variable produced by an LT can be consumed by several LTs.

Exchanges

The value of the variable is only sent on the network when it changes. For each variable produced, set positive and negative hysteresis to adjust the network load according to the needs of your application. Each new value issued on the network is dated. For each variable consumed, the dating of the last received value, as well as a quality flag are available. An overall status of the connection for each connected producer is also available.

Limitations

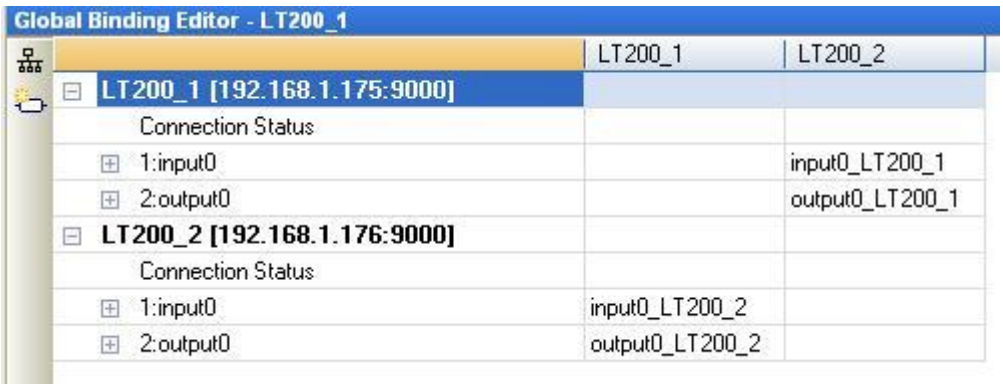
The maximum number of variables produced by an LT is limited to 256 variables.
Only Boolean, integer and TIME variables can be exchanged. STRING variables and function block instances are not supported by the protocol.

Using the global binding editor

The workbench offers a global tool for defining the data exchange among several projects. To build a distributed application, you need to:

- Create all source and destination projects,
- from the main window, run the Tools / Global Binding Editor menu command.

Definition of the configurations



Click on the button "Add/Remove projects" of the Global Binding Editor to initiate a new distributed configuration and define the nodes linked in the configuration.

Click on the button "Insert variable" to add variable of the existing project to bind.
Each node is referred to as a "project" in the global binding editor, and is identified by a name, an IP address and an ETHERNET port number (the port number used for publishing variables is 9000).

Redundant Ethernet networks

In the case of redundant Ethernet networks, specify two IP addresses instead of one for each node involved. For this, enter two IP addresses separated by a division sign "/". Information "Connection Status" remains available in the case of redundant links. In this case, the value of the variable connected to the state reflects the state of the two connections. The state is stored as an integer. The state of the first connection is in bits 0 to 3, and the state of the second in bits 4 to 7. For each connection, the value "0" means OK

Chapter

8

DNP3.0 slave protocol

Overview

This chapter describes the serial and Ethernet DNP3.0 slave implementation over serial and Ethernet networks on the LT.

We detail in this chapter:

- the communication principle
- the DNP3 slave configuration
- the DNP3 variables wiring

You will find too in appendix the "DNP3 Profile document and implementation table".

Communication principle

The Distributed Network Protocol (DNP3.0) is an industry communication standard between SCADA systems and RTUs or IEDs (Intelligent Electronic Devices). DNP protocol is a "Master" to "Slave" protocol. Masters are SCADA systems, slaves are RTUs or IEDs systems. Each DNP node on the DNP3 network has an address : this address allows to the masters to request selectively datas from each slaves.

DNP is used for substation automation, Gas control or Water control.

DNP is managed by the DNP User's Group : all informations about this protocol are available on : http://www.dnp.org/

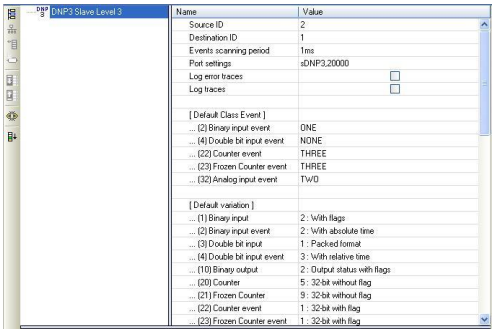
LT is a DNP3.0 slave level 2 on a serial port or Ethernet port depending on the soft settings, and the wiring:

- DNP3.0 serial slave : one or two of its serial ports can be set ; the case of two serial ports configured is for implementing a line redundancy.
- DNP3.0 Ethernet slave : two slaves connections can be set : the TCP ports have to be different: 20000 & 20001 for example.

DNP3 slave configuration

The workbench includes an integrated fieldbus configuration tool: click on the button  and then insert a new "DNP3 Slave Level3" configuration.

The configuration tool enables you to describe DNP3 slave properties.



Main properties

Name	Value
Source ID	Source identifier of the LT (slave)
Destination ID	Destination identifier for the master (SCADA)
Event Scanning period	Time refreshing (ms)
Port settings	Settings string of the COM port : examples : Ethernet : « sDNP3,20000», with 20000 as TCP port number Serial : « tts/0:9600,n,8,1,0» for a single serial port (1), « tts/0:9600,n,8,1,0;tts/1 » for a double serial port (2)
Log errors	If checked, error messages are sent to the Workbench.
Log warnings	If checked, warning messages are sent to the Workbench.

(1) detail of the serial com port settings : « tts/0:9600,n,8,1,0»

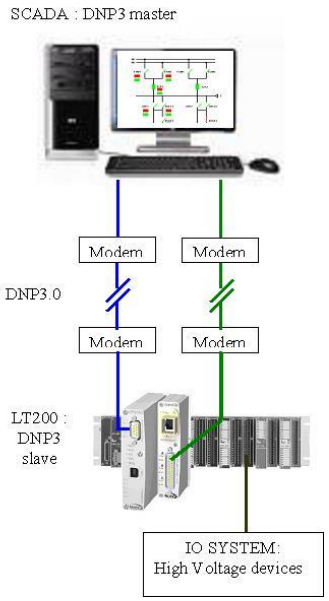
- tts/0 : number of the serial port used : 0 for com0, 1 for com1, 2 for com2, 3 for com3
- 9600 : baudrate : available speeds : 2400, 4800, 9600, 19200, 38400.
- n : parity : N (none) or E (even) or O (odd)
- 8 : number of data bits (7 or 8)
- 1 : number of stop bits (1 or 2)
- 0 : mode : choose of the type of port:
    - 0 for RS232
    - 1 for RS485

(2) detail of the double serial port settings : « tts/0:9600,n,8,1,0;tts/1»

- tts/1 : number of the second serial port used : 0 for com0, 1 for com1, 2 for com2, 3 for com3

The second port must be different from the one. The other settings for this second port (baudrate, parity…) are the same than for the first serial port.

This feature allows you to use a redundant line : the master will switch in when the first line fails.



Advanced Parameters:

They are arranged in sections:

- Default class event
- Default variation
- Event Mode
- Unsollicited messages
- Device attributes
- Misc

The symbol (*) in the following arrays indicate the default value.

Default Class Event

Specifies the default class that will be used for unsolicited responses. If class is not allowed by master unsolicited responses are not send for the class.

Name	Value
(2) Binary input event	NONE (no effect), ONE (*), TWO, THREE
(4) Double bit input event	NONE (no effect) (*), ONE, TWO, THREE
(22) Counter event	NONE (no effect), ONE, TWO, THREE(*)
(23) Frozen Counter event	NONE (no effect), ONE, TWO, THREE(*)
(32) Analog input event	NONE (no effect), ONE, TWO(*), THREE

Default variation

Specifies the variation that will be used for unsolicited responses and in response to a read requesting variation 0.

Name	Value
(1) Binary input	1(*), 2
(2) Binary input event	1, 2, 3(*)
(3) Double bit input	1(*), 2
(4) Double bit input event	1, 2, 3(*)
(10,12) Binary output	1, 2(*)
(20) Counter	1, 2, 5(*), 6
(21) Frozen Counter	1, 2, 5, 6, 9(*), 10
(22) Counter event	1(*), 2, 5, 6
(23) Frozen Counter event	1(*), 2, 5, 6
(30) Analog input	1, 2, 3(*), 4, 5
(32) Analog input event	1(*), 2, 3, 4, 5, 7
(40) Analog Output status	1(*), 2, 3

Event mode

Specifies the event mode that will be used for unsolicited responses:

- ALL : Sequence of Events, return all events
- MOST RECENT : only most recent event

Name	Value
(2) Binary input event	ALL(*), MOST RECENT
(4) Double bit input event	ALL(*), MOST RECENT
(22) Counter event	ALL, MOST RECENT(*)
(23) Frozen Counter event	ALL(*), MOST RECENT
(32) Analog input event	ALL, MOST RECENT(*)

Unsollicited messages

Name	Value	Description
Unsolicited allowed	Checkbox : Yes = checked NO = not checked	Determines whether unsolicited responses are allowed. If "Unsolicited Allowed" is set to FALSE no unsolicited responses will be generated and requests to enable or disable unsolicited responses will fail.
Unsolicited event mask	NONE(*), ONE, TWO, THREE, ALL	Specify the initial/new state of the unsolicited event mask. This mask is used to determine which event class(es) will generate unsolicited responses. According to the DNP specification, unsolicited responses should be disabled until an 'Enable Unsolicited Response' request is received from the master. Hence this value should generally be NONE, but some masters do not generate the 'Enable Unsolicited Response' message, in which case they must be enabled here.

Unsolicited retry number	3	Specify the maximum number of unsolicited retries before changing to the 'offline' retry period. This parameter allows you to specify up to 65535 retries.
Unsolicited retry delay	5s	Specifies the time to delay after an unsolicited confirm timeout before retrying the unsolicited response.
Unsolicited offline retry delay	30s	Specifies the time to delay after an unsolicited timeout before retrying the unsolicited response after Unsolicited retry number have been attempted.
Class 1 : Unsolicited events number	5	If unsolicited responses are enabled, Unsolicited events number specifies the maximum number of events in the corresponding class to be allowed before an unsolicited response will be generated.
Class 2 : Unsolicited events number	5	
Class 3 : Unsolicited events number	5	
Class 1 : Unsolicited events delay	5s	If unsolicited responses are enabled, Unsolicited events delay specifies the maximum amount of time after an event in the corresponding class is received before an unsolicited response will be generated.
Class 2 : Unsolicited events delay	5s	
Class 3 : Unsolicited events delay	5s	

Device attributes (object 0)

All following device attibutes can be modified. Double click on the "DNP3 Slave Level 3" item to open the Device attributes window. Then you can enter the appropriate value for each attribute.

(Variation) Name
(211) Identifier of support for user-specific attributes
(212) Number of master-defined data set prototypes
(213) Number of outstation-defined data set prototypes
(214) Number of master-defined data sets
(215) Number of outstation-defined data sets
(216) Max number of binary outputs per request
(217) Local timing accuracy
(218) Duration of timing accuraccy
(219) Support for analog output events
(220) Max analog output index
(221) Number of analog outputs
(222) Support for binary output events
(223) Max binary output index
(224) Number of binary outputs
(225) Support for frozen counter events
(226) Support for frozen counters
(227) Support for counter events
(228) Max counter index
(229) Number of counter points
(230) Support for frozen analog inputs
(231) Support for analog input events
(232) Maximum analog input index
(233) Number of analog input points
(234) Support for double-bit binary input events
(235) Maximum double-bit binary input index
(236) Number of double-bit binary input points
(237) Support for binary input events
(238) Max binary input index

(239) Number of binary input points
(240) Max transmit fragment size
(241) Max receive fragment size
(242) Software version
(243) Hardware version
(245) User-assigned location name
(246) User assigned ID
(247) User assigned device name
(248) Serial number
(249) DNP subset and conformance
(250) Product name and model
(252) Manufacturer's name

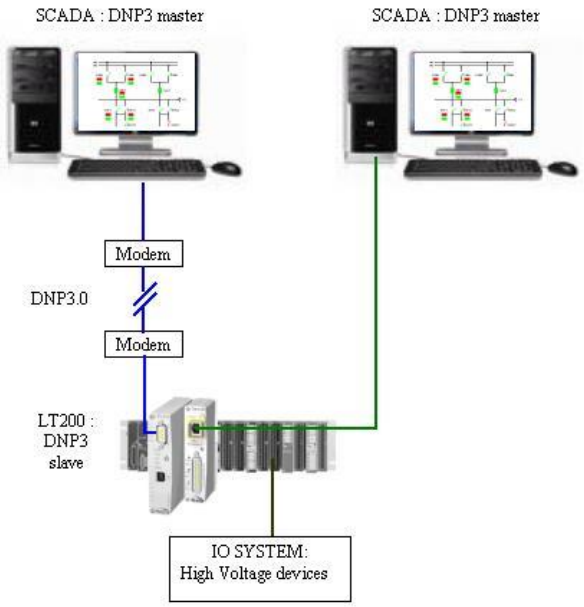
Misc

Name	Value	Description
Self address enable (code (Oxfffc))		Specify whether or not to enable self address functionality on this slave device. Slave will respond to address 0xfffc as though it received a request for its configured address. It will respond with its own address so the master can automatically discover the slave address.
Clock valid period	30s	Specifies how long the local clock will remain valid after receiving a time synchronization.
Application confirm timeout	10s	Application confirm timeout specifies how long the slave DNP device will wait for an application layer confirmation from the master. This in combination with Unsolicited retry delay or Unsolicited offline retry delay will determine how frequently an unsolicited response will be resent.
Output select timeout	5s	SelectTimeout specifies the maximum amount of time that a select will remain valid before the corresponding operate is received. If an operate request is received after this period has elapsed since the previous select the select will not be valid and the operate request will fail.
Integrity poll response groups	1,3,10,20 ,21,30,40	Object groups included in response to read static data request.
Second port settings		Settings string of the second COM port for a multiport management: examples : Ethernet : « sDNP3,20000», with 20000 as TCP port number Serial : « tts/0:9600,n,8,1,0» for a single serial port (1), « tts/0:9600,n,8,1,0;tts/1 » for a double serial port (2)

The "second port settings" feature allows you to add a second DNP3 master to the LT. It will then respond to each DNP3 masters connected through Ethernet or a serial link :

Several network configurations can be done: DNP3 masters can be :

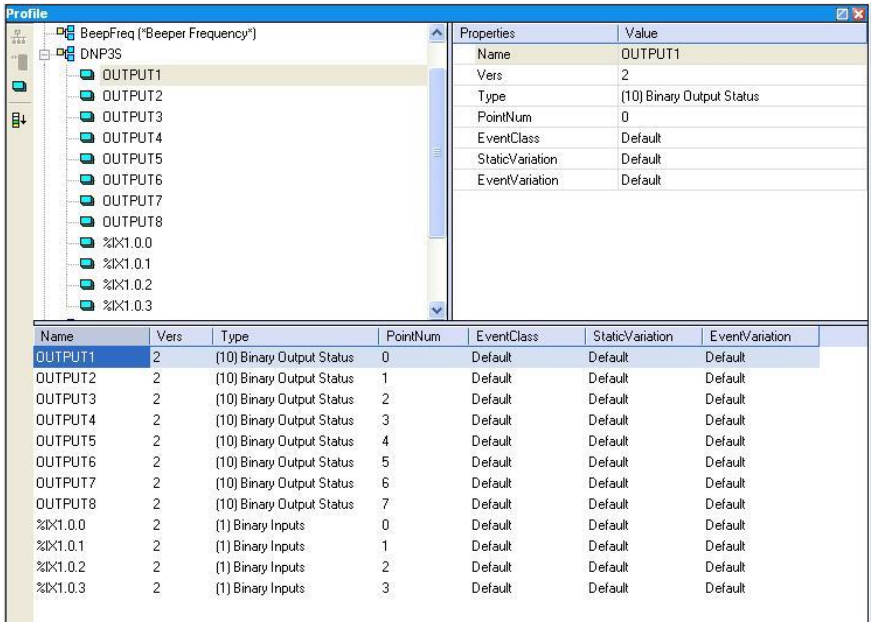
- Ethernet masters
- serial masters
- one Ethernet master and the other a serial master.



DNP3 variables wiring

The workbench includes too an integrated variable profile tool : click on the following button to open it and then click on "DNP3S".
To map a variable to the DNP3 slave protocol, simply drag and drop it in the profile grid from dictionary.
Another way to map a variable to the DNP3 slave protocol, is to edit its properties using the Properties command of the contextual menu in the variable editor, and then select the DNP3S profile.
Then you must specify the following properties for the variable:
For each variables, you can change its properties :

Properties	Description
Type	(1)Binary Input, (10)Binary Output Status, ...
PointNum	DNP3 point number
EventClass	NONE (no effect), ONE, TWO, THREE
StaticVariation	1 to 9
EventVariation	1 to 9



Chapter

9

Monitoring and diagnostic

LED : Power Supply, CPU and I/O boards

Power supply LED

If the power supply is present and correct, the corresponding green LED lights up without flashing.

CPU leds

LED name	Color	Signal	Meaning
RUN	Green	Flashing (period : 2s)	if the user application is running
		Flashing (period : 1/2s	if the user application is in STOP
		Light on	PRM mode or step to step
		Flashing alternately with the PRM LED	CPU starting
FAIL	Orange	Light on	if the Straton kernel is corrupted.
		OFF	if operation is correct
I/O	Green	Light on	if operation is correct
		flashing	if an I/O board is not correctly inserted or if at least one I/O board status is incorrect while the program is running
		Flashing alternately with the Run led	CPU starting phase
PRM	Green	lights up without flashing	if the equipment is in PRM mode when the LT is booted.
WDG	Red	Light on	while WDG is active
		off	if Straton kernel is in RUN mode ; the hardware watchdog is refreshed by processor.
CP	Green	Ligth on	if the I/O coprocessor is runing (program downloaded).
F1	Green		Managed by the software user
F2	Green		Managed by the software user

Communication LED : serial and Ethernet

LED name	Color	Meaning
Lk	orange	Ethernet link : Light ON if the Ethernet link is wired up to another Ethernet device
Rx	green	Light ON if byte reception on the corresponding serial port
Tx	green	Light ON if byte sending on the corresponding serial port
SM	green	Not used
Te	green	Not used

Input/Output boards leds

Input/output LEDs are automatically refreshed by the kernel. Their management is as following :

Board type	Led on front I/O board face	LED State	Meaning
All	FLT : red color	ON	3 possible cases : general WDG internal board power supply in fault no monitoring from CPU
DI310 DI410 DI130 DIO210	1 green LED per input	ON	If input is in TRUE state
DIO130	1 green LED for the trip position(I2,I4,I6,I8) 1 red LED for the close position (I1,I3,I5,I7)	ON	If input is in TRUE state
DO310 DIO130 DIO210	1 orange LED per digital output channel	ON	If output is in TRUE state
AI110 AI210 AIO320	1 green LED per analog input	ON	If the CPU manages this channel
AO121 AIO320	1 green LED per analog output	ON	If the CPU manages this channel

Console link troubleshooting: PRM mode

The configuration mode (PRM) is used in order to restore the console link between the PC and the LT Straton, in the case of loss of it after a problem with the Straton application. In the PRM mode only the Straton kernel is executed but not the client application.

To switch to PRM mode:

- switch off the LT,
- Shunt the pins 7 and 8 of COM0
- switch on the LT,
- some seconds after startup, led **PRM** lights up, then the shunt can be removed,

The LT Straton start in safe mode and wait for a connection with the workbench on the Ethernet or USB console link.

Annex 1 : Time zone codes for the NTPSTART function

Code	Fuseau horaire GMT	Zone géographique
002	GMT-10:00	Hawaii
003	GMT-09:00	Alaska
004	GMT-08:00	Pacific Time (US and Canada); Tijuana
010	GMT-07:00	Mountain Time (US and Canada)
013	GMT-07:00	Chihuahua, La Paz, Mazatlan
015	GMT-07:00	Arizona
020	GMT-06:00	Central Time (US and Canada
025	GMT-06:00	Saskatchewan
030	GMT-06:00	Guadalajara, Mexico City, Monterrey
033	GMT-06:00	Central America
035	GMT-05:00	Eastern Time (US and Canada)
040	GMT-05:00	Indiana (East)
045	GMT-05:00	Bogota, Lima, Quito
050	GMT-04:00	Atlantic Time (Canada)
055	GMT-04:00	Caracas, La Paz
056	GMT-04:00	Santiago
060	GMT-03:30	Newfoundland and Labrador
065	GMT-03:00	Brasilia
070	GMT-03:00	Buenos Aires, Georgetown
073	GMT-03:00	Greenland
080	GMT-01:00	Azores
083	GMT-01:00	Cape Verde Islands
085	GMT	Greenwich Mean Time: Dublin, Edinburgh, Lisbon, London
090	GMT	Casablanca, Monrovia
095	GMT+01:00	Belgrade, Bratislava, Budapest, Ljubljana, Prague
100	GMT+01:00	Sarajevo, Skopje, Warsaw, Zagreb
105	GMT+01:00	Brussels, Copenhagen, Madrid, Paris
110	GMT+01:00	Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna
115	GMT+02:00	Bucharest
120	GMT+02:00	Cairo
125	GMT+02:00	Helsinki, Kiev, Riga, Sofia, Tallinn, Vilnius

Code	Fuseau horaire	Zone géographique
130	GMT+02:00	Athens, Istanbul, Minsk
135	GMT+02:00	Jerusalem
140	GMT+02:00	Harare, Pretoria
145	GMT+03:00	Moscow, St. Petersburg, Volgograd
150	GMT+03:00	Kuwait, Riyadh
155	GMT+03:00	Nairobi
158	GMT+03:00	Baghdad
160	GMT+03:30	Tehran
165	GMT+04:00	Abu Dhabi, Muscat
170	GMT+04:00	Baku, Tbilisi, Yerevan
175	GMT+04:30	Kabul
180	GMT+05:00	Ekaterinburg
185	GMT+05:00	Islamabad, Karachi, Tashkent
190	GMT+05:30	Chennai, Kolkata, Mumbai, New Delhi
193	GMT+05:45	Kathmandu
195	GMT+06:00	Astana, Dhaka
200	GMT+06:00	Sri Jayawardenepura
201	GMT+06:00	Almaty, Novosibirsk
203	GMT+06:30	Yangon Rangoon
205	GMT+07:00	Bangkok, Hanoi, Jakarta
207	GMT+07:00	Krasnoyarsk
210	GMT+08:00	Beijing, Chongqing, Hong Kong SAR, Urumqi
215	GMT+08:00	Kuala Lumpur, Singapore
220	GMT+08:00	Taipei
225	GMT+08:00	Perth
227	GMT+08:00	Irkutsk, Ulaanbaatar
227	GMT+08:00	Irkutsk, Ulaanbaatar
230	GMT+09:00	Seoul
235	GMT+09:00	Osaka, Sapporo, Tokyo
240	GMT+09:00	Yakutsk
245	GMT+09:30	Darwin

250	GMT+09:30	Adelaide
255	GMT+10:00	Canberra, Melbourne, Sydney
260	GMT+10:00	Brisbane
265	GMT+10:00	Hobart
270	GMT+10:00	Vladivostok
275	GMT+10:00	Guam, Port Moresby
280	GMT+11:00	Magadan, Solomon Islands, New Caledonia
285	GMT+12:00	Fiji Islands, Kamchatka, Marshall Islands
290	GMT+12:00	Auckland, Wellington
300	GMT+13:00	Nuku'alofa

Appendix 2: DNP3.0 PROFILE DOCUMENT & implementation table

This document describes the device capabilities, the current value of each parameter, or both. If it is used to show the current values, or Straton parameters if configurable ("NA" may be entered for parameters that are Not Applicable).

The conformance tests have been performed using Triangle Microworks Communication Protocol Test Harness Version 3.5.0.0 (DNP3 IED Certification Procedure Subset Level 2).

DEVICE PROFILE DOCUMENT

DNP V3.0 **DEVICE PROFILE DOCUMENT**	
Vendor Name: **Leroy Automation**	
Device Name: **LT200**	
Highest DNP Level Supported: For Requests: **Level 2** For Responses: **Level 2**	Device Function: ☐ Master ☒ **Slave**
Device manufacturer's software version string :	V1.3
Methods to set Configurable Parameters:	Software – Straton workbench
Connections Supported	☒ Serial ☒ IP Networking
Notable objects, functions, and/or qualifiers supported in addition to the Highest DNP Levels Supported (the complete list is described in the attached table): For static (non-change-event) object requests, request qualifier codes 07 and 08 (limited quantity), and 17 and 28 (index) are supported. Static object requests sent with qualifiers 07, or 08, will be responded with qualifiers 00 or 01. **Object 0 (device attributes) is supported: variations 211 to 252** **Object 3 (Double bit binary Input) is supported: variations 1, 2** **Object 4 (Double bit binary event) is supported: variations 1, 2, 3** **Object 23 (Frozen counter event) is supported: variations 1, 2, 5, 6** **16-bit, 32-bit and Floating Point Analog Change Events with Time may be requested.** **Floating Point Analog Output Status and Output Block Objects 40 and 41 are supported.** **Object 110 (Octet String Object) is supported.**	
Serial Connections	
Port name	com0, com1, com2 or com3
Serial Connection Parameters, Baud rate	Configurable (Straton 'Port settings')
IP Networking	
Type of End Point	TCP Listening
IP Address, Subnet Mask, Gateway IP Address	Configurable : Straton 'Registry Host'
TCP Listen Port Number	Configurable : Straton 'Port settings'
Multiple master connections(Outstations Only)	Supports 2 masters Method 2 (based on IP port number) used (Straton 'Port settings' & 'Second Port settings')
Time synchronization support	DNP3 Write Time (obj50, Var1)
Link Layer – Application Layer	

DNP V3.0

DEVICE PROFILE DOCUMENT

Maximum Data Link Frame Size (octets):	Maximum Application Fragment Size (octets):
Transmitted: **292** Received **292**	Transmitted: **2048** Received **2048**
Maximum Data Link Re-tries: ☐ None ☒ **Fixed : 3** ☐ Configurable from 0 to 65535	Maximum Application Layer Re-tries: ☒ **None** ☐ Configurable

Requires Data Link Layer Confirmation:

- ☒ **Never**
- ☐ Always
- ☐ Sometimes
- ☐ Configurable as: Never, Only for multi-frame messages, or Always

Requires Application Layer Confirmation:

- ☐ Never
- ☐ Always
- ☒ **When reporting Event Data (Slave devices only)**
- ☒ **When sending multi-fragment responses (Slave devices only)**
- ☐ Sometimes
- ☐ Configurable as: "Only when reporting event data", or "When reporting event data or multi-fragment messages."

Timeouts while waiting for:

Data Link Confirm:	☐	None	☒	**Fixed at 3**	☐	Variable	☐	
Complete Appl. Fragment:	☒	**None**	☐	Fixed at _____	☐	Variable	☐	Configurable
Application Confirm:	☐	None	☐	Fixed at _____	☐	**Variable**	☒	**Straton (appl.Conf. timeout).**
Complete Appl. Response:	☒	**None**	☐	Fixed at _____	☐	Variable	☐	Configurable

Others: Transmission Delay : **fixed 0**
Select/Operate Arm Timeout : **configurable Straton (output select timeout)**
Need Time Interval : **configurable Straton (clock valid period)**
Application File Timeout : **configurable Straton (file transfer timeout)**
Unsolicited Notification Delay : **configurable Straton (unsolicited events delay)**
Unsolicited Response Retry Delay : **configurable Straton (unsolicited retry delay)**
Unsolicited Offline Interval : **configurable Straton (unsolicited offline retry delay)**
Binary Change Event Scan Period : **configurable Straton (event scanning period)**
Double Bit Change Event Scan Period : **configurable Straton (event scanning period)**
Analog Change Event Scan Period : **configurable Straton (event scanning period)**
Counter Change Event Scan Period : **configurable Straton (event scanning period)**
Frozen Counter Change Event Scan Period : **configurable Straton (event scanning period)**
String Change Event Scan Period : **Never**
Virtual Terminal Event Scan Period : **Never**

DNP V3.0

DEVICE PROFILE DOCUMENT

Sends/Executes Control Operations:

WRITE Binary Outputs	☒ **Never**	☐ Always	☐ Sometimes	☐ Configurable
SELECT/OPERATE	☐ Never	☒ **Always**	☐ Sometimes	☐ Configurable
DIRECT OPERATE	☐ Never	☒ **Always**	☐ Sometimes	☐ Configurable
DIRECT OPERATE – NO ACK	☐ Never	☒ **Always**	☐ Sometimes	☐ Configurable
Count > 1	☒ **Never**	☐ Always	☐ Sometimes	☐ Configurable
Pulse On	☐ Never	☒ **Always**	☐ Sometimes	☐ Configurable
Pulse Off	☐ Never	☒ **Always**	☐ Sometimes	☐ Configurable
Latch On	☐ Never	☒ **Always**	☐ Sometimes	☐ Configurable
Latch Off	☐ Never	☒ **Always**	☐ Sometimes	☐ Configurable
Queue	☒ **Never**	☐ Always	☐ Sometimes	☐ Configurable
Clear Queue	☒ **Never**	☐ Always	☐ Sometimes	☐ Configurable

Attach explanation if 'Sometimes' or 'Configurable' was checked for any operation.

Reports Binary Input Change Events when no specific variation requested:

- ☐ Never
- ☐ Only time-tagged
- ☐ Only non-time-tagged
- ☒ **Configurable Straton (default variation)**

Reports time-tagged Binary Input Change Events when no specific variation requested:

- ☐ Never
- ☐ Binary Input Change With Time
- ☐ Binary Input Change With Relative Time
- ☒ **Configurable Straton (default variation)**

Outstation Unsolicited Response Support

Sends Unsolicited Responses:

- ☐ Never
- ☒ **Configurable Straton (unsolicited allowed)**
- ☐ Only certain objects
- ☐ Sometimes (attach explanation)
- ☒ **ENABLE/DISABLE UNSOLICITED Function codes supported**

Sends Static Data in Unsolicited Responses:

- ☒ **Never**
- ☐ When Device Restarts
- ☐ When Status Flags Change

No other options are permitted.

Default Counter Object/Variation:

- ☐ No Counters Reported
- ☒ **Configurable Straton (default variation)**
- ☐ Default Object
  Default Variation:
- ☐ Point-by-point list attached

Counters Roll Over at:

- ☐ No Counters Reported
- ☐ Configurable (attach explanation)
- ☐ 16 Bits
- ☒ **32 Bits**
- ☐ Other Value: _____
- ☐ Point-by-point list attached

Sends Multi-Fragment Responses:

- ☒ **Yes**
- ☐ No
- ☐ Configurable

DNP subset definition: Implementation table

The following implementation table identifies which object groups and variations, function codes and qualifiers the LT supports in both requests and responses. The *Request* columns identify all requests that may be sent by a Master, or all requests that must be parsed by the LT. The *Response* columns identify all responses that must be parsed by a Master, or all responses that may be sent by the LT.

For static (non-change-event) objects, requests sent with qualifiers 00, 01, 06, 07, or 08, will be responded with qualifiers 00 or 01. Requests sent with qualifiers 17 or 28 will be responded with qualifiers 17 or 28. For change-event objects, qualifiers 17 or 28 are always responded.

DNP OBJECT GROUP & VARIATION			**REQUEST** Master may issue Outstation must parse		**RESPONSE** Master must parse Outstation may issue	
Group Num	**Var Num**	**Description**	**Func Codes (dec)**	**Qual Codes (hex)**	**Func Codes (dec)**	**Qual Codes (hex)**
0	1-253	Device Attribute Specific	1	00,01,06,07,08,17,27,28	129	00,01,17,27,28
			2	00,01		
0	254	Device Attribute – Non Specific All Attributes Request	1	00,01,06, 07,08, 17,27,28	129	00, 01 17,28
0	255	Device Attribute – List of Attribute Variations	1	00,01,06,07,08,17,27,28	129	00,01,17,28
1	0	Binary Input – Any Variation	1, 22	00,01,06		
1	1 (default see note1)	Binary Input – Packed format	1	00,01,06	129	00,01,17,27,28
1	2	Binary Input – With flags	1	00,01,06	129	00,01,17,27,28
2	0	Binary Input Event – Any Variation	1	06,07,08		
2	1	Binary Input Event – Without time	1	06,07,08	129 130	17, 28
2	2	Binary Input Event – With absolute time	1	06,07,08	129 130	17, 28
2	3 (default see note1)	Binary Input Event – With relative time	1	06,07,08	129 130	17, 28
3	0	Double bit Input – Any Variation	1, 22	00,01,06		
3	1 (default see note1)	Double bit Input – Packed format	1	00,01,06	129	00,01,17,27,28
3	2	Double bit Input – With flags	1	00,01,06	129	00,01,17,27,28
4	0	Double bit Input Event – Any Variation	1	06,07,08		
4	1	Double bit Input Event – Without time	1	06,07,08	129 130	17, 28
4	2	Double bit Input Event – With absolute time	1	06,07,08	129 130	17, 28
4	3 (default see note1)	Double bit Input Event – With relative time	1	06,07,08	129 130	17, 28
10	0	Binary Output – Any Variation	1,22	00,01,06,07,08,17,27,28		
10	1	Binary Output	1 (read)	00,01,06,07,08,17,27,28	129	00,01,17,28
			1 (write)	00,01		
10	2 (default see note1)	Binary Output – Output status with flags	1	00,01,06	129	00,01
12	1	Control Relay Output Block	3,4,5,6	17,28	129	Echo of request
12	2 (default see note1)	Pattern Control Block	3,4,5,6	00,01	129	Echo of request
20	0	Counter – Any Variation	1,22	00,01,06, 07,08,17,27,28		
			7,8,9,10	00,01,06,07,08		
20	1	Counter – 32 bit with flag	1	00,01,06,07,08,17,27,28	129	00,01,17,28
20	2	Counter – 16 bit with flag	1	00,01,06,07	129	00,01,17,28

DNP OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation must parse		RESPONSE Master must parse Outstation may issue	
Group Num	**Var Num**	**Description**	**Func Codes (dec)**	**Qual Codes (hex)**	**Func Codes (dec)**	**Qual Codes (hex)**
20	5 (default see note1)	Counter – 32 bit without flag	1	00,01,06	129	00,01,17,28
20	6	Counter – 16 bit without flag	1	00,01,06	129	00,01,17,28
21	0	Frozen Counter – Any Variation	1,22	00,01,06		
21	1	Frozen Counter – 32 bit with flag	1	00,01,06	129	00,01,17,28
21	2	Frozen Counter – 16 bit with flag	1	00,01,06	129	00,01,17,28
21	5	Frozen Counter – 32 bit with flag and time	1	00,01,06	129	00,01,17,28
21	6	Frozen Counter – 16 bit with flag and time	1	00,01,06	129	00,01,17,28
21	9 (default see note1)	Frozen Counter – 32 bit without flag	1	00,01,06	129	00,01,17,28
21	10	Frozen Counter – 16 bit without flag	1	00,01,06	129	00,01,17,28
22	0	Counter Event – Any Variation	1			
22	1 (default see note1)	Counter Event – 32 bit with flag	1	06,07,08	129,130	17, 28
22	2	Counter Event – 16 bit with flag	1	06,07,08	129,130	17, 28
22	5	Counter Event – 32 bit with flag and time	1	06,07,08	129,130	17, 28
22	6	Counter Event – 16 bit with flag and time	1	06,07,08	129,130	17, 28
23	0	Frozen Counter Event – Any Variation	1	06,07,08		
23	1 (default see note1)	Frozen Counter Event – 32 bit with flag	1	06,07,08	129,130	17, 28
23	2	Frozen Counter Event – 16 bit with flag	1	06,07,08	129,130	17, 28
23	5	Frozen Counter Event – 32 bit with flag and time	1	06,07,08	129,130	17, 28
23	6	Frozen Counter Event – 16 bit with flag and time	1	06,07,08	129,130	17, 28
30	0	Analog Input – Any Variation	1,22	00,01,06,07,08,17,27,28		
30	1	Analog Input – 32 bit with flag	1	00,01,06,07,08,17,27,28	129	00,01,17,28
30	2	Analog Input – 16 bit with flag	1	00,01,06,07,08,17,27,28	129	00,01,17,28
30	3 (default see note1)	Analog Input – 32 bit without flag	1	00,01,06,07,08,17,27,28	129	00,01,17,28
30	4	Analog Input – 16 bit without flag	1	00,01,06,07,08,17,27,28	129	00,01,17,28
30	5	Analog Input – Single prec flt pt with flag	1	00,01,06,07,08,17,27,28	129	00,01,17,28
32	0	Analog Input Event – Any Variation	1	06,07,08		
32	1 (default see note1)	Analog Input Event – 32 bit Without time	1	06,07,08	129,130	17, 28
32	2	Analog Input Event – 16 bit Without time	1	06,07,08	129,130	17, 28
32	3	Analog Input Event – 32 bit With time	1	06,07,08	129,130	17, 28
32	4	Analog Input Event – 16 bit With time	1	06,07,08	129,130	17, 28
32	5	Analog Input Event – Single prec flt pt without time	1	06,07,08	129,130	17, 28
32	7	Analog Input Event – Single prec flt pt with time	1	06,07,08	129,130	17, 28
40	0	Analog Output Status – Any Variation	1	00,01,06,07,08,17,27,28		
40	1 (default see note1)	Analog Output Status – 32 bit with flag	1	00,01,06,07,08,17,27,28	129	00,01,17,28
40	2	Analog Output Status – 16 bit with flag	1	00,01,06,07,08,17,27,28	129	00,01,17,28
40	3	Analog Output Status – Single prec flt pt with flag	1	00,01,06,07,08,17,27,28	129	00,01,17,28
41	0	Analog output	22	00,01,06,07,08,17,27,28		
41	1	32 bit Analog output	3,4,5,6	17,27,28	129	Echo of request
41	2	16 bit Analog output	3,4,5,6	17,27,28	129	Echo of request
41	3	Short floating point Analog output	3,4,5,6	17,27,28	129	Echo of request

DNP OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation must parse		RESPONSE Master must parse Outstation may issue	
Group Num	**Var Num**	**Description**	**Func Codes (dec)**	**Qual Codes (hex)**	**Func Codes (dec)**	**Qual Codes (hex)**
50	1	Time and Date – Absolute time	1	07	129	07
			2	07		
51	1	Time and Date CTO – Absolute time, synchronized			129,130	07
51	2	Time and Date CTO – Absolute time, unsynchronized			129,130	07
52	1	Time Delay – Coarse			129	07
52	2	Time Delay – Fine			129	07
60	1	Class Objects – Class 0 data	1	06		
60	2	Class Objects – Class 1 data	1	06,07,08		
			20,21,22	06		
60	3	Class Objects – Class 2 data	1	06,07,08		
			20,21,22	06		
60	4	Class Objects – Class 3 data	1	06,07,08		
			20,21,22	06		
80	1	Internal Indications – Packed format	2 (default see note3)	00 (index =4 or 7)		
110	String length	Octet String Object	1 (read), 22	00,01,06,07,08,17,27,28	129	00,01
			2 (write)	00,01,07,08,17,27,28		
No object (function code only)			13 (cold restart)			
No object (function code only)			23 (delay measure)			

Note 1: A Default variation refers to the variation responded when variation 0 is requested and/or in class 0, 1, 2, or 3 scans. Default variations are configurable; however, default settings for the configuration parameters are indicated in the table above.

Note 2: For static (non-change-event) objects, qualifiers 17 or 28 are only responded when a request is sent with qualifiers 17 or 28, respectively. Otherwise, static object requests sent with qualifiers 00, 01, 06, 07, or 08, will be responded with qualifiers 00 or 01. (For change-event objects, qualifiers 17 or 28 are always responded.)

Note 3: Writes of Internal Indications are only supported for index 4 or 7 (Need Time IIN1-4 or Restart IIN1-7)