


n° LM 2 CNF 2 001 UK		Ind K	UK Version	Visa O	12/11/2013
<div style="border: 1px solid black; padding: 40px; text-align: center;"> <p>FipDesigner for LINUX</p> <p>version 1.0</p> <p>User's manual</p> </div>					
Redactor	I.LANTERI	<div style="text-align: center;">  <p>35, Rue Tournefort - 75005 Paris - France</p> </div>			
Controller	C.CERCEAU				

This document cannot be used, reproduced or communicated without authorization of HLP.

Document history

Indice	Date	Page	Description
E	12/04/2000	All	Translation from the French version.
F	08/30/2005	All	Deletion management installation put back in a separate document. Deletion references in the mono medium
K	11/12/2010	All	Adaptation for LINUX
	11/11/2013	All	Integration of the new user interface screen-shots

This document cannot be used, reproduced or communicated without authorization of HLP.

This document cannot be used, reproduced or communicated without authorization of HLP.

CONTENTS

1. INTRODUCTION.....	10
1.1. FEATURES.....	10
1.2. CONFIGURATION FILES.....	10
1.2.1. OVERVIEW.....	10
1.2.2. CONFIGURATION PARAMETERS.....	10
2. PRINCIPLES OF OPERATION.....	11
2.1. FIPDESIGNER OPERATION MODES.....	11
2.2. USER INTERFACE OVERVIEW.....	11
2.2.1. CONFIGURATION STAGE.....	11
2.2.1.1. <i>Interface when launching FipDesigner.....</i>	<i>12</i>
2.2.1.2. <i>Interface when editing a configuration.....</i>	<i>12</i>
2.2.1.3. <i>Configuration file data.....</i>	<i>13</i>
2.2.1.4. <i>Automatic consistency check.....</i>	<i>14</i>
2.2.2. CONNECTION STAGE.....	14
3. USER COMMANDS.....	16
3.1. FILE MENU.....	16
3.1.1. "NEW".....	16
3.1.2. "OPEN...".....	16
3.1.3. "SAVE".....	16
3.1.4. "SAVE AS...".....	17
3.1.5. "CLOSE".....	17
3.1.6. "LIST OF MOST RECENT FILES".....	17
3.1.7. "EXIT".....	17
3.2. VIEW MENU.....	18
3.2.1. "NETWORK PANEL".....	18
3.2.2. "DATA PANEL".....	18
3.2.3. "ARBITER PANEL".....	18
3.2.4. "RUNNING PANEL".....	18

3.3. HELP MENU.....	19
4. CONFIGURATION STAGES.....	19
4.1. NETWORK PANEL – GENERAL SETTINGS.....	19
4.1.1. "TIME BASES" PARAMETERS.....	19
4.1.2. "BUS ARBITRATOR" SETTINGS.....	20
4.1.3. VARIOUS PARAMETERS.....	20
4.2. ARBITER PANEL - PROGRAMMING THE BUS ARBITRATOR.....	21
4.2.1. WRITING THE BUS ARBITRATOR PROGRAM.....	21
4.2.1.1. <i>Available instructions</i>	22
4.2.1.2. <i>Programming through the "Sequence1" or "Sequence2" windows</i>	22
4.2.1.3. <i>Programming by ASCII file</i>	28
4.2.2. EXAMPLE OF BUS ARBITRATOR PROGRAMMING.....	29
4.2.3. BUS ARBITRATOR STATES	30
4.3. DATA PANEL - THE FIP OBJECTS CONFIGURATION.....	32
4.3.1. EDITING AND CONFIGURING VARIABLES.....	33
4.3.1.1. <i>Description of the variable main parameters</i>	34
4.3.1.2. <i>Definition of the variable character (consumed / produced)</i>	35
4.3.1.3. <i>Time statuses</i>	36
4.3.1.4. <i>Choosing producing and consuming stations</i>	36
4.3.1.5. <i>Additional variable parameters</i>	38
4.3.2. EDITING AND CONFIGURING A MESSAGE.....	40
4.3.2.1. <i>Common messages parameters</i>	40
4.3.2.2. <i>Parameters specific to sent messages</i>	43
4.3.3. DATA FIELDS FORMAT.....	43
4.3.3.1. <i>Principle</i>	43
4.3.3.2. <i>Type specification table under windows</i>	45
4.3.3.1. <i>Type specification table under linux</i>	47
4.3.4. EDITING AN APERIODICAL SEND LIST.....	49
4.3.5. EDITING AND CONFIGURING STATIONS.....	50
5. CONNECTION STAGES.....	52
5.1. FIP CONNECTION/DISCONNECTION ACTIONS.....	52
5.1.1. CONNECT.....	52
5.1.2. DISCONNECT.....	52

5.2. BUS ARBITER CONTROL TOOLBAR.....	52
5.3. RUNNING PANEL.....	54
5.3.1. VARIABLE TAB.....	54
5.3.1.1. <i>Sending a variable</i>	54
5.3.1.2. <i>Viewing a consumed variable</i>	55
5.3.2. MESSAGE TAB.....	57
5.3.2.1. <i>Sending a message</i>	57
5.3.2.2. <i>Viewing a received message</i>	58
5.3.3. APERIODICAL VARIABLES TAB.....	60
5.3.4. MEDIUM CONTROL TAB.....	60
5.3.4.1. <i>Overview of communication management</i>	61
5.3.4.2. <i>Communication state byte format</i>	62
5.3.4.3. <i>Medium management commands</i>	62
6. EXAMPLES.....	63
7. ANNEX A: THE WORLDFIP BUS.....	64
7.1. PRINCIPLE.....	64
7.1.1. BUS ARBITRATOR.....	64
7.1.2. VARIABLES, VARIABLE IDENTIFIERS.....	64
7.1.3. POLLING TABLE.....	64
7.1.4. MESSAGING SERVICES.....	64
7.2. CIRCULATION OF A VARIABLE ON THE WORLDFIP NETWORK.....	64
7.2.1. MEDIUM ALLOCATION BY THE BUS ARBITRATOR.....	64
7.2.2. ACTIVATION OF PARTICIPATING ENTITIES (PRODUCER, CONSUMERS)...	64
7.2.3. DIFFUSION (REFRESHMENT NOTION).....	64
7.2.4. DATA ACQUISITION (PROMPTNESS NOTION).....	64
7.2.5. LOCAL BUFFER.....	66
7.3. PHYSICAL LAYER.....	66
7.3.1. TRANSMISSION MEDIUM.....	67
7.3.2. TRANSMISSION SPEED.....	67
7.3.3. TOPOLOGY.....	67
7.3.4. BIT TIME TBIT	67
7.3.5. BIT – LEVEL ENCODING.....	67
7.3.6. COMPOSITION OF A WORLDFIP FRAME.....	67

7.3.6.1. Presentation.....	67
7.3.6.2. Start sequence.....	67
7.3.6.3. Control and data field.....	68
7.3.6.4. End sequence.....	68
7.3.6.5. WorldFIP and FIP frames.....	68
7.4. DATALINK LAYER.....	68
7.4.1. GENERAL DESCRIPTION OF SERVICES.....	68
7.4.1.1. Transmission services.....	68
7.4.1.2. Transmission types.....	69
7.4.1.3. Addressing spaces.....	69
7.4.2. DATA FORMAT.....	70
7.4.2.1. Control and data field format.....	70
7.4.2.2. Medium allocating frames (ID_XX).....	70
7.4.2.3. Response frames (RP_XX).....	70
7.4.3. DELAY DEFINITIONS.....	71
7.4.3.1. Overview.....	71
7.4.3.2. Silence time T_s	71
7.4.3.3. Turnaround time T_r	71
7.4.4. THE BUS ARBITRATOR ENTITY.....	71
7.4.4.1. Overview.....	71
7.4.4.2. Identifier frames (ID_XX).....	71
7.4.4.3. Elementary cycle.....	71
7.4.4.4. Macrocycle.....	72
7.5. APPLICATION LAYER.....	73
7.5.1. DATA ENCODING AT APPLICATION LEVEL.....	73
7.5.1.1. Structure.....	73
7.5.1.2. Identification byte.....	73
7.5.1.3. Refreshment.....	73
7.5.1.4. Content bytes.....	73
7.5.2. ASYNCHRONOUS PROMPTNESS AND REFRESHMENT.....	73
7.5.2.1. Refreshment.....	73
7.5.2.2. Promptness.....	74
7.6. CONSTRUCTION OF A BUS ARBITRATOR MACROCYCLE.....	74
7.6.1. DEFINITION OF THE MACROCYCLE AND THE ELEMENTARY CYCLE.....	74
7.6.1.1. Macrocycle.....	75
7.6.1.2. Elementary cycle.....	75
7.6.2. REPRESENTATION OF THE MACROCYCLE.....	75
7.6.2.1. Overview.....	75
7.6.2.2. Rank.....	76
7.6.2.3. Maximum rank.....	76

7.6.3. CONSTRUCTION OF THE MACROCYCLE.....	76
7.6.4. TIME WINDOWS FOR APERIODICAL TRAFFIC.....	77
7.6.5. MACROCYCLE CONSTRAINTS.....	77

1.INTRODUCTION

1.1.FEATURES

FipDesigner is a local communication software conforming to the WorldFIP standard. Running under Windows 95, 98 and NT it is now available for LINUX. Main features are:

- 1- A network management function which:
 - May act as a bus arbitrator,
 - Enables dialog with any entity on the WorldFIP network, by adapting itself to all possible formats of data representation.
- 2- A development and application optimisation tool (with such capabilities as "step by step" operation).

Finally FipDesigner is a WorldFIP initiation tool, with a user friendly interface.

1.2.CONFIGURATION FILES

1.2.1.OVERVIEW

FipDesigner operates with configuration files compatible with all WorldFIP software and hardware from HLP Technologies.

The configuration files have the extension "cnf" and are used by FipEngine, FipDesigner and FipBusVIEW.

There are two versions:

- the "cnf3" version, compatible with FipDesigner and FipBusVIEW versions 3.X
- and the "cnf4" version compatible with FipEngine 1.X and 2.X, FipDesigner and FipBusVIEW versions 4.X.

FipDesigner version 4 (Windows) allows to read and create either of the two formats.

FipDesigner for LINUX uses only the cnf4 version.

1.2.2.CONFIGURATION PARAMETERS

The WorldFIP configuration files contain WorldFIP parameters for **one and only one PC/WorldFIP card**. Editing a configuration file with FipDesigner is equivalent to setting up all the communication parameters for the local station (PC).

The local WorldFIP configuration regroupes:

- General WorldFIP network parameters (transmission speed, turnaround time, silence time, etc.)
- Parameters of WorldFIP variables that are produced or consumed by the local station
- Parameters of WorldFIP messages that are emitted or received by the local station
- The program for one or two bus arbitrators.

2. PRINCIPLES OF OPERATION

When FipDesigner is launched, it first displays a dialog box presenting the PCIFIP cards it has detected. The user must choose and select one card in the combo box. When a card is selected by the user, its main characteristics are shown. This card will be the active card during the session. If the user clicks the "OK" button. If the user clicks the "Cancel" button, the software aborts.

Cards are indexed from one to four in accordance with the order they were inserted.

- Version is the card version.
- Speed is the PCIFIP card speed typically 1 Mbps.
- Serial number is the card serial number.
- Slot is the slot number in which the card is inserted

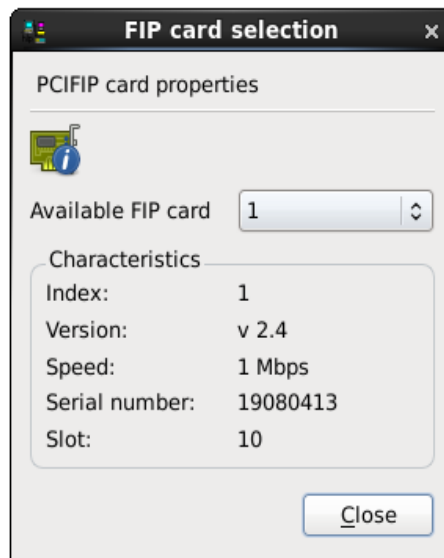


Figure 1: Selecting a card

2.1. FIPDESIGNER OPERATION MODES

FipDesigner operates in either of two modes:

- Configuration mode: to configure the PC for the WorldFIP network. It can then produce and consume variables, emit and receive messages and also support the bus arbitrator function, just like any other device on the network.
- Connection mode: to connect to the WorldFIP network and communicate with other stations.

The essential mechanisms of the WorldFIP standard are reviewed in the section 7 as an help for the first time user.

2.2. USER INTERFACE OVERVIEW

There are two stages when using FipDesigner:

- declaration of parameters and configuration of the network (configuration),
- connection to the network and data exchange (connection).

In what follows, we call these stages configuration stage and connection stage.

2.2.1. CONFIGURATION STAGE

This stage allows to edit and save the communication parameters of the user's WorldFIP station. The edition is done without being connected to WorldFIP. Connection to WorldFIP will be established during the connection stage.

2.2.1.1. Interface when launching FipDesigner

When FipDesigner is launched, after the user has selected a card, the interface is made of:

- a menu bar and two toolbars (horizontal and vertical),
- an empty window.

The commands accessible from the menu allow the user to load, or create, a WorldFIP configuration file.

2.2.1.2. Interface when editing a configuration

Once the configuration has been loaded, FipDesigner represents it as a tree in the main window.

Three panels are selectable:

- the "Network panel"
- the "Data panel"
- the "Arbiter panel".

These panels are docked and can be switched by clicking on vertical toolbar buttons.

.Tree

The tree represents the local station and its interaction with the network. Icons stand for the different network stations and the data exchanged in between them.

Please bear in mind that the configuration file can only be used to set-up only one station: **the local station**, that is the PC the user is working on.

The other stations are represented as an aid to better visualise the network and the data. This enable the user to better recreate the application.

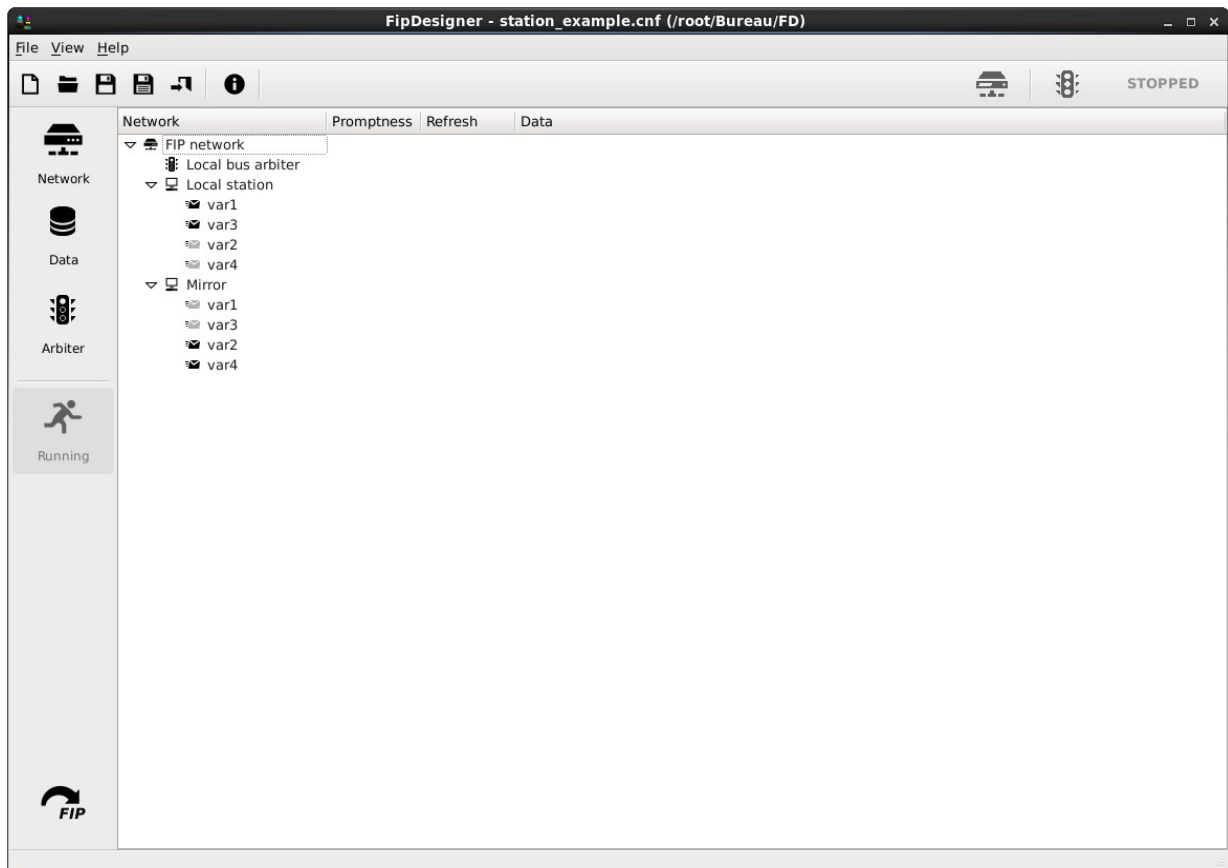


Figure 2: FipDesigner in configuration mode

.Toolbar, Menu bar, Contextual status

The menu bar allows the configuration files access ("file menu"), the opening of each FipDesigner panel ("view menu") and the help viewing ("help menu").

The horizontal toolbar is an another way of accessing to "file menu" and card information.

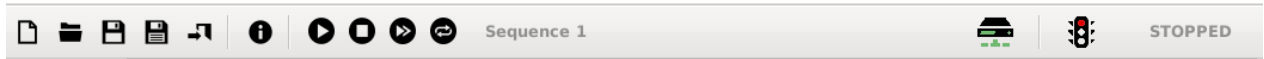


Figure 3: The horizontal toolbar

The horizontal toolbar is composed of four section: the "file" toolbar, the "card info" toolbar, the "arbiter management" toolbar and the "arbiter status" viewing.

The vertical toolbar is another way of accessing to "view menu" and connecting/disconnecting FIP.

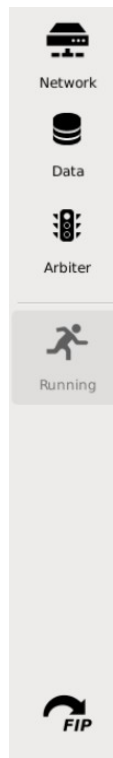


Figure 4: The vertical toolbar

2.2.1.3. Configuration file data

When a configuration has been edited, it can be saved in a "cnf" file.

The name of the current file is indicated in the title bar.

Data saved in this file can be classified in two categories:

- Local station parameters
- Additional data concerning the network graphical representation (i.e. the tree).

There are two sub-kinds of "cnf" files:

- "cnf 3" primarily used with FipDesigner 3.x and FipBusVIEW 3.x but compatible with newer versions,
- "cnf 4" used with FipDesigner 4.x and FipBusVIEW 4.x.

The "cnf 3" files format doesn't include the additional data concerning the tree. When using this format the tree will therefore contain the local station, and a mirror station whose function is to represent all the producers of data consumed by the local station.

FipDesigner for Linux, uses only the .cnf4 file.

.Local station parameters

These parameters are used to configure the local PC/WorldFIP card:

- The general WorldFIP parameters;
- The local bus arbitrator configuration;
- The configuration of variables produced and consumed by the local station;
- The configuration of messages emitted and received by the local station;
- The configuration of the aperiodical identifiers used by the local station.

.Facultative data for network graphical representation

.Network stations

The network is represented by the tree. The local station is always present on this tree. It is also possible to add other stations of the network to the tree, for visualisation purposes only. The parameters computed by FipDesigner are only related to the local station.

.Variables and messages

FipDesigner allows graphical representation of the variables and messages that are produced or consumed by the **local station**.

Graphical representation of variables and messages that are of no interest to the local station is not possible.

However, this limitation does not, of course, apply to the **bus arbiter program**. The bus arbitrator function of FipDesigner can match any user application. This is explained with more detail in paragraph 4.2.

.Variables

When the user configures a variable, it is compulsory that he indicates the corresponding values.

As an option, if the variable is either produced or consumed by the local station, the user can tell FipDesigner what the other stations interested in the variable are.

The information will then be used by the tree to place the icons "produced variable" and "consumed variable" under the appropriate stations.

.Messages

The same applies during the configuration of messages: it is mandatory for the user to indicate the source and destination identifiers of the message.

As an option, if the message is either emitted or received by the local station, the user can tell FipDesigner which other station is interested in the message.

This information will then be used by the tree to place the icons "sent message" and "received message" under the appropriate stations.

2.2.1.4. Automatic consistency check

FipDesigner carries out an automatic consistency control of the edited WorldFIP configuration. It is impossible to save a configuration on a "cnf" format file or to connect to the network if the configuration is not consistent. In this way two variables cannot possess the same identifier, a sent message can only be validated if the transmission line is correctly configured etc...

2.2.2. CONNECTION STAGE

This stage allows to establish the WorldFIP communication. Thanks to the "Running panel", the user has a global view on the state or the value of the entities which he has defined during the configuration stage.

When connected, it is possible to view the configuration of the network, but not to modify it.

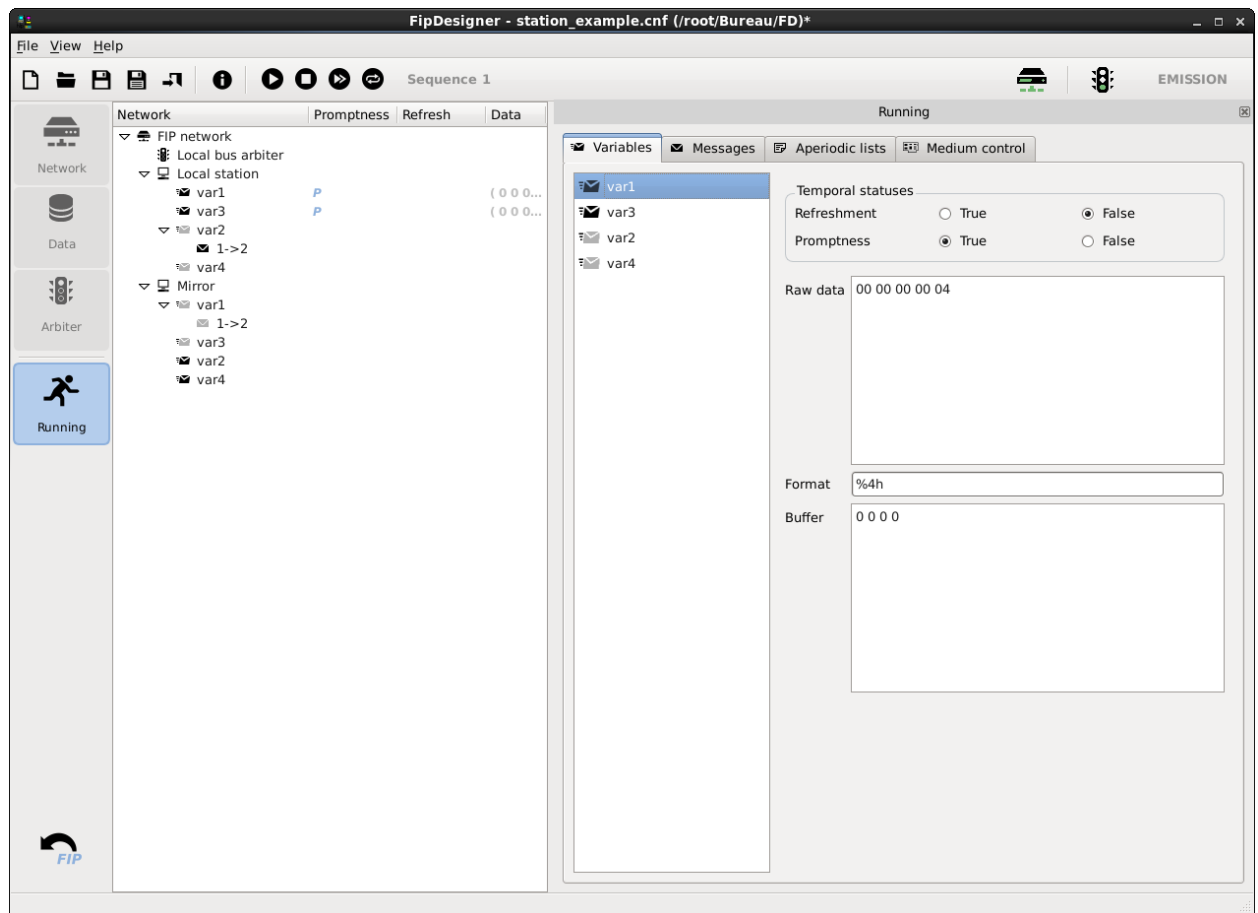


Figure 5: The Running panel

3.USER COMMANDS

The user can access a certain number of commands by the menu bar, the tool bar or the contextual menus.

It groups together all the program commands. Most commands execute a panel to input all the information necessary for the command. Some commands have an immediate action. FipDesigner dialog boxes are detailed in Paragraph 5.

- The "File" menu features all the existing commands operating on "cnf" files.
- The "View" menu allows to show or hide the different panels.
- The "Help" menu allows access to online help and to the "About box..." dialog.

3.1.FILE MENU

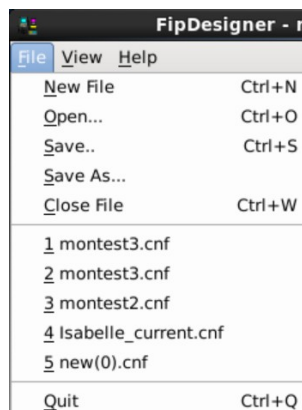


Figure 6: File menu

The set of commands from this menu is described here after. This menu also displays a list of the last 4 opened "cnf" files, on which the user can click for quick access.

3.1.1."NEW"

Also accessible by the button:



This command creates a new, empty configuration. The default name of this configuration is "new(i)" where "i" is the file index (from 0) . A minimum tree is displayed.

3.1.2."OPEN..."

Also accessible by the button:



This command allows to open a "cnf" file for editing. For the Linux version only the "cnf 4" file format is supported. The standard open file dialog box is called to carry out the operation.

3.1.3."SAVE"

Also accessible by the button:



This command saves the current edition of the WorldFIP configuration in a "cnf" file, in "cnf 4" format. If this configuration originates from an existing "cnf 4" file, saving will take place directly.

If the current edition is a new configuration which has not yet been saved, the "Save as..." dialog box is called. The reader can refer to the following section for further details.

3.1.4. "SAVE AS..."

Also accessible by the button:



This command allows to save the current edition of the WorldFIP configuration in a "cnf" file at "cnf 4" format. The standard dialog box "Save as..." is called. The user can also choose the folder and the name of the file that he is saving.

3.1.5. "CLOSE"

Also accessible by the button:



This command allows to close the current WorldFIP configuration.

3.1.6. "LIST OF MOST RECENT FILES"

This command allows to see and open directly the last opened files (up to 5)

3.1.7. "EXIT"

This command allows to close FipDesigner. If the WorldFIP configuration has changed during edition, a box prompting to save on file is displayed.

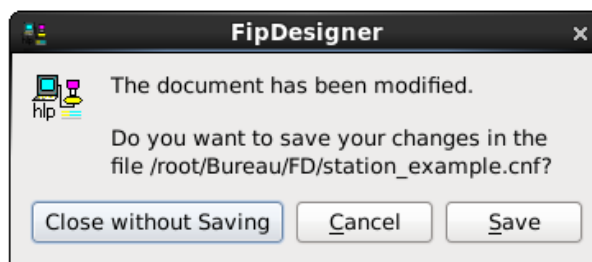


Figure 7: Standard "Save as" dialog box

If the user chooses "Save", the save dialog box is displayed. If the user chooses "Close without Saving", the application is stopped without saving the modifications on file. If the user chooses "Cancel", saving and application closing are cancelled.

3.2.VIEW MENU

This menu groups together the commands necessary for editing the WorldFIP settings.

3.2.1.“NETWORK PANEL”

Also accessible by the button (vertical toolbar):



This command shows or hides the panel “Network Panel” which allows the general WorldFIP network parameters to be entered (transmission speed, etc...). This panel is described in section 4.1.

3.2.2.“DATA PANEL”

Also accessible by the button (vertical toolbar):



This command shows or hides the panel “Data Panel” which allows to edit and configure stations and FIP communication objects (variables, messages, aperiodic lists). This panel is described in section 4.3

3.2.3.“ARBITER PANEL”

Also accessible by the button (vertical toolbar):



This command shows or hides the panel “Arbiter Panel” which allows the local bus arbitrator parameters to be entered. This panel is described in section 4.2.

3.2.4.“RUNNING PANEL”

Also accessible by the button (vertical toolbar):



This command shows or hides the panel “Running Panel” which allows the local bus arbitrator parameters to be entered. This panel is described in section 5.3.

3.3.HELP MENU

This menu gives access to the application on line help with the command "Help Topics". This command can also be accessed by the button. The command "About FipDesigner..." displays a dialog box which indicates the software version being used.

4. CONFIGURATION STAGES

The set of parameters which constitute a WorldFIP configuration are typed in using panels called up via the commands previously described. This section presents these panels.

4.1. NETWORK PANEL – GENERAL SETTINGS

This panel allows the user to edit general configuration settings.

The screenshot shows a 'Network' configuration window with the following settings:

- Time base:**
 - Speed: 1000 Kb/s
 - Turn around: 41,250 μ s
 - Silence: 264,000 μ s
- Bus arbiter enabled:** ☒
- Start BA delay:** 159,488 ms
- Priority level:** 0
- BA election time:** 116.48 ms
- Padding identifier:** 7530 hexa
- Frame type:** ☐ WorldFIP frames, ☒ FIP frames
- Clock multiplier:** ☒

Figure 8: Panel for editing the general settings of the WorldFIP network.

4.1.1. "TIME BASES" PARAMETERS

The following parameters form the base of correct communication on the bus. All stations connected on the same segment of network must use the same values.

- "Speed": transmission speed adopted. This setting must agree with the PC/WorldFIP card inserted in the PC. The three compatible standard speeds are: 31,25 kb/s; 1 Mb/s or 2,5 Mb/s.
- "Turn around": turn around time used. The user selects the required value from among the list of available values with the "+" and "-" buttons. Section 7.4.3 gives further details on this time.
- "Silence": silent time used. The user selects the required value from among the list of available values with the "+" and "-" buttons. Section 7.4.3 gives further details on this time.

4.1.2. "BUS ARBITRATOR" SETTINGS

- "Bus arbiter enabled": indicates when checked that the local station has a bus arbitrator program which can be activated at any moment. If this parameter is not checked, the local station cannot be the bus arbitrator and the parameters below appear in grey.
- "Start BA delay": start delay of the bus arbitrator. After the delay has passed, the bus arbitrator goes into ELECTION mode (see Figure 17). The start delay must be longer than the election stage of the bus arbitrators i.e. the delay must always be set higher than the longest election delay of all stations on the network. A default value of $8712 * T_{silence}$ allows the last condition to be respected.
- "Priority level": parameter used to calculate the election delay of the local station's bus arbitrator. When the bus arbitrator has passed its start delay, it goes into election stage. If during this delay it detects another bus arbitrator is already present, it will remain in ELECTION mode. If not it goes into the active state. For bus arbitrator redundancy, each station on the network must have a different election delay. When the active arbitrator stops, it is detected by all arbitrators in ELECTION mode and the station having the shortest election delay which will become the arbitrator only.
- "BA election time": displayed as read only. It indicates the election delay currently set. It is calculated as follows:

$$2 * (256 * (\text{Priority} + 1) + \text{Physical Address} + 3)$$
- "Padding identifier": value of the identifier used by the bus arbitrator in the WAIT state. It must not be produced or consumed by the network stations. By default it is set at 0x7530.

4.1.3. VARIOUS PARAMETERS

- "Clock multiplier": allows the range of turnaround and silence times available to be modified when the transmission speed is 1 Mb/s.

Multiplier	$T_{silence}$ min.	$T_{silence}$ max.	$T_{turnaround}$ min.	$T_{turnaround}$ max.
Checked	168 μ s	2,056 ms	13,25 μ s	253,25 μ s
Non checked	150 μ s	355 μ s	10 μ s	40,625 μ s

Tableau 1: Range of turn around and silence times according to the "Clock multiplier" setting. If this is checked, the available time reach longer delays.

- "WorldFIP frames": if this parameter is checked, the frame headers are those specified by the CENELEC EN61158-2 standard, see Figure 57. If not the headers are those shown in Figure 56.

4.2.ARBITER PANEL - PROGRAMMING THE BUS ARBITRATOR

This panel groups all the parameters necessary for programming the bus arbitrator. The user can edit up to two bus arbitrator programs or load a ready made program in an ASCII file. A "Default program" button allows a bus arbitrator to be generated automatically taking into account all the variables and messages of the local station.

The 'Arbiter' panel is a software interface for programming a bus arbitrator. It features two main columns for 'Sequence 1' and 'Sequence 2', each containing a text area for program code and a 'Default program' button. The central 'Instructions' area includes input fields for 'ID_DAT(identifier)', 'ID_MSG(identifier)', 'SEND_APER(date)', 'SEND_MSG(date)', and 'WAIT(date)'. Below these are fields for 'Minimum' and 'Maximum' values (0 and 255.938), a field for 'Enter wanted date (ms)', a field for 'Computed date (ms)', and a 'SUSPEND(event number)' field. At the bottom, there is a checkbox for 'Program the bus arbitrator with a file', a 'Browse' button, and a 'Slot time' field set to 62.5 µs. Navigation arrows and a 'WAIT(0)' field are also present.

Figure 9: "Arbiter" panel.

4.2.1.WRITING THE BUS ARBITRATOR PROGRAM

Writing the bus arbitrator program involves going back to describe the elementary transactions of a macrocycle. The bus arbitrator loops infinitely (as long as it is active) on this macrocycle.

- The "instructions" are the elementary transactions of the macrocycle.
- The "commands" allow this macrocycle to be controlled. These commands are implemented by one the following software: FipEngine, FipDesigner or FipBusVIEW.

4.2.1.1. Available instructions

The instructions available are the following:

ID_DAT(identifier)

Sends an *ID_DAT* frame with the identifier given as a parameter.

ID_MSG(identifier)

Sends an *ID_MSG* frame with the identifier given as a parameter.

SEND_MSG(date)

Processes the aperiodic message requests until the date given as a parameter is reached. If all the requests are processed before this date, *SEND_MSG* ends and the program goes on to the next instruction.

SEND_APER(date)

Processes the aperiodic transfer requests until the date given as a parameter is reached. If all the requests are processed before this date, *SEND_APER* ends and the program goes on to the next instruction.

SUSPEND(event number)

This is a special instruction which puts the bus arbitrator in a *SUSPENDED* state. When the bus arbitrator meets this instruction, it sends back the event number given as a parameter to the event queue. This instruction is normally intended for synchronisation with an external clock, in order to authorise very long macrocycles. The bus arbitrator program can be restarted using a specific command. It is possible to use *SUSPEND* as a stopping point in the bus arbitrator sequence.

WAIT(date)

Sends padding frames on the bus until the date given as a parameter is reached.

4.2.1.2. Programming through the "Sequence1" or "Sequence2" windows

The user can automatically generate a bus arbitrator or manually edit one, instruction by instruction. The automatically generated program can also be modified instruction by instruction.

The available instructions are displayed at the center of the panel. When the user selects one instruction, the entry for matching parameters appear.



Figure 10: Buttons "Add to sequence 1" and "Add to sequence 2"

The current selected instruction and parameters are displayed down. One click on the button "Add to sequence 1" adds the instruction at the sequence 1 and one click on the button "Add to sequence 2" adds the instruction at the sequence 2.

.Manually typing instructions

- 1- The checkbox "Program bus arbitrator with file" must be unchecked.
- 2- The user selects his instruction from the instructions list. The parameter for the instruction are then indicated:
 - For the "identifier" parameter:
 - An identifier of a variable which has already been configured. The user selects a variable from the list. The identifier of this variable will be automatically used as a parameter for the instruction.



Figure 11: Choosing an identifier number for the ID_DAT and ID_MSG instructions with an already configured variable.

- Another identifier. The identifier is not one of the local variables already configured. The user selects the “Other identifier” item in the combo box and enters the hexadecimal value identifier in the edit field down.

The screenshot shows a software interface titled "Instructions". It contains a list of instruction types: ID_DAT(identifier), ID_MSG(identifier), SEND_APER(date), SEND_MSG(date), WAIT(date), and SUSPEND(event number). The "ID_DAT(identifier)" instruction is highlighted. Below this list, a sub-dialog is open, showing a dropdown menu with "Other identifier" selected. Below the dropdown, there is a text input field containing "ID_DAT(5)".

Figure 12: Choosing a not configured identifier

- For the "date" parameter, the user indicates the desired time in milliseconds in the "wished" date display. The read only "minimum" and "maximum" times indicate the range that will be respected for the date. The date taken into account for the instruction is the one displayed in the sequence windows or indicated in "actual" time. The time counters for the date limit are referenced (reset to zero) at the start of each macrocycle.

Instructions

ID_DAT(identifier)

ID_MSG(identifier)

SEND_APER(date)

SEND_MSG(date)

WAIT(date)

Minimum	Maximum
0	255.938

Enter wanted date (ms)

100

Computed date (ms)

100

SUSPEND(event number)

WAIT(100 ms)

Figure 13: Entering a date for the WAIT instruction

- For the "event number" parameter, the spinbox allows the user to choose a value from 1 to 255.

Figure 14: Entering an event number for the SUSPEND instruction.

- 3- Once the parameter is set, it is left to the user to insert the instruction into one of the two bus arbitrator programs (sequence 1 or sequence 2). In order to do that he must select the insertion point of the instruction. The actual insertion will be made just before the selection, when the user clicks on the “Insert” button.
- 4- The user continues to type the program:
 - Insertion of new instructions.
 - Deletion of one or more instructions with the “Delete” context menu.

When programming is finished, the user clicks on “Local save” to validate the two programs.

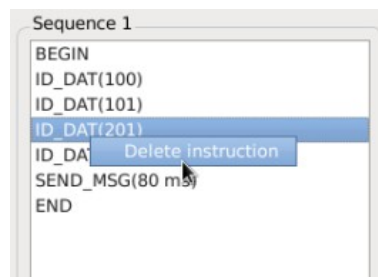


Figure 15: Deleting one instruction.

.Automatic generation of a bus arbitrator program

The "Default program" button allows to automatically generate a bus arbitrator program. Automatic generation creates an elementary cycle which is also a macrocycle, i.e. all the variables have the same periodicity which is equal to the duration of a macrocycle.

Automatic generation lists all the variables locally configured and all the periodic messages for generating periodic traffic. At this stage, user parameters are required in order to complete the generation process. The "Default program parameters" dialog box is displayed.

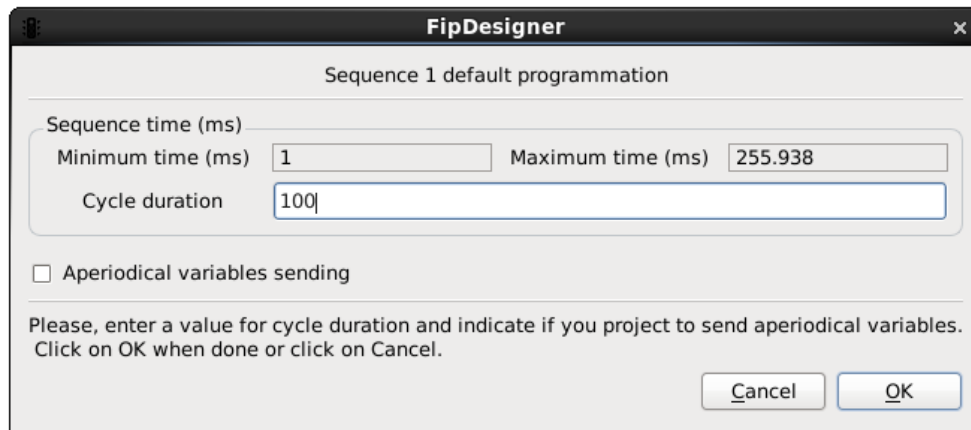


Figure 16: Setting the parameters for the automatic generation of a bus arbitrator program.

This dialog box requires the periodicity of the macrocycle. The user indicates the periodicity in milliseconds in "Cycle duration". The indicated value must fall between the limits "Minimum time" and "Maximum time". The minimum time displayed is that calculated with the detected periodic traffic. In addition the user indicates if he wants a window for sending aperiodic variables.

The generation process checks if local aperiodic messages exist in order to create an aperiodic message window. An aperiodic variable sending window is created if the "Aperiodical variables sending" is checked. These aperiodic variable or message windows are placed after the periodic traffic until the "Cycle duration" has passed. If there is none, a temporisation waits for the "Cycle duration" to pass.

When the program is generated, it is displayed in the corresponding sequence window. The user can modify it manually.

4.2.1.3. Programming by ASCII file

- 1- The user must edit an ASCII file containing a bus arbitrator program. The programming files of the bus arbitrator are simple text files. Their format is as follows:

```
Comment zone

START
ID_DAT(XXXX)
ID_DAT(YYYY)
.
.
.
SEND_MSG(XXXX)
NEXT_MACRO_2()
END
```

The program will be interpreted between the key words **BEGIN** and **END**.

The authorised instructions appear in section 4.2.1.1.

The instruction arguments must be given in hexadecimal.

The "date" argument must be expressed as multiples of "slot time" and in hexadecimal (the "slot time" parameter is the one indicated in the "Bus arbitrator programming" dialog box). The maximum value allowed is 0x0fff. Any higher value is simply masked with 0x0fff. The time counters for the date limit are referenced (reset to zero) at the start of each macrocycle.

The "identifier" parameter is directly the value of the identifier in hexadecimal.

The instruction **NEXT_MACRO_2()** must be added to the end before the key word **END**.

When the program is finished, it must be saved in a file with the extension ".ba".

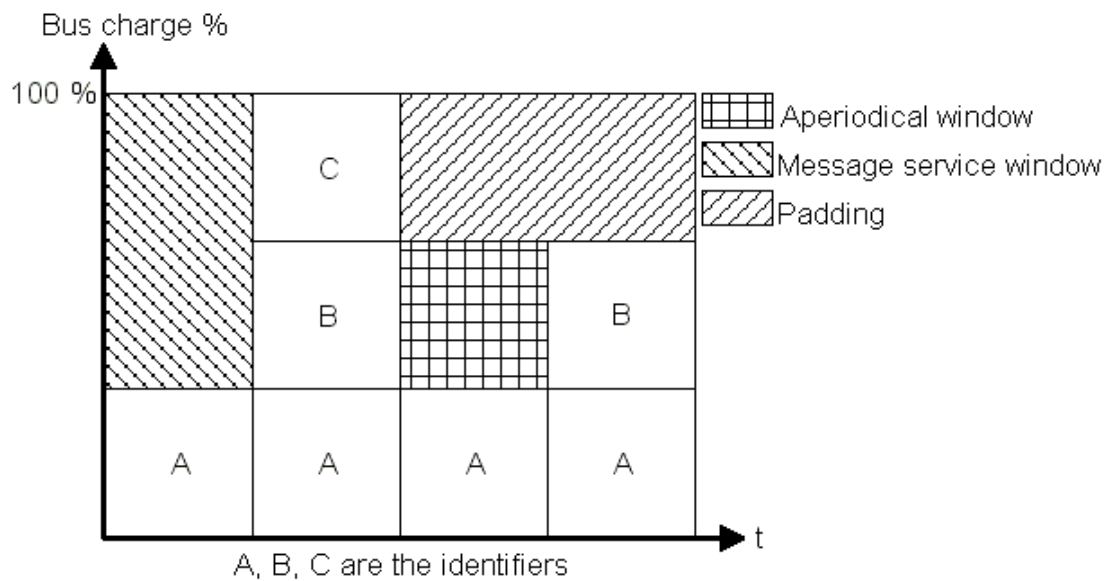
- 2- The user specifies the parameter "Program bus arbitrator with file" in the "Bus arbitrator programming" panel. The other parameters appear in grey except for "Slot time" which is used to calculate the arguments "date". The user indicates the pathname of the ASCII file in "Program bus arbitrator with file" or looks for it manually by clicking on "Browse...".

Notes:

- An ASCII file can only contain one bus arbitrator program.
- The data saved in the configuration file "cnf" is only the path to the "*.ba" format file, and not its contents.

4.2.2.EXAMPLE OF BUS ARBITRATOR PROGRAMMING

The following macrocycle will be programmed:



This cycle will be programmed by the following bus arbitrator sequence:

```
ID_DAT(A);
SEND_MSG(3*T/12);
WAIT(3*T/12);
ID_DAT(A);
ID_DAT(B);
ID_DAT(C);
ID_DAT(A);
SEND_APER(8*T/12)
WAIT(9*T/12);
ID_DAT(A);
ID_DAT(B);
WAIT(T);
```

T is the duration of the macrocycle. N.B.: the message service window of the first elementary cycle is also a transfer window for aperiodic variables.

4.2.3.BUS ARBITRATOR STATES

For memorising purposes and for clarifying these notes the eight possible states in which an active bus arbitrator can be found are presented here:

- **STOPPED:** the bus arbitrator is not active, nothing is transmitted by it on the network.
- **STARTING:** the bus arbitrator is starting up. Its start delay elapses.
- **IDLE:** the local bus arbitrator starts up and goes into election mode. Nothing is transmitted on the network during this stage. It tries to detect some activity on the network. At the end of this stage, it automatically goes into EMISSION mode if no active bus arbitrator has been detected. If not will remain in this state as long as the detected active bus arbitrator remains active.
- **PENDING:** the bus arbitrator is suspended following a SUSPEND instruction. In this state it sends padding frames.
- **WAIT:** the bus arbitrator waits until the specific time in the WAIT instruction has elapsed. In this state it sends padding frames.
- **SENDING:** the bus arbitrator sends the *ID_DAT* and *ID_MSG* frames of the aperiodic traffic.
- **APER_WND:** the bus arbitrator sends aperiodic variable traffic until the time specified in the *SEND_APER* has elapsed. If there is no aperiodic variable traffic it automatically goes to the next instruction of the macrocycle.
- **MSG_WND:** the bus arbitrator sends aperiodic message traffic until the time specified in the *SEND_MSG* instruction has elapsed. If there is no aperiodic message traffic, it automatically goes to the next instruction in the macrocycle.

Status machine of the bus arbitrator

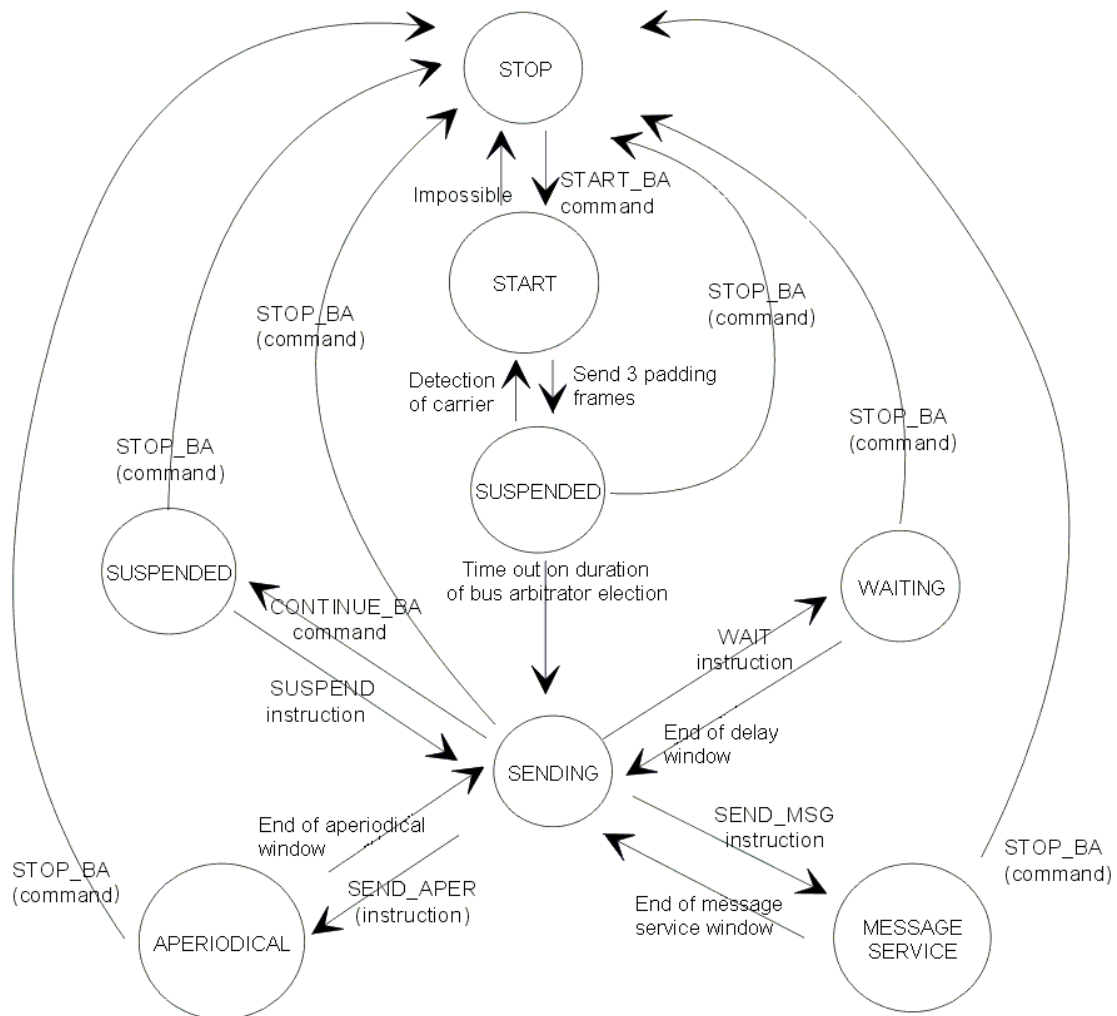


Figure 17: Different states of the bus arbitrator.

4.3. DATA PANEL - THE FIP OBJECTS CONFIGURATION

The “data panel” allows the user to edit and configure stations and FIP communication objects like variables, messages, aperiodic lists. This panel is mainly composed of two parts:

- up: choice of the current type of object (variable, message, aperiodic list, station) by tabs.
- down: list of known objects of the selected object type and tree view of the network. The characteristics of the current selected object in the list are displayed.

On the right side, a vertical tool bar allows the user to perform main operations : like creating, deleting, duplicating.

When modifications are performed, the local network tree, lists and fields are updated. Modifications are automatically locally stored but the configuration file is not modified. If the user wants to save the changes in the configuration file, he must operate the command save.

At start, the variables list is displayed by default.

From this panel, it is possible to operate the following operations:

- Creation: creates an object with the type of the current selected tab.
- Deletion: deletes the current selected object.
- Duplication: duplicates the current selected object (not applicable to stations objects).



Figure 18: (Creation, Deletion, Duplication) Buttons of the “Data” panel toolbar

The 'Data' panel is a configuration window for variables. It includes tabs for Variables, Messages, Aperiodic lists, and Stations. The 'Variables' tab is active, showing configuration for 'var1'. The 'General' section includes Name, Identifier (hexadecimal), and Format. The 'Frame' section includes Length and PDU type (hexadecimal). The 'Type' section has radio buttons for Produced and Consumed. The 'Stations' section has buttons for Production and Consumption, and dropdowns for Mirror and Local station. The 'Messages service' section has checkboxes for Reception and Emission, and radio buttons for Aperiodic and Periodic. The 'Temporal statuses: periods in ms' section has checkboxes for Refreshment and Promptness, and a text field for 500. At the bottom, there are two panels: 'Variables list' showing var1, var3, var2, and var4; and 'Network tree' showing a hierarchy of FIP network, Local bus arbiter, Local station, and Mirror, with variables listed under each.

Figure 19: Data panel.

4.3.1.EDITING AND CONFIGURING VARIABLES

Figure 20: Configuration of variables.

4.3.1.1. Description of the variable main parameters

- **"Name"**: name of the variable. This must not exceed 32 characters for a "cnf 4" format and 30 characters for a "cnf 3" format.
- **"Identifier"**: identifier of the variable. It cannot be any value. Some values are reserved. There are 3 distinct kinds of identifier:
 - Physical allocation, a value is attributed according to the physical address of the station; the lower byte of the identifier is the physical address. This allows 64 identifiers to be defined (zone 1, 2, 3 and 5) 32 of these are for network management.
 - Global allocation, the identifiers are universally known on the network; they are independent of the physical address of the station and of the user process; 4096 identifiers are defined in zone 6.

- Free allocation, the values are attributed according to the users needs (zone 4, 7 and 8).
- **"Length"**: length of the variable's useful data field. This does not include the PDU type byte, the length byte and the refreshment byte (if there is one). The maximum length is therefore 126 bytes if there is no refreshment or 125 if there is one.
- **"PDU type"**: the value of this byte indicates the nature of the data transported. This cannot be given any value. A user variable uses a 0x40 type PDU. For network management variables (for example presence or identification) the PDU type equals 0x50.
- **"Format"**: specification string of types used in the useful data of the variable. It is only used with FipDesigner and allows a quick view of the contents of the variable with the software. The type specification is detailed in Section 4.3.3.

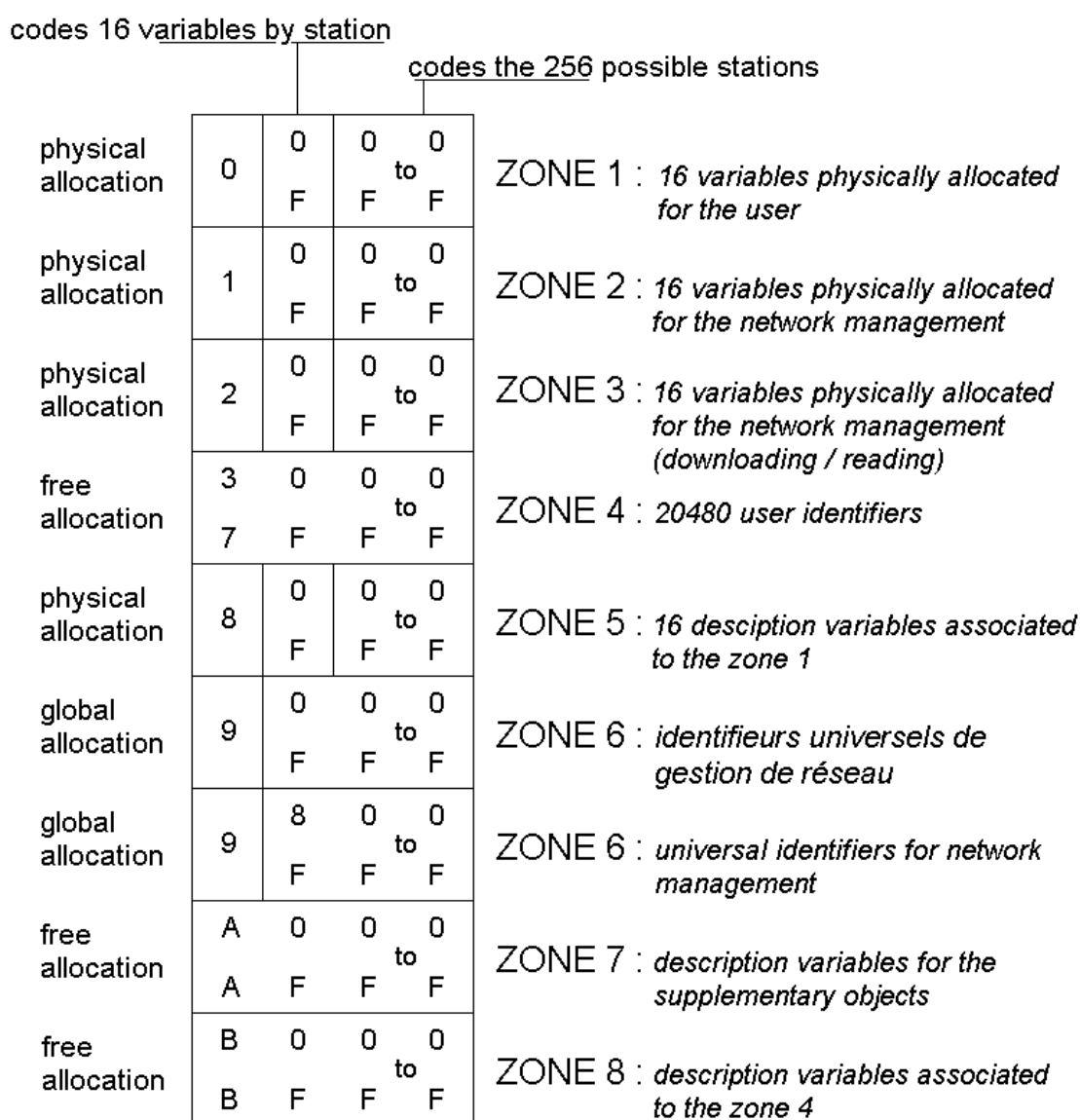


Figure 21: Sharing identifier space.

4.3.1.2. Definition of the variable character (consumed / produced)

The variable being edited is always a variable produced or consumed by the local station.

The user has the choice of viewing all stations which interact with the variable being edited. To do that he must complete the lists proposed by the "Producing station" and "Consuming stations". The tree will update all concerned stations by adding a variable icon on each of them. In this way the tree forms an accurate representation of the user's WorldFIP application.

The user may want to only view the variable on the local station, either because he does not know the other stations or because he does not want to create a station icon for them. In this case, he does not select any station in the "Producing station" dialog box in the case where the variable is consumed, or "Consuming stations" in the case where the variable is produced. "Not represented" is displayed in the place of a station name. The tree will only update the local station by adding a variable icon.

4.3.1.3. Time statuses

.Refreshment

For a produced variable, the user can request the elaboration of the refreshment status by clicking on the "Refreshment" checkbox. In this case, he must specify the period during which the refreshment is valid in the corresponding edition box.

The refreshment edition box may remain grey because the refreshment has no meaning for a consumed variable. Meanwhile, the user must specify if a refreshment status exists on this variable, because this adds a byte to the received frame. The variable will not be received if this information does not conform.

.Promptness

For a consumed variable, the user can ask for the elaboration of the promptness status by clicking on the "Promptness" checkbox. In this case, he must specify the period during which the promptness is valid in the corresponding edition box.

The promptness edition and check boxes may remain grey because the promptness has no meaning for a produced variable.

4.3.1.4. Choosing producing and consuming stations

The variable being edited is always a variable produced or consumed by the local station.

The user has the choice of viewing all the stations which interact with the variable being edited. To do that he must complete the lists proposed by the **Producing station** and **Consuming stations** lists. The local tree updates all concerned stations by adding a variable icon on each of them. In this way the tree forms an accurate representation of the user's WorldFIP application.

The user may want to only view the variable on the local station, either because he does not know the other stations or because he does not want to create a station icon for them. In this case, he does not select any station in the local tree. The label **Not represented station** is displayed in the place of a station name.

If the variable is produced, the combobox in the production field displays the value "Local station" and the combobox in the consumption field displays the value "Not represented station".

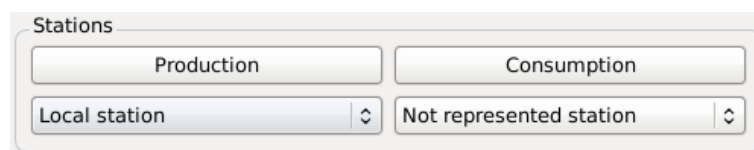


Figure 22: Producing and consuming stations for a produced variable.

To select the consuming station for a produced variable, select it in the network tree and press the right mouse button. The context menu allows to choose a station for consumer.

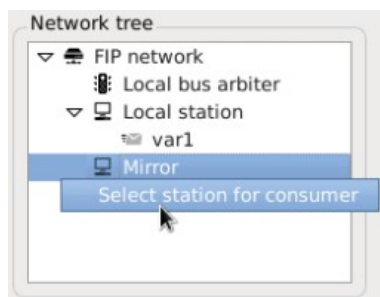


Figure 23: Choosing a consuming station for a produced variable.

If the station is chosen, the variable is added as a sub item of the station in the local tree.

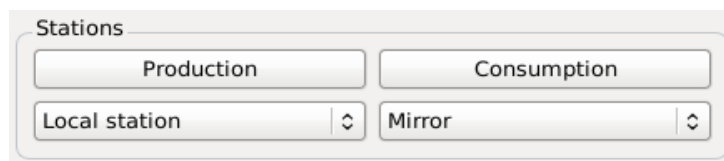


Figure 24: Choosing a consuming station for a produced variable.

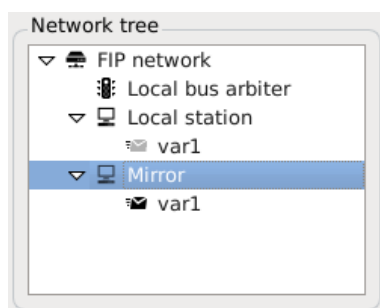


Figure 25: Choosing a consuming station for a produced variable.

If the variable is consumed, the combobox in the consumption field displays the value "Local station" and the combobox in the production field displays the default value "Not represented station".

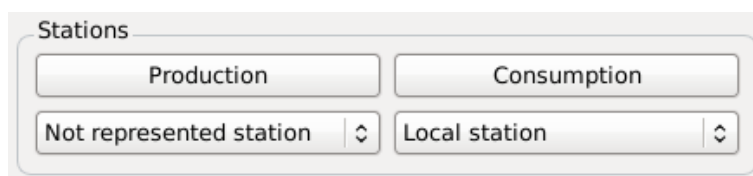


Figure 26: Producing and consuming stations for a consumed variable.

To select the producing station for a consumed variable, select it in the network tree and press the right mouse button. The context menu allows to choice the station for producer.

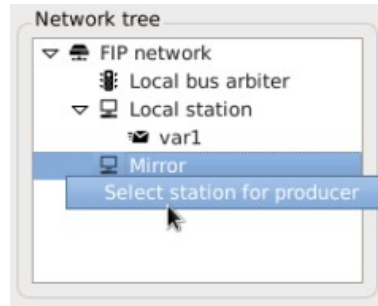


Figure 27: Choosing a producing station for a consumed variable.

If the station is chosen, the variable is added as a sub item of the station in the local tree.

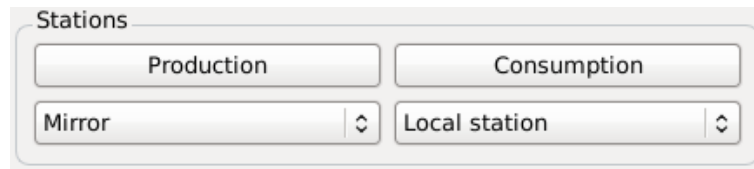


Figure 28: Choosing a producing station for a consumed variable.

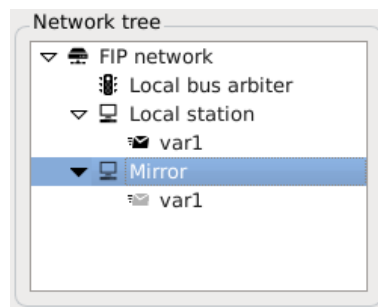


Figure 29: Choosing a producing station for a consumed variable.

4.3.1.5. Additional variable parameters

The variable is the object which allows to manage the transmission queues of the local station to receive messages. Some parameters must be set so that these roles can be fulfilled:

- **"Message service – Enable reception"**: a variable identifier can be a message receiver. If the option is not checked, the station will not receive the messages sent to this identifier.
- **"Message service – Enable emission"**: the transmission of a message is an asynchronous event which must be processed by a transmission queue. Checking this option allows the variable to avail of a message transmission queue. The queue can be reserved for an aperiodical or periodical message transmission. This choice is made by checking one of two "Type (emission)" options.

- **"Message service – Type (emission) – Aperiodical"**: if this option is checked, the variable uses an aperiodic message transmission queue. Even though the local station can only use one queue of this type, several variables can be associated with the queue. This option can only be checked for a produced variable. In fact, the emission request is made on the response frame of this variable by sending a *RP_DAT_MSG* frame. The identifier of the variable authorised to emit aperiodic messages has nothing to do with the source and destination identifiers of a stacked message. This variable simply offers a communication channel for the messages being sent.
- **"Message service – Type (emission) – Periodical"**: if this option is checked, the variable uses a periodical message transmission queue. The local station can use eight queues of this type at most. If eight variables are already using this option, a warning dialog box appears and it is impossible to save the variable. As before, the variable must be produced by the local station and its identifier has nothing to do with the source and destination identifiers of a stacked message. This variable simply offers a communication channel for the messages being sent.
- **"Enable aperiodical requests using this variable"**: if this option is checked, the variable is associated with the queue of aperiodic variable transfer requests. Although the local station can only use one queue of this type, several variables can be associated with this queue. This option can only be checked by a produced variable. In fact, the transfer request is done on the response frame of this variable by a *RP_DAT_RQ* frame.

4.3.2.EDITING AND CONFIGURING A MESSAGE

To display configured messages, the user must just click the “Messages” tab.

The screenshot shows the 'Data' window with the 'Messages' tab selected. The window has a title bar 'Data' and a close button. Below the title bar are four tabs: 'Variables', 'Messages' (selected), 'Aperiodic lists', and 'Stations'. On the right side of the window, there are three icons: a plus sign, a minus sign, and a refresh icon.

The 'Messages' tab contains the following sections:

- General:**
 - Name: msg
 - Format: %10s
- Source:**
 - Identifier: 100
 - Segment: 0
- Destination:**
 - Identifier: 200
 - Segment: 0
- Type:**
 - ☒ Received
 - ☐ Transmitted
- Mode:**
 - ☐ Periodical
 - ☒ Aperiodical
 - Identifier of the variable associated with the transmission: (empty dropdown)
- Event number:** 1
- Acknowledged message:**
 - ☐ Yes
 - ☒ No

At the bottom of the window, there are two panels:

- Messages list:** A list box containing 'msg'.
- Network tree:** A tree view showing the network structure:
 - FIP network
 - Local bus arbiter
 - Local station
 - var1
 - var3
 - var2
 - msg (selected)
 - var4
 - Mirror

Figure 30: Editing messages.

4.3.2.1.Common messages parameters

- **"Name"**: name of the message. This cannot exceed 32 characters for a "cnf 4" format and 30 characters for a "cnf 3" format.
- **"Format"**: format string used to display the message data. It is only used with FipDesigner and allows a quick view of the message contents with this software. If the user does not have this software, it is not necessary to specify a value here. The type specification is detailed in section 4.3.3.

- **"Source"**: source address of the message. This is made up of a segment number and the identifier of the source variable of the message:

Figure 31: Configuring the source and destination identifiers of a message.

- "Segment": the WorldFIP network can be broken down into segments of 32 stations at the most. The segment number where the source variable is, is indicated here.
- "Identifier": identifier of the source variable of the message. If it is a sent message, the source variable is a local produced variable.

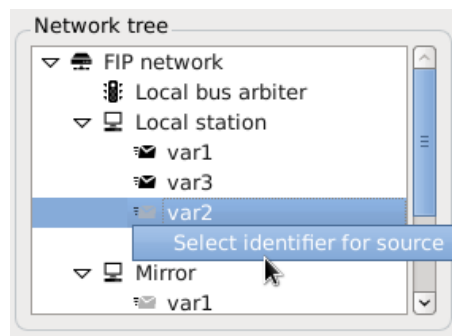


Figure 32: Configuring the source identifier of a sent message.

The user selects the variable in the local network tree. The icon of a *sent message* appears under the variable selected from the local station.

If it is a received message, the source identifier is a variable from a station different from the local station. The source variable may be an already configured consumed variable. In this case the user selects the variable in the local network tree.

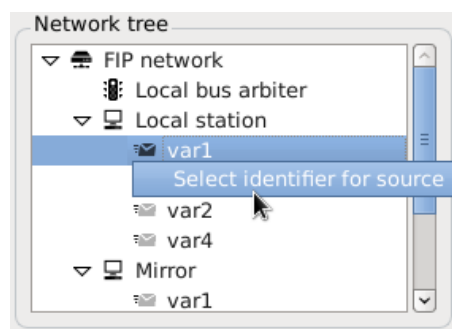


Figure 33: Configuring the source identifier of a received message.

In other cases, the user enters directly the identifier.

- **"Destination"**: address of the message destination. This is made up of segment number of the identifier of the message destination variable:

- "Segment": the WorldFIP network can be broken down into segments of 32 stations at the most. The segment number where the receiver variable is, is indicated here.
- "Identifier":
 - 1) If it is a sent message, the displayed box depends on the nature of the message service. In the case of an **acknowledged** message, there is only one destination station. The user selects the value of the destination identifier. This may be an identifier that has already been configured. The user can select a configured consumed variable in the local network tree. The *message received* icon will be displayed under the selected destination variable. The value of the destination identifier can be a non-configured value. The user enters the value directly into the "Identifier" field. In this case since no station has been selected, no *message received* icon will be displayed.

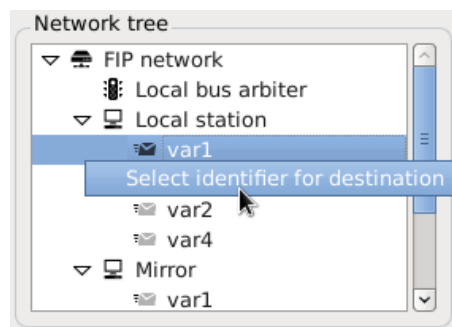


Figure 34: Choosing a configured destination variable for a transmitted message.

In the case of a **non acknowledged** sent message, the destination stations can be multiplied. FipDesigner cannot know if all the stations having a destination identifier are really authorised to receive a message. In case of doubt, FipDesigner does not display *message received* under the potential destination stations. The choice of the user is therefore limited to indicating the value of the destination identifier.

- 2) If it is a received message, the user selects a local variable in the network tree. This variable must be **authorised to receive**. The message icon is added at the network tree under the variable.

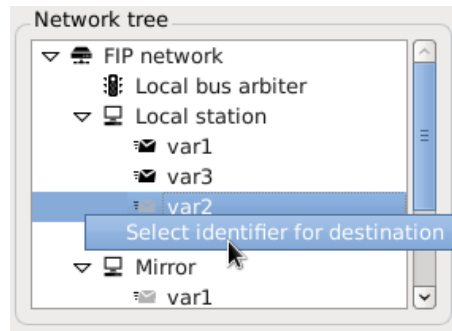


Figure 35: Choosing the destination identifier of a received message.

4.3.2.2. Parameters specific to sent messages

- **"Mode"**: the user checks one of two options, "Aperiodical" or "Periodical". If he selects periodical he then indicates which variable has access to the transmission queue which will be used. To do that the "Identifier of the variable associated with transmission" display, lists the variable identifiers authorised to transmit the type of message selected:
 - In the "Aperiodical" case, several variables may be authorised but there is only one transmission channel which is why the field is shaded. The identifier of at least one of these variables must be in the bus arbitrator program because it will be at the base of the transmission of the request `ID_DAT(authorised identifier) – RP_DAT_MSG`.
 - In the "Periodical" case, eight variables at most can be authorised. The user selects them from among those listed. If there is no identifier listed, it is necessary to authorise a variable for periodic message emission before the configuration of such a message can be saved.
- **"Acknowledgement"**: defines if the sent message is to be acknowledged by the destination. If this option is checked, the destination of the message produces an acknowledgement frame `RP_ACK`. As a consequence, in the acknowledge case, the message is only transferred from point to point, and it may only have one destination. Multiple destinations would cause a collision on the network during the emission of an acknowledgement. In the case of no acknowledgement however, the destination does not produce an acknowledgement frame. There can be many destination stations.
- **"Event number"**: number of the event which will be generated at the local station level during message transmission. It must fall between 1 and 251.

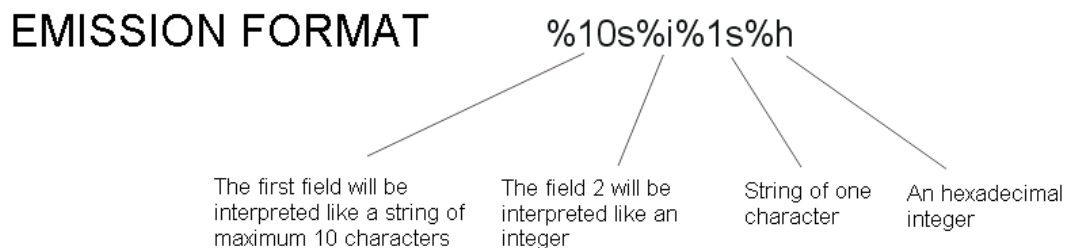
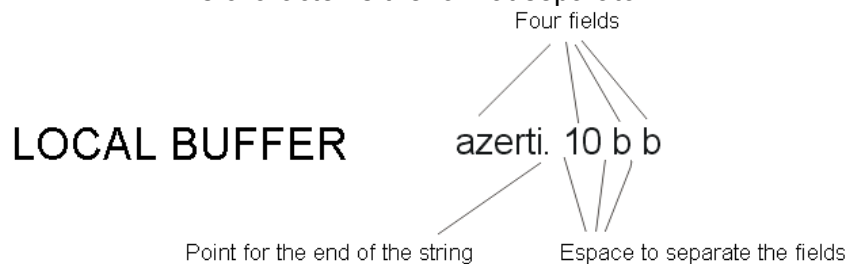
4.3.3. DATA FIELDS FORMAT

4.3.3.1. Principle

The formats are designed to allow the user to be able to interpret the data received on the bus regardless of the production station coding, and to be able to send data the way a consumer expects. The formats are used during typing or reading of the data field by the FipDesigner software. They are saved in the configuration file as complementary information. Their definition is useless outside the use of this software. The formats give flexibility to the use of FipDesigner which can be adapted to all the accompanying standards of any WorldFIP entity. The formats are made up of a string of alphanumerical characters which contain the format specifications.

These have the following form: % [size] [type]

% This character is the format separator.



[size] Optional specification. By default it is equal to 1 (except for the "s" format where it is mandatory).

[type] A character giving the conversion type.

Emission format:

- For a produced variable, the length of the data field produced is equal to that of the variable. If the format string gives a byte length inferior to that of the variable, the produced data field will be completed by padding bytes. The format string must not give a byte length greater than the variable length.
- For an emitted message, the format string gives the length of the message emitted by FipDesigner. Formatting stops if the format is invalid or after the format specifications have been used up.
- In all cases: the number of data fields indicated by the user must correspond exactly to the number of specifications of the format string. The field separator is a blank, except for character strings. This is so that the (0x20 ASCII) blanks can be sent on the bus with the format specification string of characters (%*ns* where *n* is the length of the character string). The character strings must be ended by a fullstop if their length is inferior to the length specified in the format, so that the field can be completed by padding bytes.

Reception formats:

Data formatting does not impose a length on the formatted data. If the length in bytes taken up by the format string is less than the real size of the data, the remaining data will be ignored.

4.3.3.2. Type specification table under windows

Specification	Argument	Frame length	Displayed length
i or I	Integer	n*2	variable
l or L	Long integer	n*4	variable
f or F	Floating	n*4	variable
g or G	Scientific	n*4	variable
u or U	Unsigned integer	n*2	variable
h or H	Hexadecimal byte	1	2
c or C	Character	ASCII	1
s or S	Character string	n	variable
z or Z	INTEL integer (inverted)	n*2	variable
T or T	Long INTEL integer (inverted)	n*4	variable
w or W	Floating INTEL (inverted)	n*4	variable

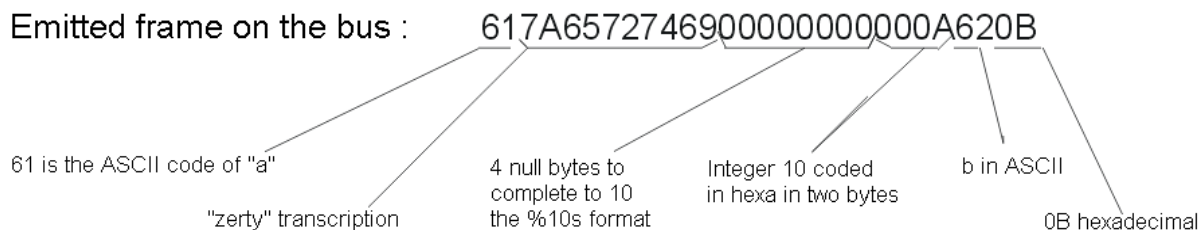
Table 2: Available type specifications (*n* is the size specified by the format field size).

NB: the format %*n*[type] is equivalent to *n* times %[type] except for character chains.

Example:

The user enters the following format and local buffer into FipDesigner.

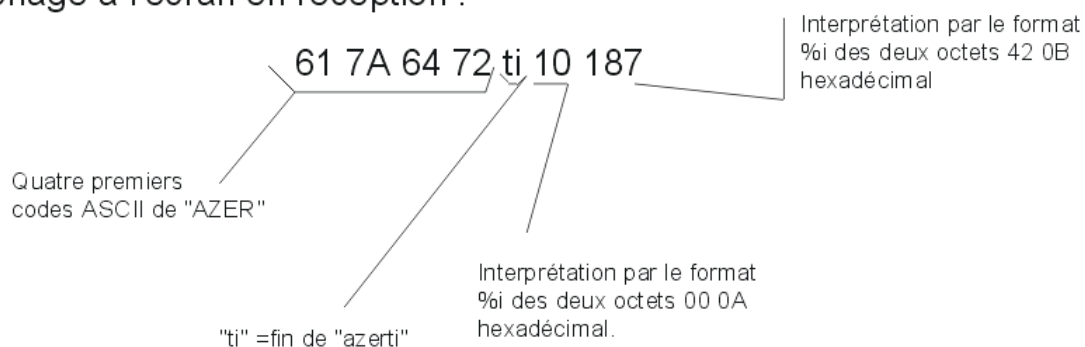
This results in:



The same frame can be interpreted differently on reception:

FORMAT DE RECEPTION : %4h%6s%i%i

Affichage à l'écran en réception :



4.3.3.1. Type specification table under linux

Windows FipDesigner and Linux FipDesigner can communicate but Linux has more available formats.

NB- Little endian is the Intel format and big endian is the Motorola format.

Specification	Type
H or h	Hexadecimal byte
C or c	ASCII character
S or s	String
K or k	Unsigned 16 bits integer little endian
U or u	Unsigned 16 bits integer big endian
Z or z	Signed 16 bits integer little endian
I or i	Signed 16 bits integer big endian
D or d	Unsigned 32 bits integer little endian
V or v	Unsigned 32 bits integer big endian
T or t	Signed 32 bits integer little endian
L or l	Signed 32 bits integer big endian
O or o	Unsigned 64 bits integer little endian
P or p	Unsigned 64 bits integer big endian
Q or q	Signed 64 bits integer little endian
X or x	Signed 64 bits integer big endian
W or w	32 bits float little endian
F or f	32 bits float big endian
J or j	64 bits double little endian
G or g	64 bits double big endian

Table 3: Available type specifications under Linux

In all the windows where the format field is present, you can access the help format panel if you click in the format edition field and then key "Shift F1".

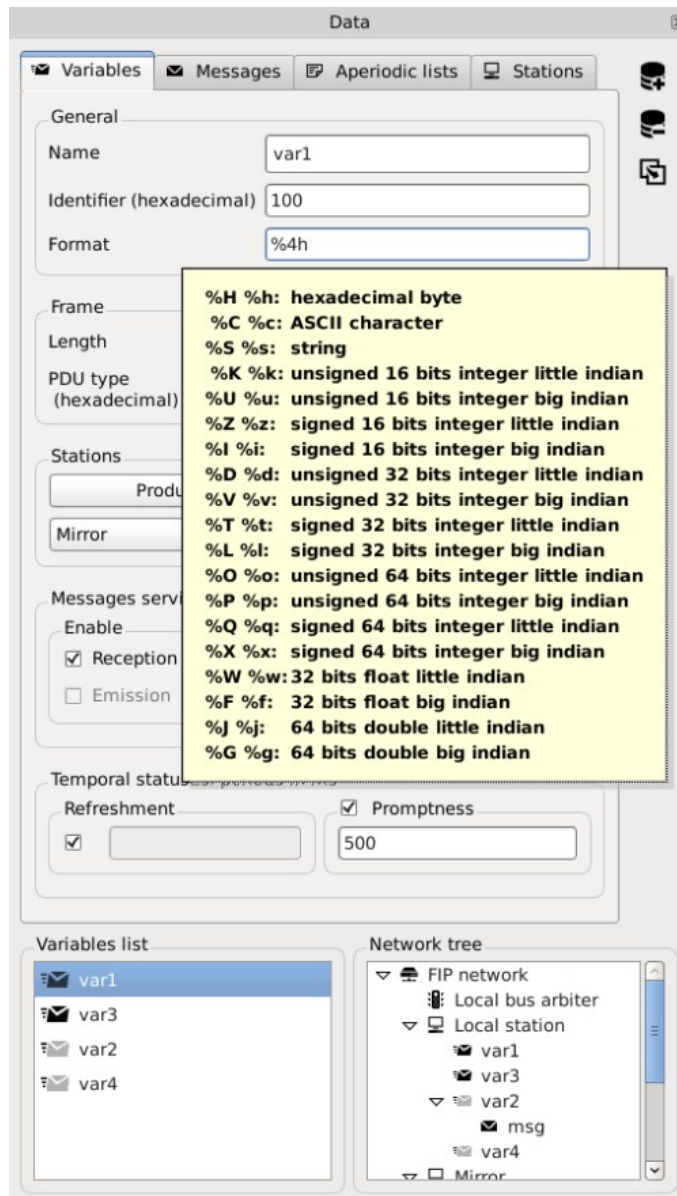


Figure 36: The Help format panel

4.3.4.EDITING AN APERIODICAL SEND LIST

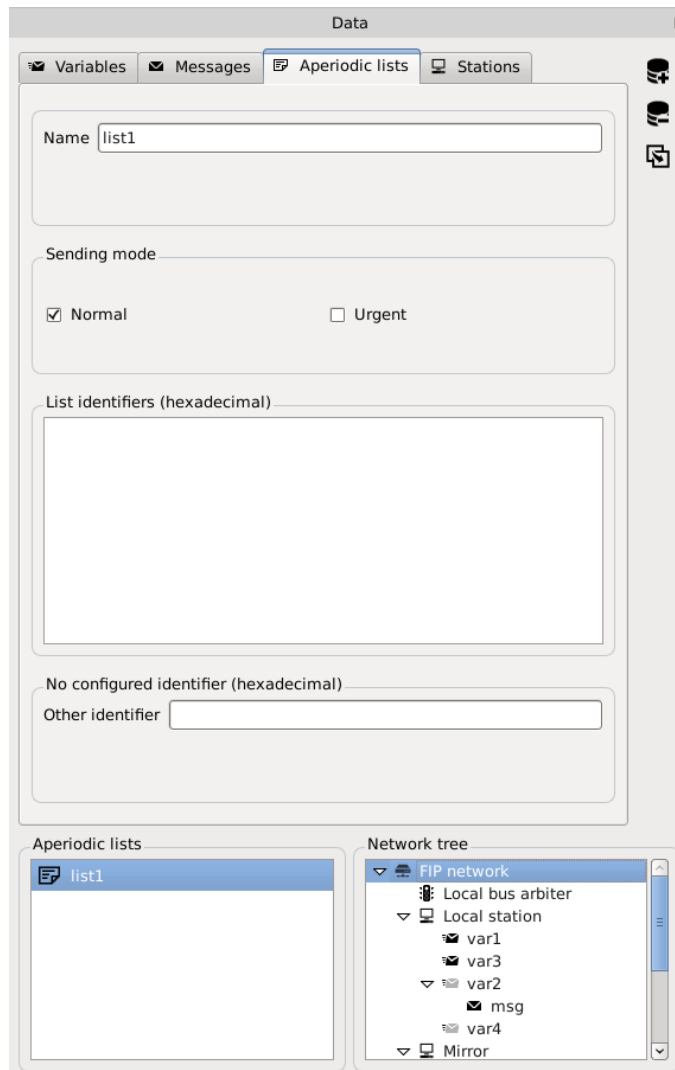


Figure 37: Panel for editing a list of variables to be sent aperiodically.

The list name is contained in the first edition field. It cannot exceed 32 characters for a "cnf 4" format, and 30 characters for a "cnf 3" format, and it must be unique.

The send mode can be normal or urgent.

To add a new identifier, just select it in the local network tree by opening the context menu on the variable or add it in the edit field "Other identifier".

If the identifier selected in this way is already on the list, it is not added.

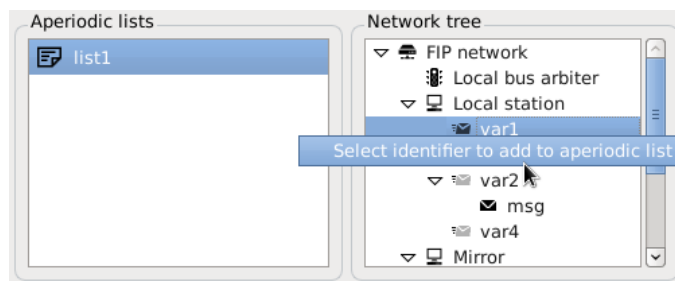


Figure 38: Adding an identifier to the list.

To delete an identifier, just select it in the list and open the context menu by clicking the right mouse button.

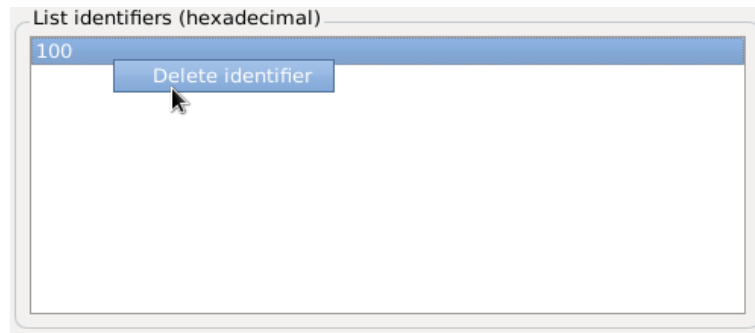


Figure 39: Deleting an identifier from the list.

4.3.5.EDITING AND CONFIGURING STATIONS

The user can view and configure stations in this panel:

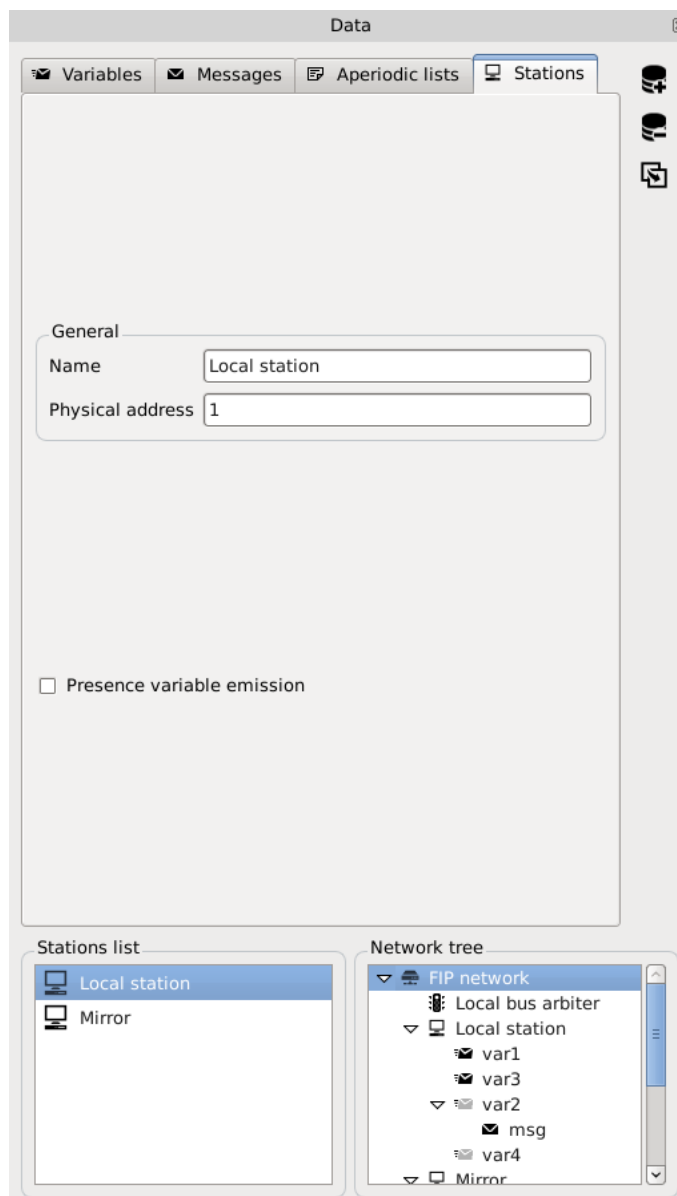


Figure 40: Panel displaying a station parameters.

A station does not contain relevant parameters to configure the PC/WorldFIP card. The station is only used for viewing the user application realistically on the tree. When the user creates his variables and local messages, he indicates the stations concerned by these local objects. The main parameter of a station is its name. This name should not exceed 32 characters for a "cnf 4" format and 30 characters for a "cnf 3" format.

Each station connected on WorldFIP uses a physical address which falls between 0 and 255, and must emit a presence variable. The high byte of the identifier is 14 hexadecimal and the lower part is the physical address. FipDesigner automatically generates a presence variable for a station if the user checks "Presence variable emission". The user will have set the correct physical address of the station in the corresponding display beforehand. Configuration of the variable generated is of course local. If it is the local station, the presence variable is configured as produced. If it is a presence variable of another station, it is configured as consumed or in other words, the local station consumes the other presence variables.

When a station parameters are set, the tree updates any modifications.

5.CONNECTION STAGES

The connection stages are composed of 2 visual elements that are automatically displayed when the user requests a connection and this succeeds.

This panel is mainly composed of 3 parts:

- FIP connection/disconnection button on vertical toolbar
- The “bus arbiter control” on horizontal toolbar.
- The “running panel” to access to configured FIP objects and medium control.

5.1.FIP CONNECTION/DISCONNECTION ACTIONS

5.1.1.CONNECT

This action is accessible through the vertical toolbar:



This command allows to connect to the WorldFIP network provided the settings are correct (i.e. have been validated by the automatic consistency check function). If the connection is done this button is replaced by the “disconnect button” one.

5.1.2.DISCONNECT

This action is accessible through the vertical toolbar:



This command allows to disconnect from the WorldFIP network, and closes the running panel. If the disconnection is done this button is replaced by the “connect button” one.

5.2.BUS ARBITER CONTROL TOOLBAR



Figure 41: Bus arbiter control toolbar.

This toolbar appears when the FIP connection is active.

The user can:

- Start the bus arbitrator, if it is stopped.
- Stop the bus arbitrator, if it is not stopped, nor starting up.
- Put the bus arbitrator back into an active state, if it has been suspended (“Continue” button).

- Switch from one sequence to another (if the bus arbitrator program does not originate from an ASCII file). Switching will take place at the end of the current sequence.

For further details on the bus arbitrator entity, refer to section 7.6 which deals with the principle of constructing a bus arbitrator macrocycle and section 4.2.1 which concerns the making of a bus arbitrator program.

5.3.RUNNING PANEL

This panel appears when the FIP connection is active. It can be shown or hidden using “Running” button on vertical toolbar.

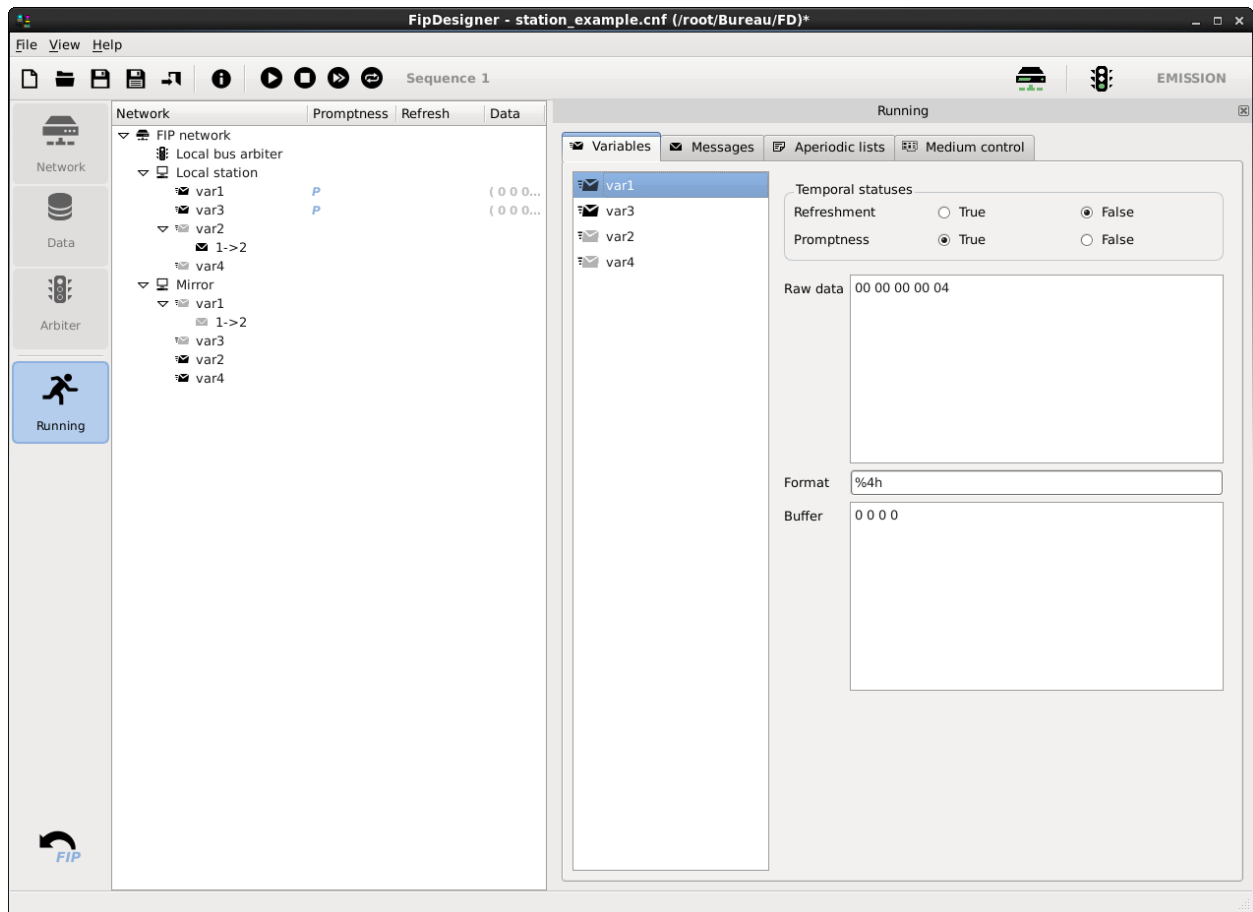


Figure 42: Running panel.

Tabs allow the user to perform his choices. With the panel, the user chooses what kind of configured object he wants to view (variables, messages, aperiodic lists). All objects of this category are displayed in a list box.

When one object is selected, the matching panel appears depending of the object status: received or sent by the local station.

The user can also choices medium control tab to manage it.

5.3.1.VARIABLE TAB

5.3.1.1.Sending a variable

If the user selects a produced variable in the list box, he has access to the panel “Sending a produced variable”.

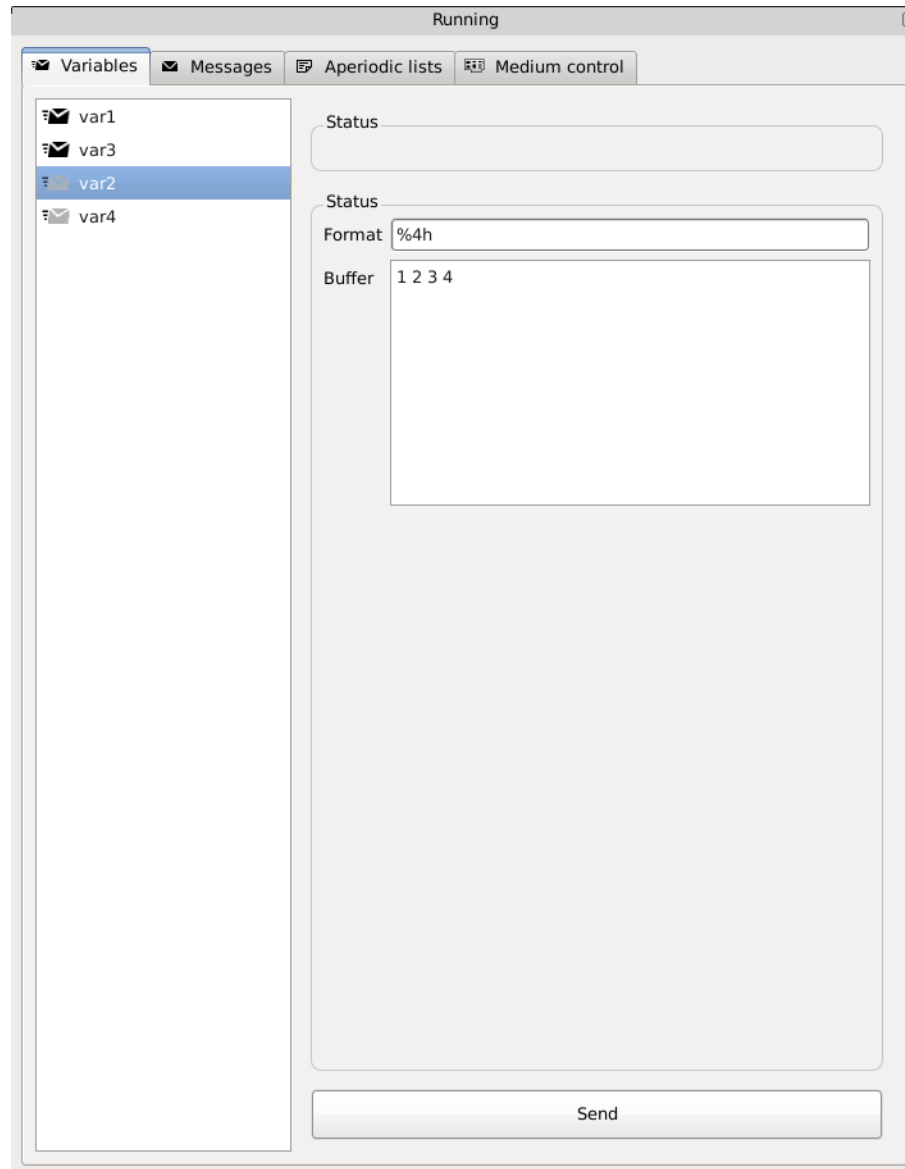


Figure 43: Sending a variable produced by the local station.

The "Status" field informs the user about the progress of the variable transmission (current stage, end of transmission, error...).

The "Format" field contains the string specifying the types used in the useful variable data (see section 4.3.3). When the variable is first selected, the indicated format is that of the variable configuration.

Pressing "Send" sends the contents of the variable to the local buffer while waiting for the next *ID_DAT* instruction associated with this variable.

5.3.1.2. Viewing a consumed variable

If the user selects a consumed variable, he has access at the panel "Viewing a consumed variable".

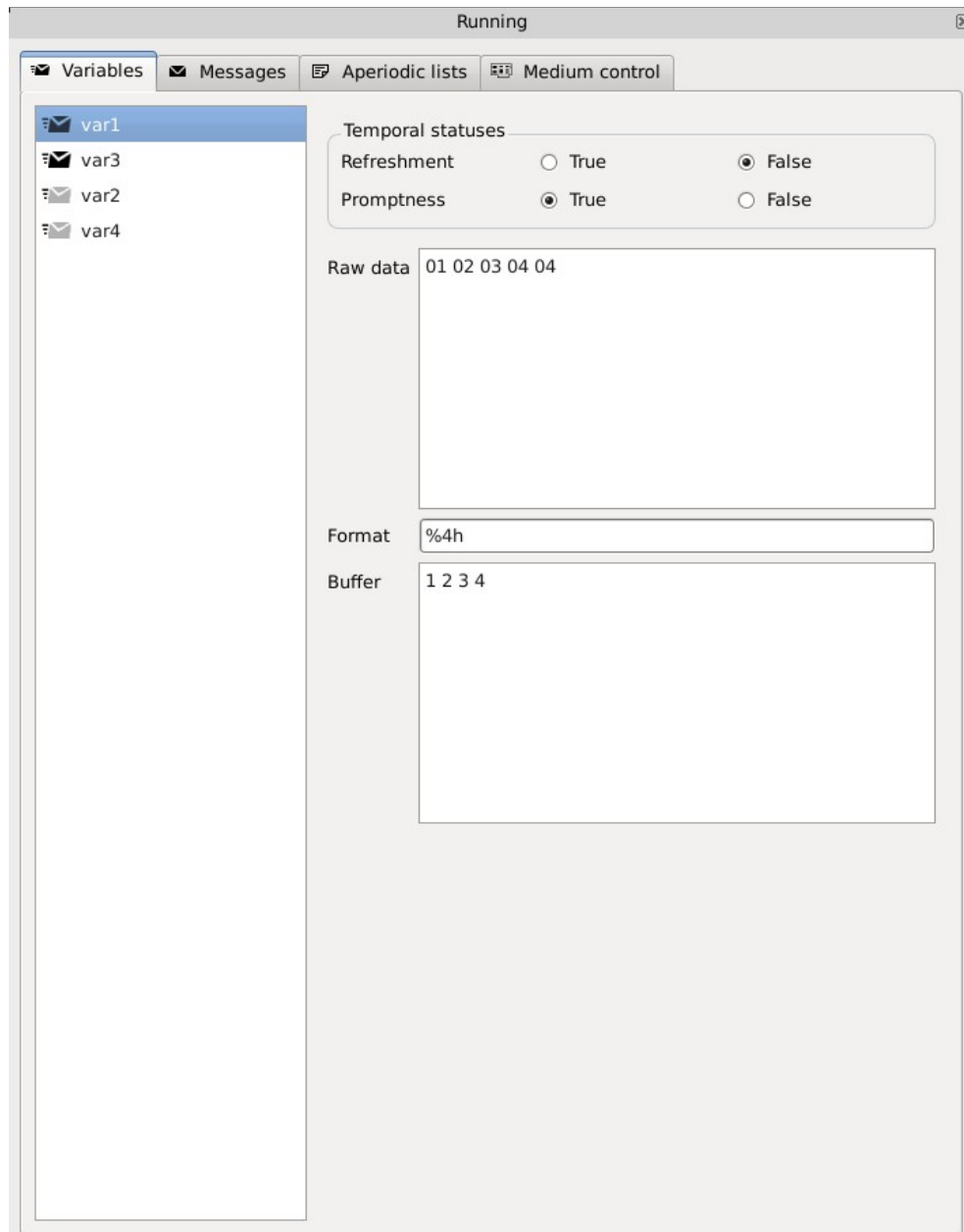


Figure 44: Viewing a variable consumed by the local station.

The temporal validation of the refreshment and promptness statuses is indicated if they are required in the configuration.

The "Format" field contains the string specifying the types used in the useful variable data (see section 4.3.3). As for a produced variable, when this variable is first selected, the indicated format is that of the variable configuration. The user can modify this field.

The "Raw data" field contains the raw hexadecimal bytes.

The "Buffer" field contains the variable of the received variable, interpreted according to the format entered.

The variable contents as well as the temporal statuses are regularly updated.

5.3.2.MESSAGE TAB

5.3.2.1.Sending a message

If the user selects a transmitted message in the list, he has access to the panel "Sending a message".

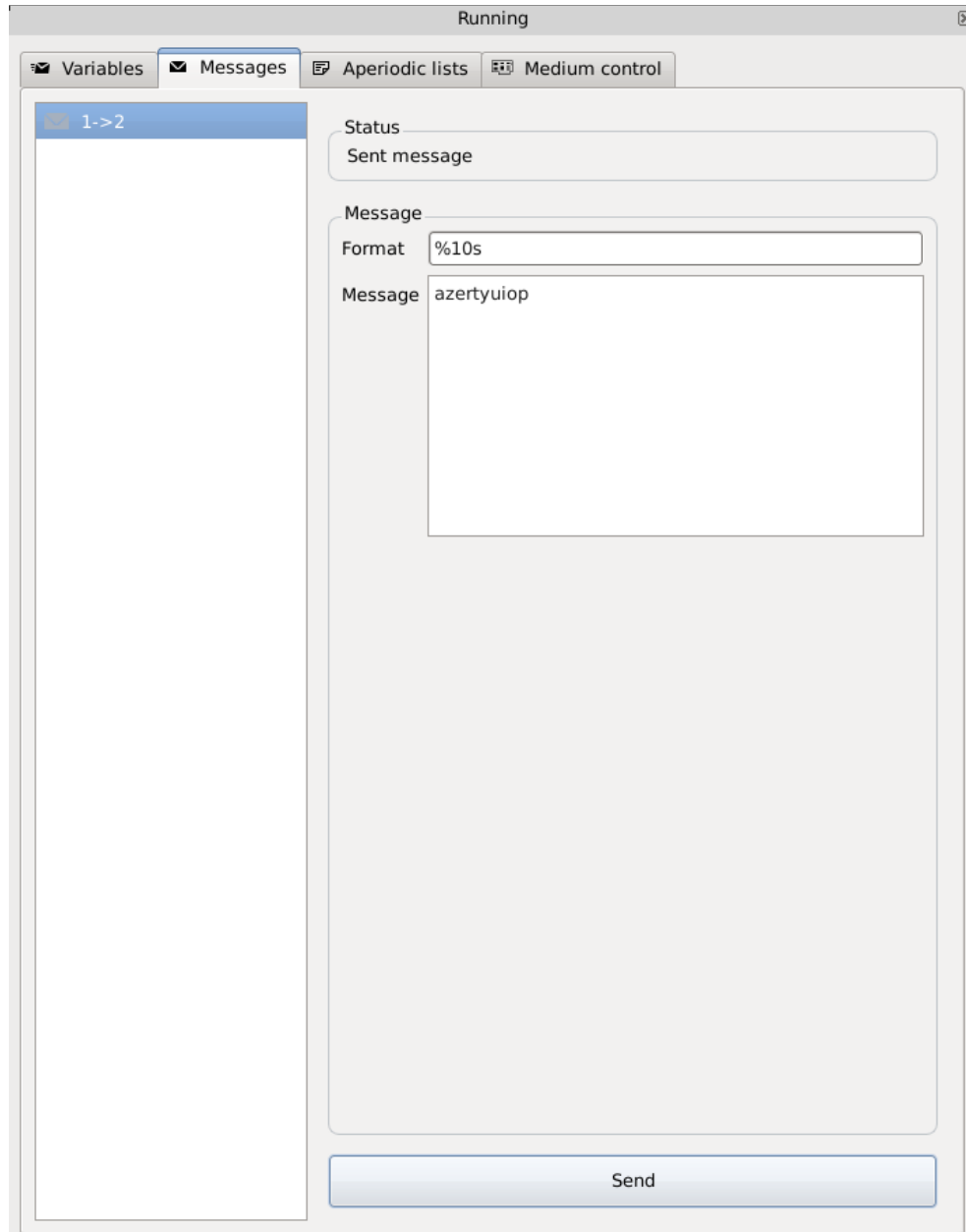


Figure 45: Sending a message from the local station.

The "Status" field informs the user about the progress of the message transmission (current stage, end of transmission, error...).

The "Format" field contains the string specifying the types used in the useful message data (see section 4.3.3). When the message is first selected, the indicated format is that of the message configuration.

Pressing "Send" sends the contents of the message to the local buffer while waiting for the next *ID_DAT* instruction associated with this message.

5.3.2.2. Viewing a received message

Each message received has its own reception stack where up to 64 "message" events can be stored.

.Configured message

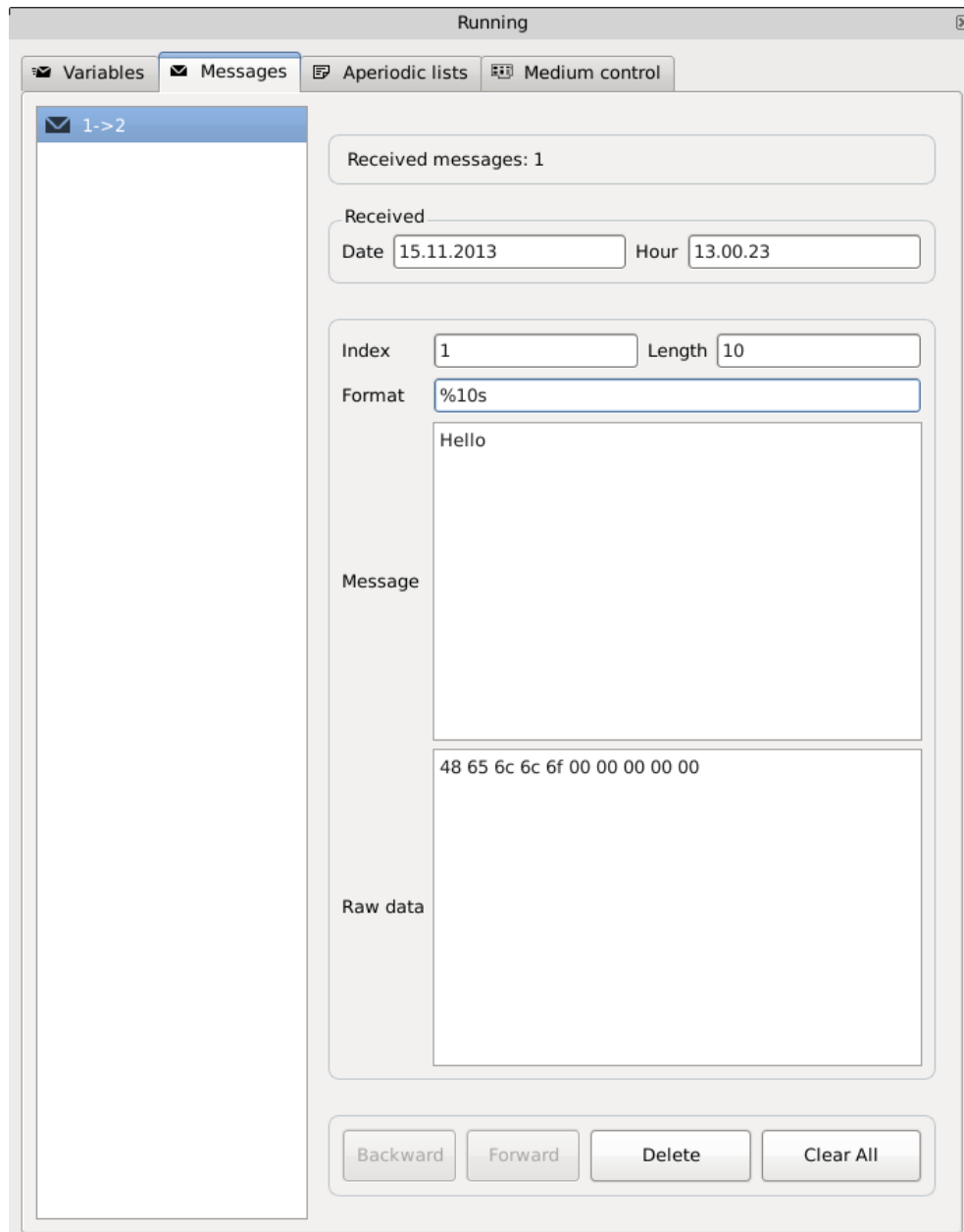


Figure 46: Viewing a configured received message.

.Event information

The "Index" field contains the number of the message event, starting from 1.

The "Messages number" indicates the total number this message has been received.

The length displayed is the length of the message contents in bytes.

The date is the date when the message was received referenced to the PC date, in the format day/month/year.

In the same way, the time when the message was received is displayed in the format hour/minute/second.

As for the variables, the "Format" field contains the string specifying the types used in the useful message data (see section 4.3.3). When the message is first selected, the indicated format is that of the message configuration.

The "Raw data" field contains the raw hexadecimal bytes.

The "Message" field displays the message content according to the format.

Once the user modifies the format, the interpretation of the contents is also modified.

The content of the message is interpreted according to the current format displayed.

.Stack actions

To go from one message to another, one must use the "Backward" and "Forward" buttons which allow to go to the previous message and the next message respectively.

The "Delete" button allows to delete the message currently displayed in the stack while the "Clear all" button will completely empty the pile.

.Non configured message

When a message is received a new icon is added to the messages list box, as long as the message is not configured. The message name is composed of the source identifier concatenated with the destination identifier and the prefix "vers".

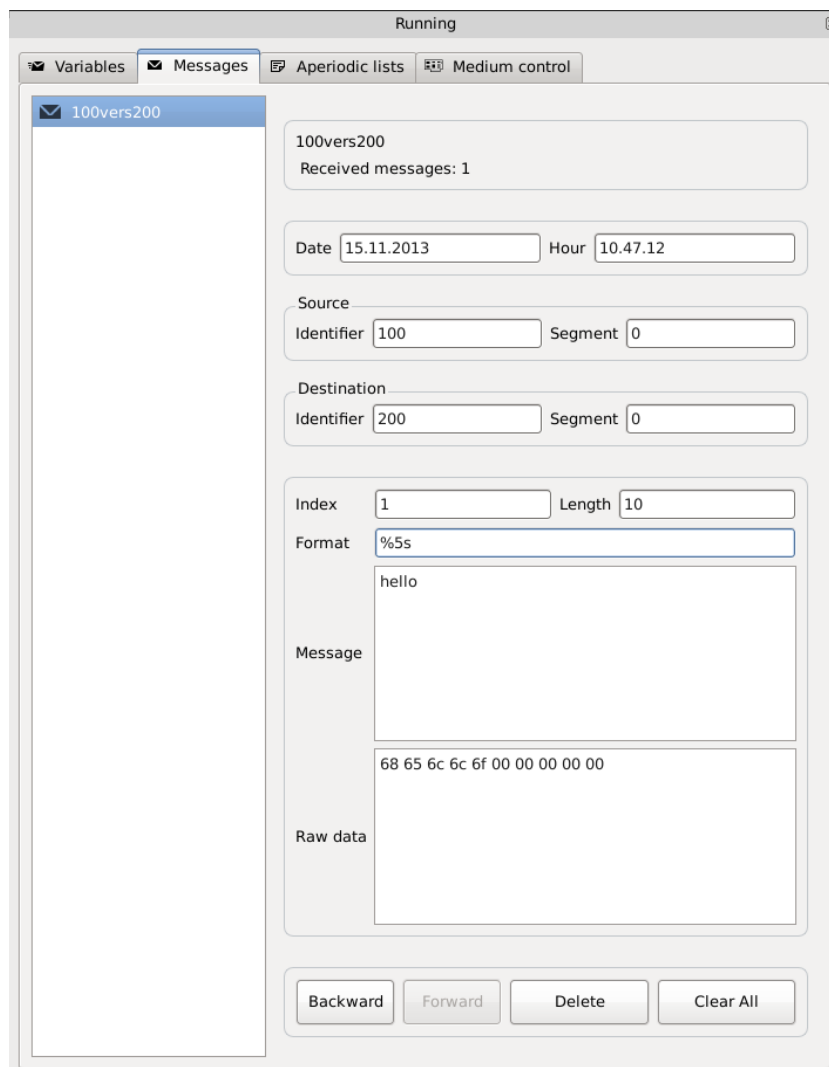


Figure 47: Viewing a received message while it is not configured.

The viewing panel contains 4 extra fields for the source and destination addresses in relation a configured message.

The source address of the message is made up of the segment and identifier numbers of the source variable of the message. The segment number on which the source

variable is found is indicated here, the WorldFIP network being broken down into segments of 32 stations maximum.

In the same way, the destination address of the message is made up of the segment and identifier numbers, of the message destination variable.

5.3.3.APERIODICAL VARIABLES TAB

The user can send aperiodic list by selecting one in the list. For this, he has just to press the button "Send".

If one error occurred, a message box will be displayed.

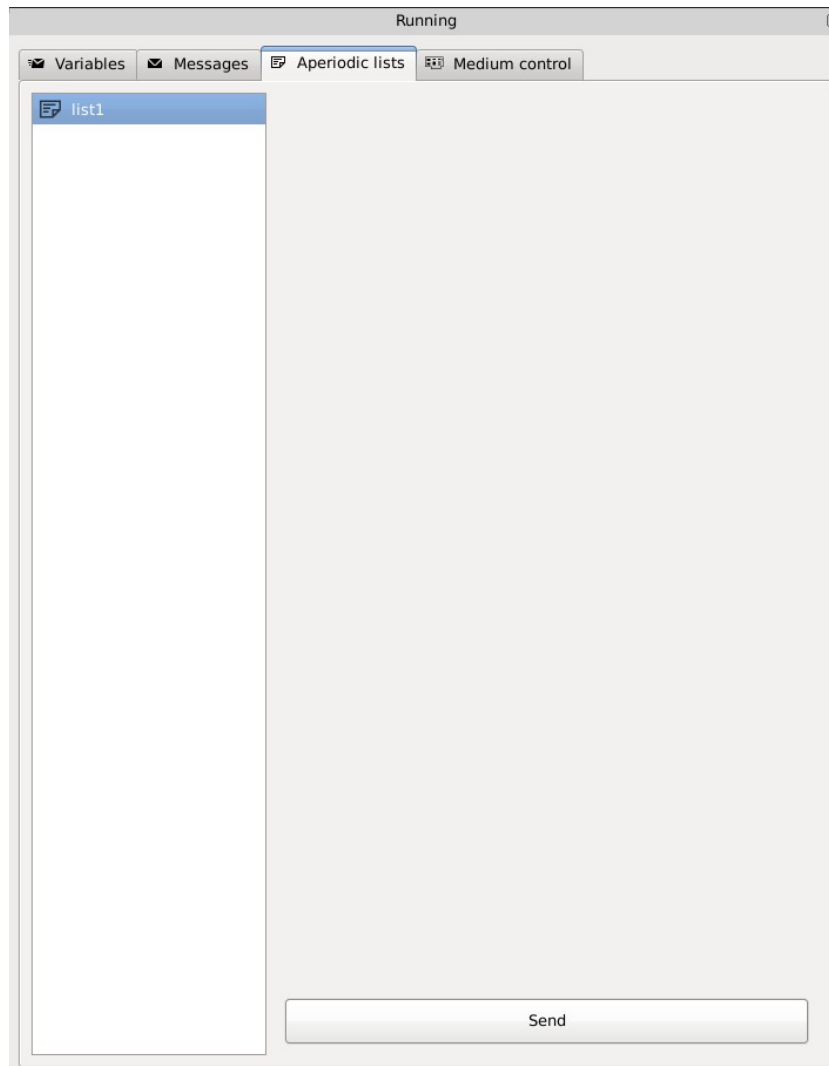


Figure 48: Sending an aperiodical configured list of variables.

Pressing "Send" sends the contents of the list to the local buffer, while waiting for the `SEND_APER` instruction.

5.3.4.MEDIUM CONTROL TAB



Figure 49: Communication medium management panel.

5.3.4.1. Overview of communication management

In order to have correct communication, the user must use a validated communication medium (bit ValX at 1) without a set error bit. Initialisation and validation of the mediums are made during the connection to WorldFIP. By default, the two mediums are validated.

It is advised to periodically supervise the communication state in order to ensure an error free medium. An erroneous line can be automatically invalidated to the benefit of the second (redundant medium).

When no line is validated, the test mode (local transmission loop on reception) is automatically selected. The user must therefore validate one of the mediums, or both of them in order to communicate and exit this mode. Connecting FipDesigner to the network carries out the following tasks:

- Reinitialisation of the line components (commands 2 and 3);
- Deletion of the transmission and reception errors (command 1);
- Validation of the two lines (command 6).

When the two mediums are validated, the frames are simultaneously emitted on the two mediums. The two lines are monitored continuously: the first which receives a

frame is connected to the card processor. If a frame appears simultaneously on both mediums, medium 1 is connected by default.

When a transmission error occurs, the incriminated line is automatically invalidated at the end of the current frame. Transmission on the line is not affected. If only one medium is validated, in case of transmission error, the active medium is not invalidated. The error bit EtX remains set as long as a command 1 is not applied.

When a watchdog error appears, the incriminated line is automatically interrupted. Validation of the line is not affected if the error disappears, but the error bit EwX will remain set while the line components are not reinitialised by commands 2 or 3, or by reconnection to the network.

Reception errors do not influence the validation of the concerned medium. The error bit ErX remains set as long as command 1 is not applied.

5.3.4.2. Communication state byte format

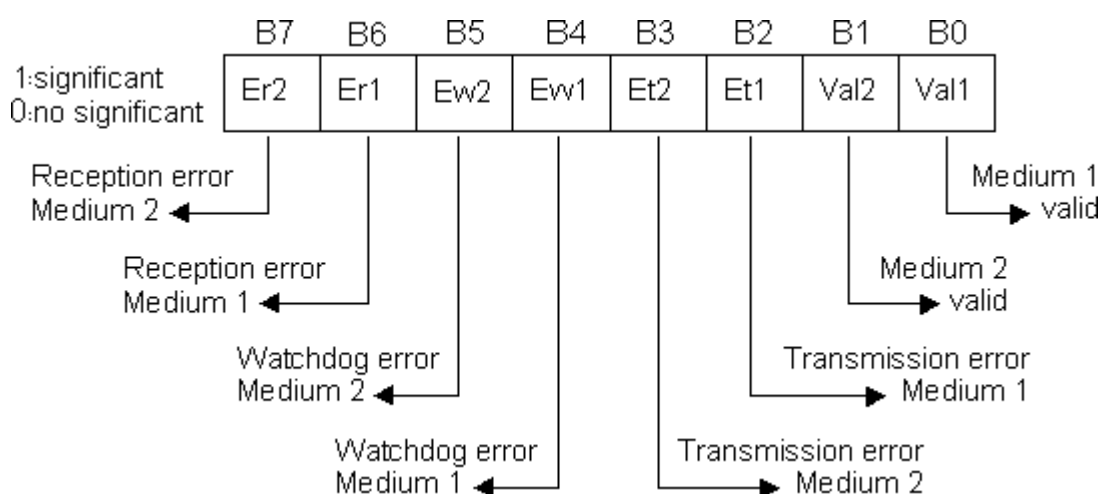


Figure 50: Format of the state byte.

The signification of the different bits is as follows:

- The ErX bits indicate reception errors. These may be a carrier loss of the received frame, a Manchester code error or a redundant periodical code error.
- The ErX bits indicate watchdog errors. This is a transmission error activated when the station no longer stops transmitting.
- The ErX bits indicate other transmission errors. These may be a bit which lasts too long, a transmission level which is too high or too low, or the absence of a cable.
- The ValX bits indicate if the X communication medium is validated, i.e., if the line components emit and receive frames.

5.3.4.3. Medium management commands

The commands available allow:

- 0: the communication medium state byte to be obtained
- 1: the deletion of communication errors on reception and transmission (only the bits ErX and EtX are concerned)
- 2: Reinitialisation (reset) of the line components of medium 1
- 3: Reinitialisation (reset) of the line components of medium 2
- 4: authorisation of medium 1, medium 2 is inhibited
- 5: authorisation of medium 2, medium 1 is inhibited

- 6: authorisation of mediums 1 and 2
- 7: execution of test mode (looping of transmission loop on reception)

6.EXAMPLES

The installation program copies two configuration examples with FipDesigner, STATION1.CNF and STATION2.CNF. These configurations correspond to 2 stations being able to:

- each produce and consume a variable;
- send and receive a message.

The first consumes the variable produced by the other and vice versa. The first receives the message sent by the other and vice versa.

7.ANNEX A: THE WORLDFIP BUS

7.1.PRINCIPLE

WorldFIP is a centralised access control network.

7.1.1.BUS ARBITRATOR

The entity charged with controlling access to the communication medium is called the *bus arbitrator* (or *BA*). There is only one active bus arbitrator at a given time. However, in case of malfunction, backup bus arbitrators may take over.

7.1.2.VARIABLES, VARIABLE IDENTIFIERS

On WorldFIP, it is the information to be circulated which is referenced, not the connected stations. The elementary packet of information is called a *variable*. Variables are identified by a 16 bit integer called the *identifier*.

7.1.3.POLLING TABLE

The bus arbitrator's *polling table* formed by the set of identifiers used on the WorldFIP network.

7.1.4.MESSAGING SERVICES

In addition to the variable exchange services, WorldFIP manages messaging services between a source-address and a destination-address. Optional reception acknowledgement is also supported.

7.2.CIRCULATION OF A VARIABLE ON THE WORLDFIP NETWORK

The circulation of a variable WorldFIP takes place in four stages:

7.2.1.MEDIUM ALLOCATION BY THE BUS ARBITRATOR

The bus arbitrator allocates the transmission medium for the current identifier in the polling table. It circulates the corresponding *ID_DAT* frame on the media (Figure 51).

7.2.2.ACTIVATION OF PARTICIPATING ENTITIES (PRODUCER, CONSUMERS)

The entities interested in the data recognise each other. There is only one emitting entity on the entire network by design. This entity is called the *producer* of the variable. The receiving entities are called *consumers* of the variable (Figure 52).

7.2.3.DIFFUSION (REFRESHMENT NOTION)

The producer circulates a *RP_DAT* response frame on the network. This frame includes the variable, that is the actual data (Figure 53).

It may also contain a time validation of the data. This validation is called *refreshment* on WorldFIP, it is computed by the producer of the variable. It measures how long the variable has been available to the network.

7.2.4.DATA ACQUISITION (PROMPTNESS NOTION)

When the consuming entities detect a *RP_DAT* frame, they acquire the data.

They may also compute reception time validation (Figure 54). This validation is called *promptness*, it measures how long the data available has been available to the consuming entity.

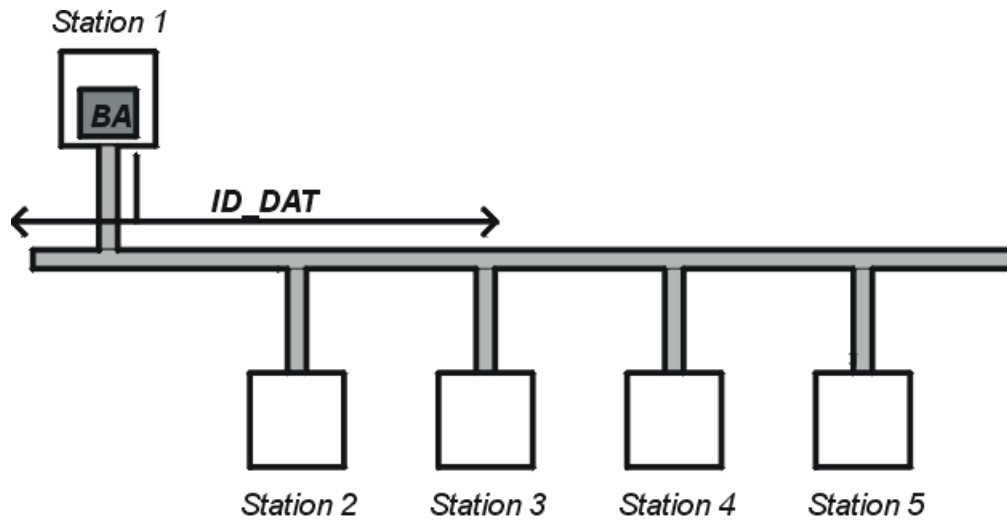


Figure 51: Circulation of an identifier on the bus arbitrator

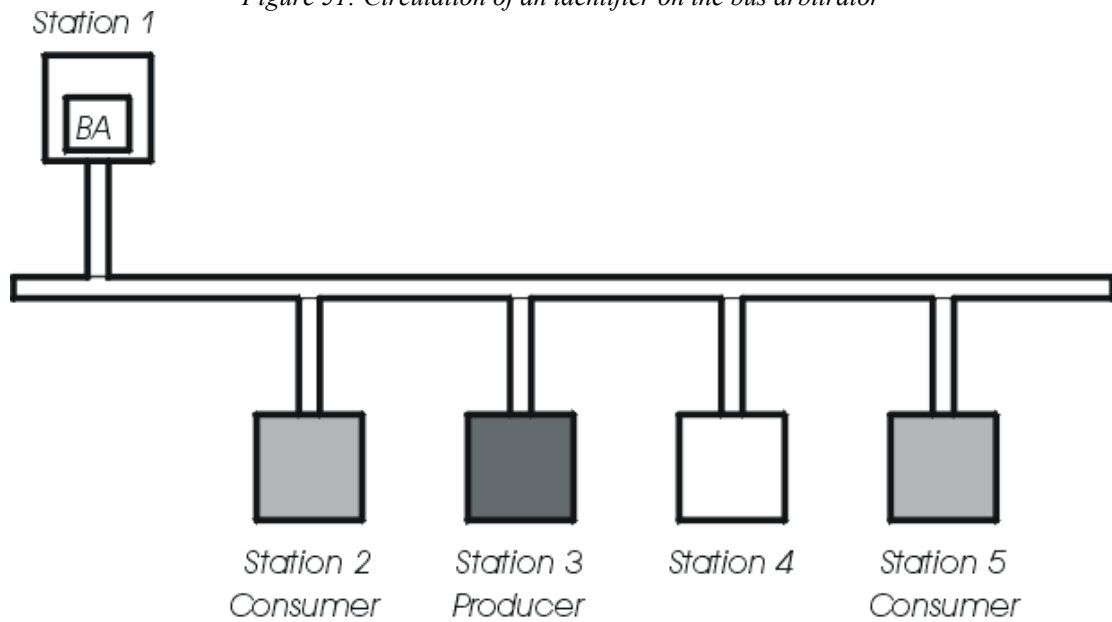


Figure 52: Recognition of the identifier by the interested entities.

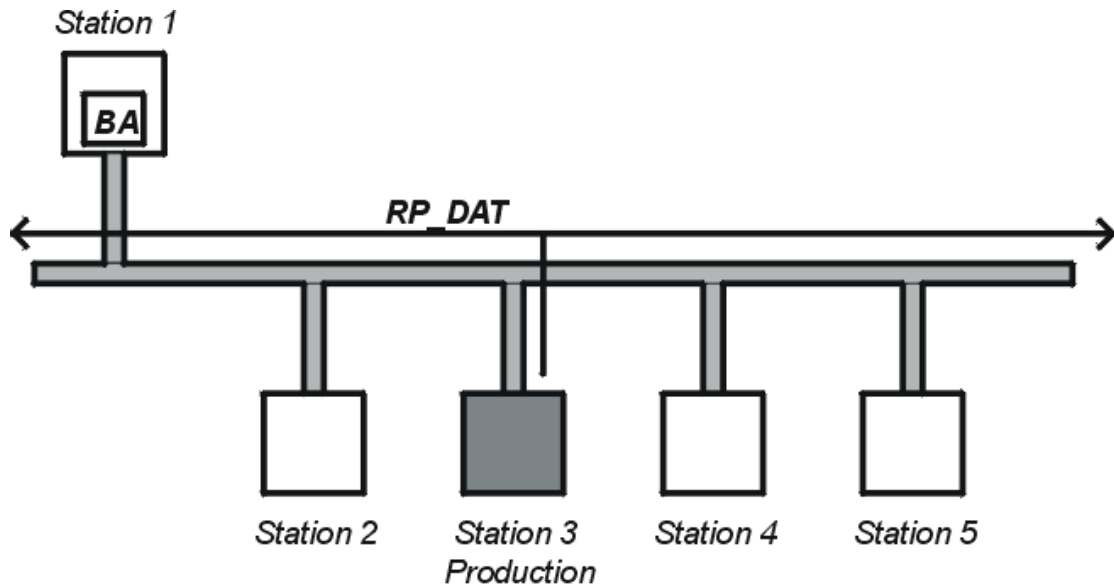


Figure 53: Circulation of the response frame by the producing station.

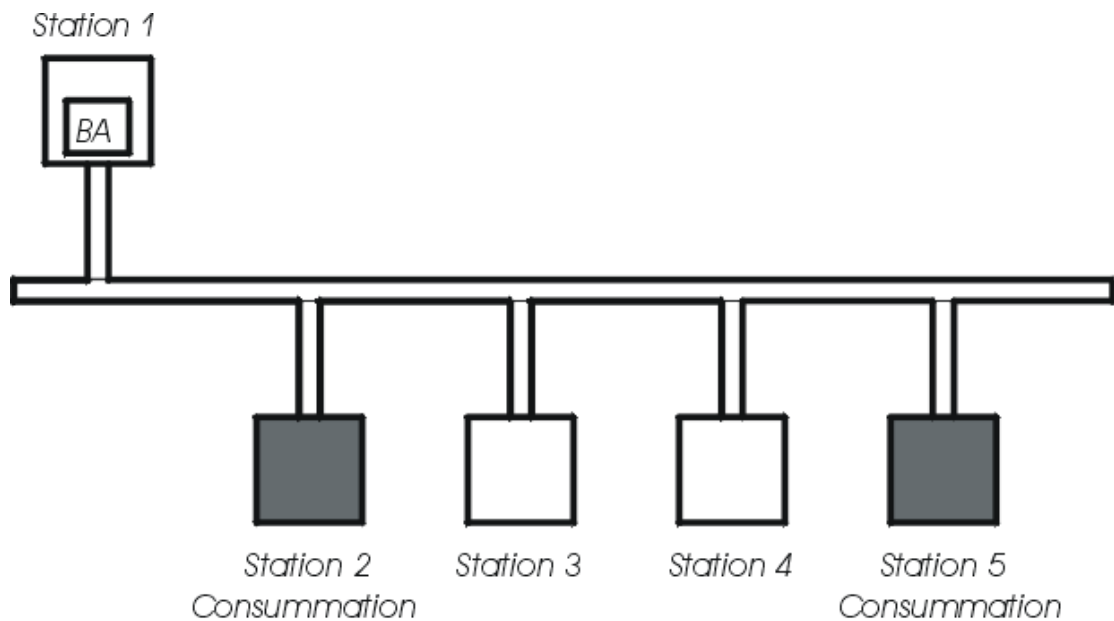


Figure 54: Consumption of the variable by the interested entities.

7.2.5.LOCAL BUFFER

Each device has a memory buffer where the data to be produced or consumed on the bus is stored. The local buffer of a device therefore contains the next value to be emitted in the case of a produced variable, or the last value received in the case of a consumed variable.

7.3.PHYSICAL LAYER

The physical layer ensures the transfer of information between the different devices connected to the bus.

7.3.1.TRANSMISSION MEDIUM

The transmission medium can be a shielded twisted pair or an optical fibre.

7.3.2.TRANSMISSION SPEED

The WorldFIP standard has defined three transmission speeds: 31,25kbit/s, 1Mbit/s and 2,5Mbit/s. The standard speed is 1Mbit/s.

7.3.3.TOPOLOGY

The WorldFIP topology is a bus topology to which the stations are connected by shunts.

7.3.4.BIT TIME T_{BIT}

It is the time corresponding to the emission of one bit:

$$T_{bit} = 32\mu s \text{ to } 31,25 \text{ kbit/s;}$$

$$T_{bit} = 1\mu s \text{ to } 1\text{Mbit/s;}$$

$$T_{bit} = 400ns \text{ to } 2,5\text{Mbit/s.}$$

7.3.5.BIT – LEVEL ENCODING

The physical layer codes the bits sent by the data link layer, using the Manchester code II. If T_{bit} is the interval of time allowing the coding of one bit, the principle of the Manchester code is to insert a transition in the middle of the T_{bit} interval. A rising edge codes a logical "0", a falling edge codes a logical "1".

Violations of this principle are also legitimate, and have special meaning.

The different cases are shown on Figure 55.

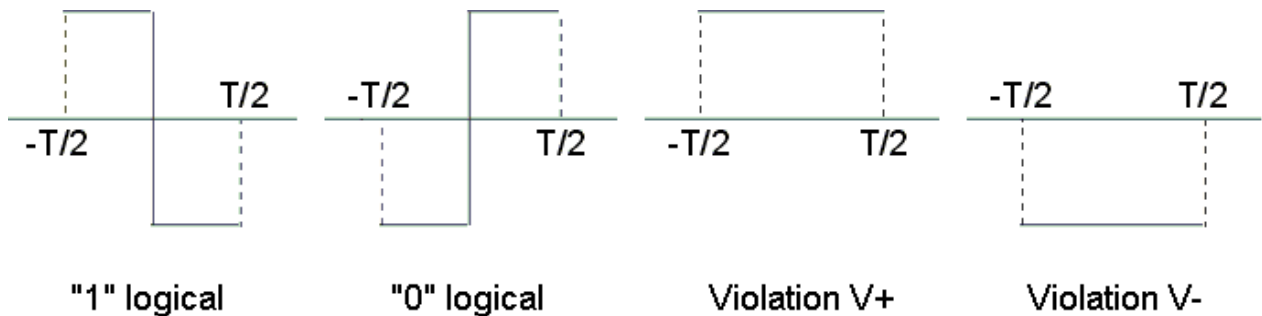


Figure 55: Bit encoding and violations.

7.3.6.COMPOSITION OF A WORLDVIP FRAME

7.3.6.1.Presentation

All WorldFIP frames are made up of three parts:

- 1- The start sequence;
- 2- The control and data field;
- 3- The end sequence.

The physical layer adds the start and end sequences. Upper layers are responsible for the control and data field.

7.3.6.2.Start sequence

The start sequence consists in:

- A prelude (PRE), which allows receivers to synchronise themselves with the emitter clock;

- A start delimiter (DDT), which indicates to the data link layer where it can find the beginning of useful information.

7.3.6.3. Control and data field

This field contains the actual data.

7.3.6.4. End sequence

It is a frame end delimiter, which indicates to the data link layer the end of the useful information.

The different sequences are shown in Figure 56 and Figure 57.

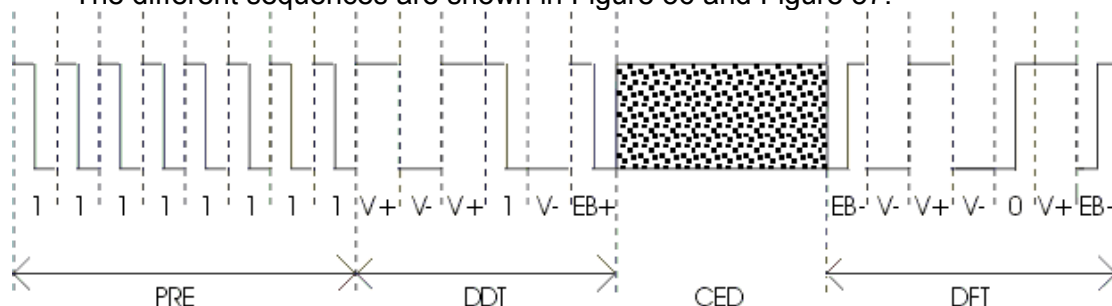


Figure 56: Sequences added by the physical layer to the FIP byte. These sequences are those specified by the French Standard NF C 46-604. In practice they are called FIP bytes.

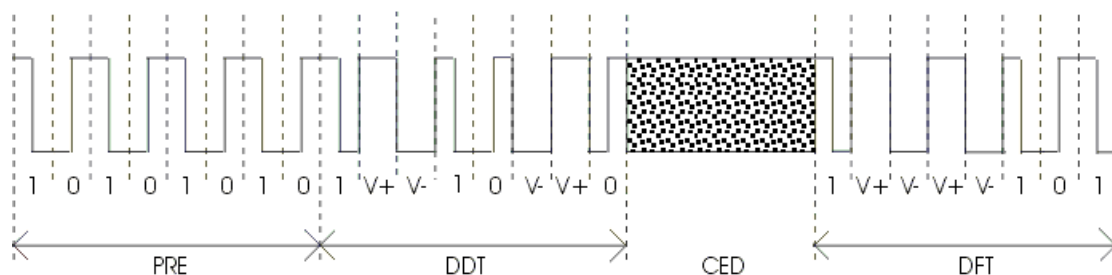


Figure 57: Sequences added to a FIP byte by the physical layer. These sequences are those specified by the European standard CENELEC EN61158-2. In practice they are called WorldFIP bytes.

Figure 57 shows the sequences as they are specified in the European standard CENELEC EN61158-2. More commonly this type of frame is called a WorldFIP frame.

7.3.6.5. WorldFIP and FIP frames

Devices constructed before the European standardisation (WorldFIP) can only support FIP frames.

This explains why certain applications still communicate with the old specification NF C 46-604.

Recent equipment can be configured for either FIP or WorldFIP frames.

Only one kind of frame can be used for a given user application.

7.4. DATALINK LAYER

7.4.1. GENERAL DESCRIPTION OF SERVICES

7.4.1.1. Transmission services

Two types of information transmission services are offered by the datalink layer:

- variable exchange
- message transfer.

7.4.1.2. *Transmission types*

Each transmission type exists in two kinds: periodical and aperiodical.

.Periodical transmission

Resources and time windows are fixed at system configuration time according to the user's application requirements.

The periodical transfer is triggered independently by the communication system.

.Aperiodical transmission

The information transfer takes place on an explicit user request.

7.4.1.3. *Addressing spaces*

The two types of transmission services yield two different addressing spaces in the WorldFIP system: one for variables, the other for messages.

.Variable addressing, identifier notion

For variable addressing, an *identifier* is associated with each system variable. This provides an unequivocal identification.

As a consequence, the entities taking part in the exchange of a variable are not identified explicitly but implicitly as *producer* or *consumer* of the identified variable.

.Message addressing, address notion

The transfer of messages takes place point to point, or in multipoint manner on the same segment.

For message addressing, one or several addresses which allow access to a message transfer service are defined in each network entity. These addresses identify an access point to a messaging service in a given user application.

During the transaction two addresses are used to relate the communicating entities. These are the source address and the destination address of the message. Both are 24 bit numbers to code the network segment number of the user application and its sub-address in the segment.

7.4.2.DATA FORMAT

7.4.2.1.Control and data field format

The data link layer information in a WorldFIP frame is the control and data field. It is made up as follows (see Figure 58):

- a control byte, which codes the type of frame (*ID_DAT*, *RP_DAT*, *ID_RQ1* or *2*, *RP_ACK*, etc.);
- data bytes, which may range from 0 to:
 - a maximum of 128 bytes for a variable response word,
 - a maximum of 256 bytes for a message response word;
- finally two bytes allowing a receiver to verify the integrity of the data (CRC, periodical redundancy code).

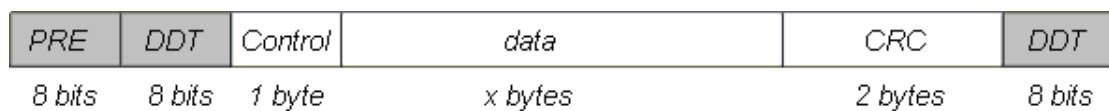


Figure 58:Link layer format of WorldFIP frames.

7.4.2.2.Medium allocating frames (*ID_XX*)

For the identifier frames allocating the medium, there are only two bytes of data. They specify the identifier of the variable for which the medium is allocated.

7.4.2.3.Response frames (*RP_XX*)

For the response frames, different cases exist:

- The variable response frames use a data field of 1 to 128 bytes. These are the data from the application layer.
- The response request frames use a data field made of a list of identifiers. 1 to 64 identifiers may be used and as a consequence the length of the data field varies from 2 to 128 bytes.
- The message response frames use a data field made up of:
 - 3 source address bytes,
 - 3 destination address bytes,
 - 1 to 250 bytes for data.
- Finally the acknowledge and end of transaction frames do not contain any data bytes. Their control byte codes either a positive receipt frame (*RP_ACK+*), a negative receipt frame (*RP_ACK-*) or a transaction end frame (*RP_FIN*).

7.4.3.DELAY DEFINITIONS

7.4.3.1.Overview

WorldFIP is a network with time constraints. The control of operation at datalink level is assured by delays. The definition of delays and their length are necessary for interoperability between connected equipment.

The common criterion for all stations to start delay countdown is the absence of activity on the network.

7.4.3.2.Silence time T_s

A reference delay T_o is defined, it is the *maximum allowable silence time on a segment of the network*. T_o (or *silence time*) is a global parameter of the system.

7.4.3.3.Turnaround time T_r

The reception of the last symbol of a frame is signalled to a station by an indication of silence from the physical layer. The reception or the emission by this same station of the first symbol of the following frame is signalled by an activity indication coming from the physical layer. The span of time between the two events defines the *turnaround time*, T_r .

In this way, the time between the *ID_DAT* frame and the next *RP_DAT* frame is T_r ; as it is between this *RP_DAT* frame and the next *ID_DAT* frame. The interval of time passed between a *ID_DAT* frame without frame response and the next *ID_DAT* frame however is the silence time, T_o .

7.4.4.THE BUS ARBITRATOR ENTITY

7.4.4.1.Overview

The medium allocation mechanism is totally centralised in an entity called the *bus arbitrator*.

Every station connected to the network can include a bus arbitrator function in addition to the usual data consuming/producing function.

However, a single station on the whole network effectively behaves as the bus arbitrator. The other stations featuring the bus arbitrator function are kept in reserve. They can become active bus arbitrators in case of malfunction.

7.4.4.2.Identifier frames (*ID_XX*)

The bus arbitrator is responsible for granting the right to talk to information producers, while taking into account the service requirements of the user application. This is done by allocating the medium with identifier frames.

Several types of identifier frames have been defined and a hierarchy exists in the order of emission of these identifiers. The greatest priority is to allocate the medium for periodical transfer of variables, requests or messages. The aperiodical transfer types are of a lower priority.

7.4.4.3.Elementary cycle

.Definition

The *elementary cycle* is defined as the polling by the bus arbitrator of:

- A set of non-exhaustive (from the user application point of view) identifiers, variables, requests and messages, all periodical. This set characterises the elementary cycle in question.
- A time window for the aperiodical exchange of variables.
- A time window for the aperiodical exchange of messages.
- A time window for synchronisation.

.Elementary cycle stages, comments

It follows that an elementary cycle is made of up to four stages.

- stage 1: periodic polling of variables, periodic triggering of variables, periodic triggering of messages;
- stage 2: triggered polling of messages;
- stage 3: triggered polling of variables;
- stage 4: wait until the end of the time window allocated for the elementary cycle.

An elementary cycle is made up of stage 1 at least.

Extra comments:

- The order of stages 2 and 3 is unimportant.
- During stage 4, a wait variable is emitted. This variable is associated to an identifier that is neither produced consumed. The corresponding frame is called a *padding frame*.
- The precision on the start of the next cycle is at best equal to the duration of an identifier frame followed by the delay T_0 .

7.4.4.4. Macrocycle

The *macrocycle* is defined as the juxtaposition of elementary cycles ensuring the exhaustive polling (from the user application point of view) of all the periodical identifiers.

The macrocycle thus covers all the periodical communication needs of the application. It also provides a window for triggered transactions.

A macrocycle is made up of at least one elementary cycle.

See paragraph 7.6.3 for further details on the construction of the macrocycle.

7.5.APPLICATION LAYER

7.5.1.DATA ENCODING AT APPLICATION LEVEL

7.5.1.1.Structure

The *data* field of a WorldFIP variable frame is shown in Figure 59.



Figure 59: Structure of the data field of a variable frame.

7.5.1.2.Identification byte

The identification byte code the PDU (Protocol Data Unit) type of the application layer, i.e. the nature of the data transported by the variable.

Generally the user application uses a 0x40 PDU type.

For the network management variables (presence or identification for example), the PDU type is equal to 0x50.

7.5.1.3.Refreshment

The WorldFIP application layer adds the value of the refreshment state in a byte at the end of the *data* field.

7.5.1.4.Content bytes

The standard limits the length of the field to 128 bytes. After coding of the application layer the user is left with 126 available bytes if there is no refreshment or 125 if there is one.

7.5.2.ASYNCHRONOUS PROMPTNESS AND REFRESHMENT.

These are the statuses which inform the consumers on the validity of the received variable.

7.5.2.1.Refreshment

The asynchronous refreshment status is generated by a variable producer. It is defined by a maximum refreshment duration T_r .

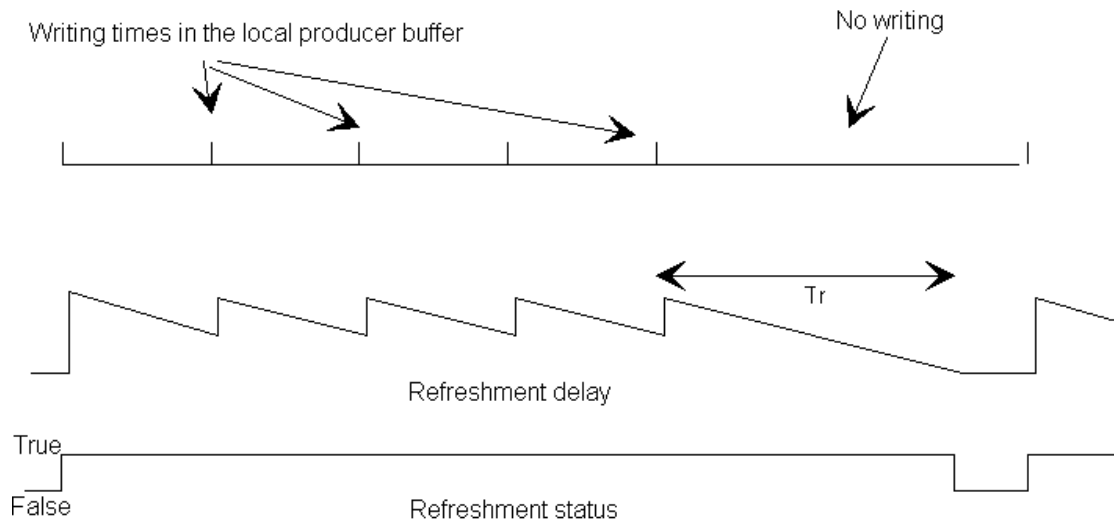


Figure 60: Asynchronous refreshment.

This status is located after the data in the *RP_DAT* frame.

7.5.2.2. Promptness

The asynchronous promptness status is generated by the variable consumer; it is a local status. It is defined by the maximum promptness duration T_p .

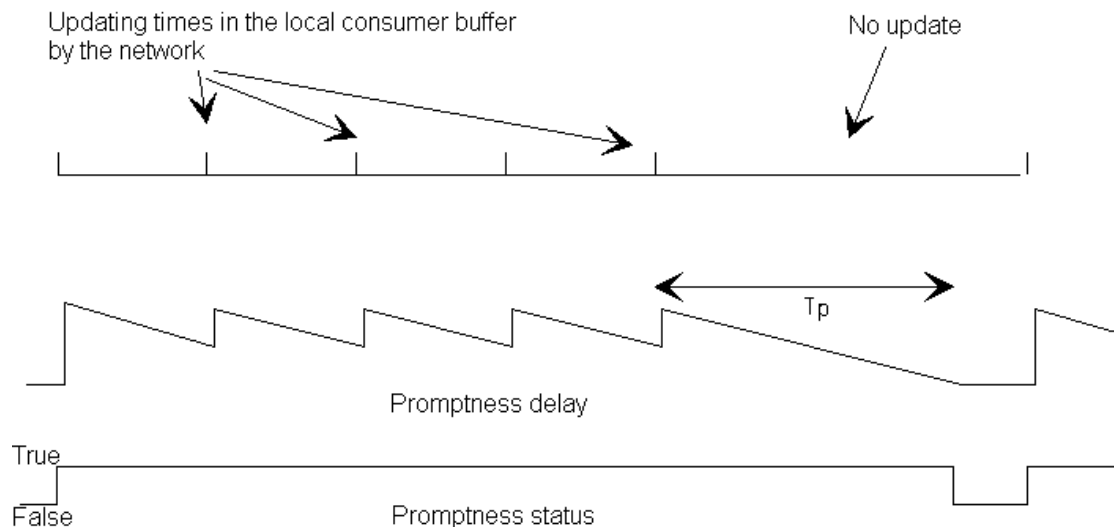


Figure 61: Asynchronous promptness.

7.6. CONSTRUCTION OF A BUS ARBITRATOR MACROCYCLE

7.6.1. DEFINITION OF THE MACROCYCLE AND THE ELEMENTARY CYCLE

It has already been mentioned that the bus arbitrator macrocycle is mainly built for periodical exchanges.

As a consequence, macrocycle construction begins with periodical exchanges analysis.

7.6.1.1. Macrocycle

Generally speaking, the user works with p variables. Each has a polling periodicity T_i , where $i \in [1, p]$.

Time needed to poll of a variable, or network occupation, is expressed by the following formula:

$$\tau = T_{ID_DAT} + T_r + T_{RP_DAT} + T_r,$$

where T_r is the turnaround time, T_{ID_DAT} and T_{RP_DAT} , the duration of the corresponding frames. Only T_{RP_DAT} depends on the variable length.

If we assume that all variables have the same length, τ is constant and allocation of the network by the bus arbitrator is cut into slices equal of time τ . This assumption does not change the principle of construction of the macrocycle presented in this section.

A common periodicity T_M is found for the p variables. It is the LCM¹ of the p periodicities. It can also be deduced that the polling of p variables by the bus arbitrator is summarised by the description of the buffer transfers during T_M . In fact, it is possible to prove that T_M the length of the *macrocycle*.

The description of these buffer transfers forms the polling table of the bus arbitrator.

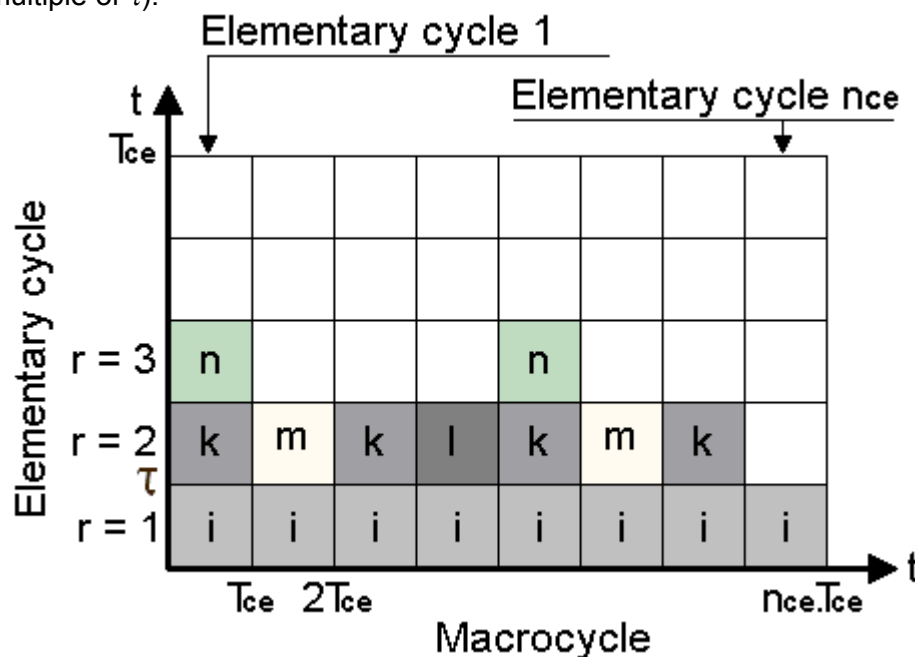
7.6.1.2. Elementary cycle

The LCM can be divided by the GCD² of the p periodicities. A whole number that will be called n_{ce} is found. This expresses an internal periodicity to the macrocycle that is called the *elementary cycle*. The macrocycle is a succession of elementary cycles and n_{ce} is the number of elementary cycles contained in the macrocycle. Each elementary cycle lasts T_{CE} , which is in fact the LCM of p periodicity.

7.6.2. REPRESENTATION OF THE MACROCYCLE

7.6.2.1. Overview

From the preceding conclusions, we can deduce a representation of the macrocycle in two dimensions (see Figure 14): the x-axis gives the macrocycle whose unit is the elementary cycle (multiple of T_{CE}); the y-axis is the elementary cycle whose unit is time (multiple of τ).



¹ Lowest common multiple

² Greatest Common Divider

Figure 62: Representation of the macrocycle in two dimensions constructed from the polling of variables i, k, l, m, n . Visualisation of the rows r of macrocycle. The white cases form the time left free by the periodical traffic.

7.6.2.2.Rank

The rank r of the macrocycle can be defined here as the set of polls of the r^{th} variable of the elementary cycles.

7.6.2.3.Maximum rank

This maximum is defined as the ratio T_{CE} / τ , that is the maximum number of pollings it is possible to make during T_{CE} (in the case where all variables are of the same length, as explained before).

7.6.3.CONSTRUCTION OF THE MACROCYCLE

The two dimensional representation of the macrocycle allows visualisation of the conditions to be respected so as to respect the periodicities of the variables:

- Condition n°1: the variable i can only be scanned once during the elementary cycle.
- Condition n°2: the polling of variable i always takes place at the same row r of the macrocycle.

7.6.4.TIME WINDOWS FOR APERIODICAL TRAFFIC

One is to understand that Figure 62 indicates the maximal time span for the macrocycle, $N_{CE} * T_{CE}$. Any longer time span would make it impossible to respect every variable periodicity. This is why the user cannot, in this representation, add a supplemental time window after $N_{CE} * T_{CE}$ to provide time for aperiodical exchanges.

This doesn't mean, though, that this construction of the macrocycle doesn't allow aperiodical exchanges.

In fact, time windows for aperiodical exchanges exist in every elementary cycle whose rank is lower than the maximum rank.

There are cases, however, where there is no time window for aperiodical exchanges. This happens when the respective periodicities of the variables make it so each elementary cycle has a rank equal to the maximum rank.

7.6.5.MACROCYCLE CONSTRAINTS

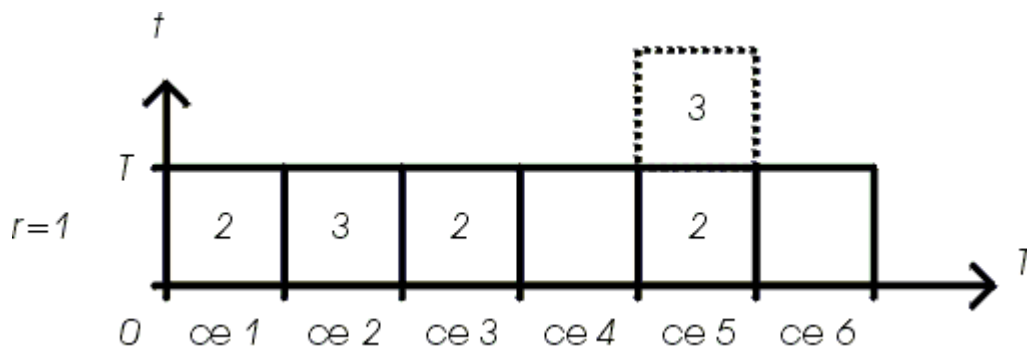


Figure 63: Example of a macrocycle that is not feasible.

It is sometimes not possible to build a macrocycle. Figure 63 shows a macrocycle with 2 variables of periodicity 2 and 3τ . The LCM is equal to 1τ and the GCD is equal to 6τ . Maximum rank is 1. When trying to construct the macrocycle, one sees that two variable pollings are necessary in the same elementary cycle, which is impossible as the corresponding rank would be 2.

It can be shown that when the periodicities of the variables are integers among themselves, polling them requires different rows. By construction the row number is limited in the macrocycle. For easier construction, it is better to take multiple periodicity's between them (for example 10τ , 20τ , 40τ).