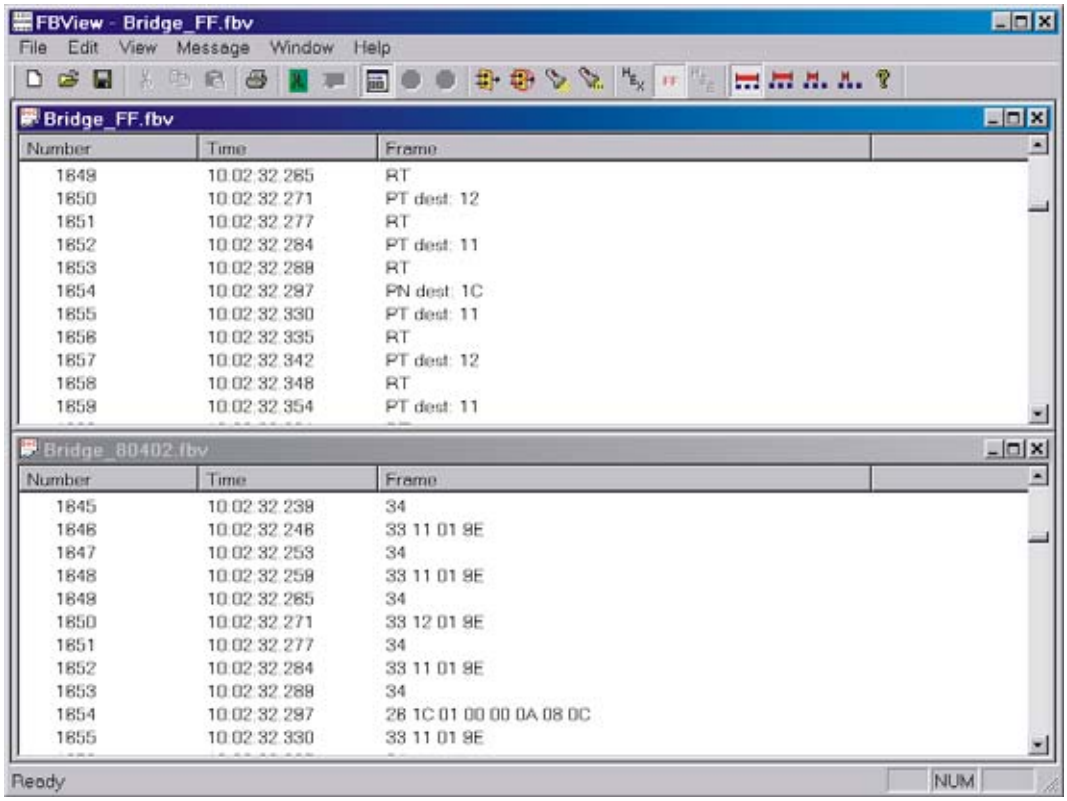
FBVIEW

smar

FIRST IN FIELDBUS

USER'S MANUAL

FIELDBUS NETWORK ANALYZER



JAN / 05

FBVIEW

Version 4.1

FOUNDATION™

FBVIEWME

smar

web: www.smar.com

Specifications and information are subject to change without notice.
For the latest updates, please visit the SMAR website above.

BRAZIL
Smar Equipamentos Ind. Ltda.
Rua Dr. Antonio Furlan Jr., 1028
Sertãozinho SP 14170-480
Tel.: +55 16 3946-3510
Fax: +55 16 3946-3554
e-mail: smarinfo@smar.com

ARGENTINA
Smar Argentina
Soldado de La Independencia, 1259
(1429) Capital Federal – Argentina
Telefax: 00 (5411) 4776 -1300 / 3131
e-mail: smarinfo@smarperifericos.com

CHINA
Smar China Corp.
3 Baishiqiao Road, Suite 30233
Beijing 100873, P.R.C.
Tel.: +86 10 6849-8643
Fax: +86-10-6894-0898
e-mail: info@smar.com.cn

FRANCE
Smar France S. A. R. L.
42, rue du Pavé des Gardes
F-92370 Chaville
Tel.: +33 1 41 15-0220
Fax: +33 1 41 15-0219
e-mail: smar.am@wanadoo.fr

GERMANY
Smar GmbH
Rheingaustrasse 9
55545 Bad Kreuznach
Germany
Tel: + 49 671-794680
Fax: + 49 671-7946829
e-mail: infoservice@smar.de

MEXICO
Smar México
Cerro de las Campanas #3 desp 119
Col. San Andrés Atenco
Tlalnepantla Edo. Del Méx - C.P. 54040
Tel.: +53 78 46 00 al 02
Fax: +53 78 46 03
e-mail: ventas@smar.com

SINGAPORE
Smar Singapore Pte. Ltd.
315 Outram Road
#06-07, Tan Boon Liat Building
Singapore 169074
Tel.: +65 6324-0182
Fax: +65 6324-0183
e-mail: info@smar.com.sg

USA
Smar International Corporation
6001 Stonington Street, Suite 100
Houston, TX 77040
Tel.: +1 713 849-2021
Fax: +1 713 849-2022
e-mail: sales@smar.com

Smar Laboratories Corporation
10960 Millridge North, Suite 107
Houston, TX 77070
Tel.: +1 281 807-1501
Fax: +1 281 807-1506
e-mail: smarlabs@swbell.net

Smar Research Corporation
4250 Veterans Memorial Hwy.
Suite 156
Holbrook , NY 11741
Tel: +1-631-737-3111
Fax: +1-631-737-3892
e-mail: sales@smarresearch.com

INTRODUCTION

FBView is a powerful tool for those who want to work with fieldbus devices.

FBView provides the user with the messages that pass through the fieldbus buses. **FBView** can capture, analyze and decode the messages, showing all information. The information reports the message type, the address of the device that sent the message, and the address of the receiver. It is also possible to decode the messages from each level of the *Fieldbus* protocol.

FBView is easy to use and runs on Windows 2000 and XP.

FBView captures the messages from the bus and sends it to the computer using Smar's *DFI302*.

Characteristics

- Captures messages from any fieldbus bus (ISP, FF, HSE and Profibus PA).
- The captured messages can be sent to the printer and/or saved to files.
- Decodes FF messages.
  - Recognizes the message type.
  - Identifies the addresses.
  - Identifies the data.
  - Splits the messages from each level from the protocol.
- The messages can also be displayed in hexadecimal format.
- Messages filters. It means that it captures only the desired messages.
- Search tools. This tool is useful to find a pattern in the captured messages.
- Time Measure.
  - Evaluates the final time of each message. This time can be absolute or relative to a specific message.
  - Evaluates the time interval between the ends of each consecutive message.
  - Evaluates the time interval that the bus had no messages (Idle Time).
- Counts the number of captured messages.
- Verifies the FCS of the messages to check that the message was received with no errors and point out the messages with errors.
- Calculates the percentage of invalid messages.

Table of Contents

1. INSTALLATION

1.1 System Requirements

1.1.1 Minimum

Operational System	→	Windows 2000 - Service Pack 3
Processor	→	Pentium 333 MHz
RAM	→	64 MB
HD Free Space	→	10 MB
Monitor	→	1024x768 - 64 Kcolors
CD-ROM		

1.1.2 Recommended

Operational System	→	Windows 2000 - Service Pack 3
Processor	→	Pentium 1 GHz
RAM	→	128 MB
HD Free Space	→	10 MB
Monitor	→	1280x1024 - True Color
CD-ROM		

1.2 Installing FBView

Place *SYSTEM302* CD installation at the CD-ROM driver. The *Installation* dialog box will open automatically. Click the *SYSTEM302* button.

Follow the instructions in the dialog boxes to complete the installation. **FBView** and other programs that compound *System302* will be installed.

To initialize **FBView**, click the Windows *Start* button, at the *Task Bar*, point the cursor to the item *Programs*. Select the item *System302* and click the *FBView* icon. See the following figure:

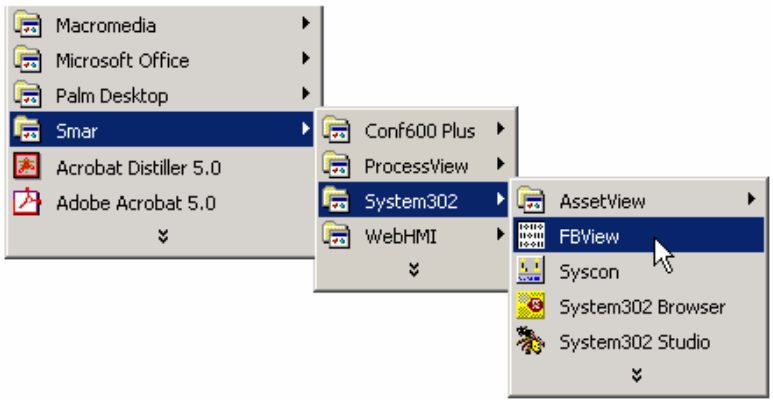


Figure 1.1. Running FBView

The **FBView** window will open:

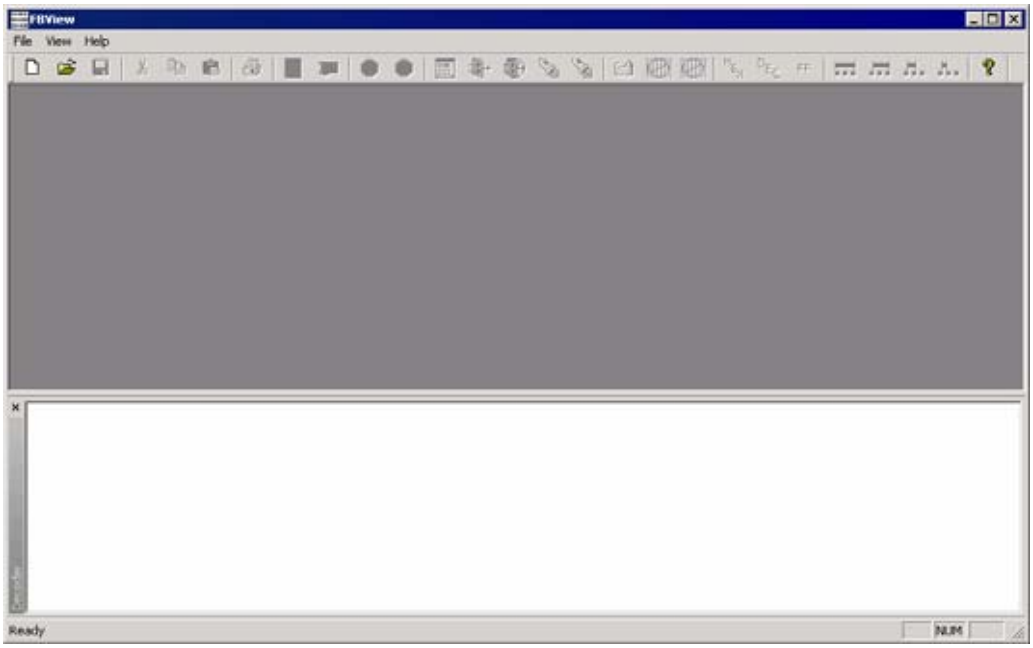


Figure 1.2. FBView Screen

2. OPERATION

FBView is easy to operate. It can be operated using the keyboard or selecting an icon on the toolbar or a menu option. The captured messages are stored in a buffer, named *Reception Buffer*. This buffer is fulfilled when **FBView** is on *Capture* mode. This buffer can also be loaded from a file. The messages stored in this buffer are displayed on the screen.

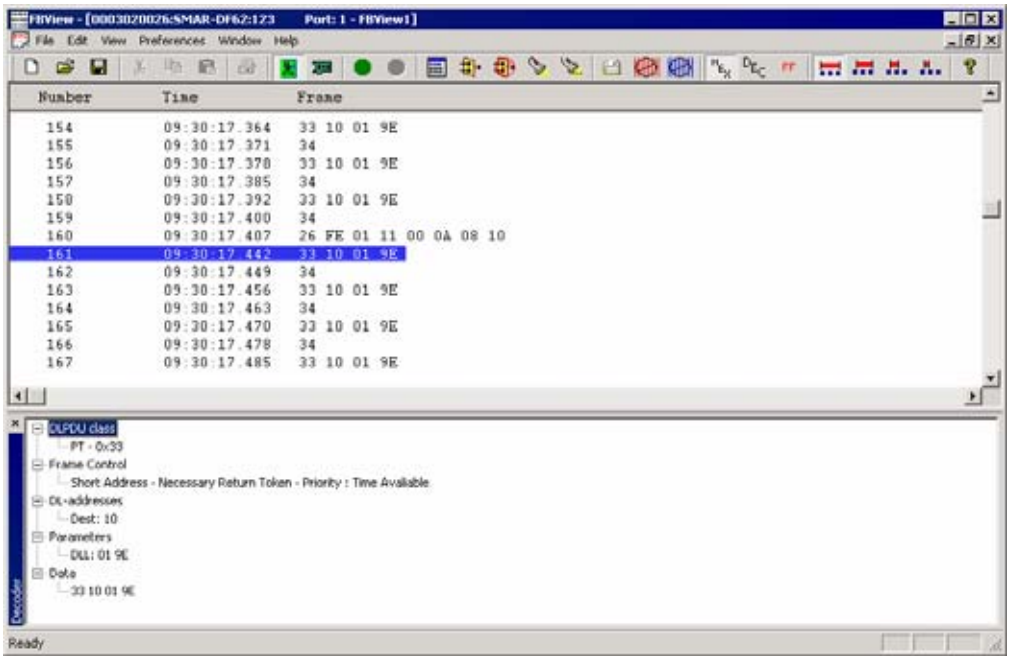


Figure 2.1. FBView Screen

For some tasks, the user has to select a message in this buffer. For doing so, select the message with the mouse. The selected message is displayed in blue.

If the message is too long and it is not completely displayed, it can be read rolling the screen horizontally. Use the “left” and “right” arrows, or use the horizontal bar to roll the screen.

2.1 Toolbar

New

Creates a new file with an empty buffer. After clicking this button, the user must select the communication network: H1 or HSE. Refer to section ***"Selecting the Communication"***.

Open

Click this button to fill the *Reception Buffer* with messages previously saved in a file.

Save

The user can save the reception buffer to a file, to be used later. Click this button and a dialog box will open. Select the folder where the file will be saved and type the name for the new file. The files will be saved with the extension **.fbv* for the H1 network and **.ntv* for the HSE network.

Print

Send the messages stored in the *Reception Buffer* to the printer. Uses the *Windows* default printer dialog box.

Initialize Communication

Start ou stop the communication with the interface. When clicking this button to initialize the communication, a dialog box will open to the user to select the interface.

Interface

Click this button to configure the interface that will capture the messages, according to the communication network selected.

Start

Starts to fill the buffer with the messages captured in the bus. The captured messages are temporarily stored in the *Capture Buffer*. The messages will be captured until the user selects *Stop Capture* or until the *Capture Buffer* is completely full. The messages will be stored in the buffer until a new capture is initialized. During the capture, the messages can be displayed on the screen, or the *View* mode can be selected to display the statistic of the captured frames.

Stop

Stops the fulfillment of the *Reception Buffer*. When the capture is suspended, the user can verify the contents of the messages that were stored in the buffer. It is also possible to use or configure other software functions, such as filters and search tools.

Enable/Disable Frame Statistics

Change the visualization of the messages in the buffer. The user can view the frame statistics or the description of the messages.

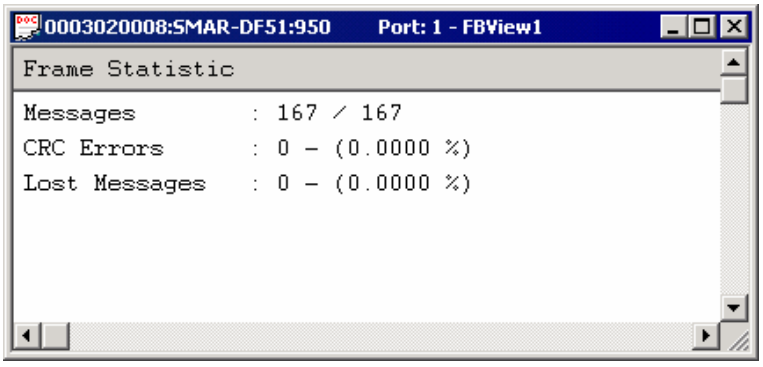


Figure 2.2. Statistics View

In the *Statistics View*, there are two counters for the item ***Messages***. The first counter counts the number of messages that were captured after they passed through the filter. The second counter displays the number of total messages that passed through the bus. In this example, no filter was applied, therefore the number of filtered messages is equal to the number of messages sent through the bus.

The ***CRC Errors*** counter displays the amount of messages with errors after they passed through the filter.

The ***Lost Messages*** counter displays the amount of messages that were lost. That is, the number of messages that were sent to **FBView** by the interface but **FBView** did not received. If the computer or the interface is too slow this counter increases excessively.

Enable Filters

When this function is active, only the filtered messages are stored in the buffer. An important function of the filter is the ability to capture a specific frame and trigger the frame counting according to the user specifications. Once the frame counting reaches the specified number the program stops capturing all frames. This function is named *Filter Trigger* and can only be activated if the user is online.

Another filter functionality is the *Filter Profile*. With this filter, the user can save all filters configured with specific names related to their function. The user creates a library with configured filters, being able to add/remove one or all filters from the library.

Refer to section ***"Filters"*** in the ***FBView-H1*** and ***FBView-HSE*** modes.

Disable Filter

This tool is used to disable all filters and restore all frames that were captured, if the frame analysis has been executed offline. If the user selects a frame after it has passed through the filter and then click this button, all filters will be disabled and the message will still be selected, displaying what happens after the message.

Find

Click this button to search for a pattern in the buffer. The user can search for a default byte or a error message, specified by the user.

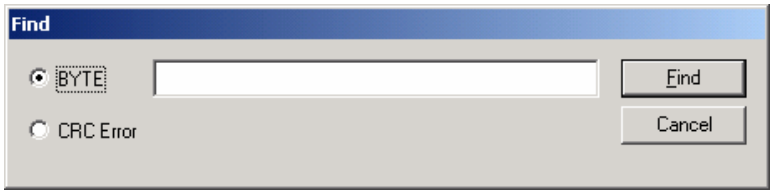


Figure 2.3. Find Dialog Box

To search for the first occurrence of a default byte, select the item **BYTE**, type the byte to be located and click **Find**.

To search for the first occurrence of an error message, select the item **CRC Error** and click **Find**.

Find Next

Click this button to repeat the last search, starting from the point where the last search stopped. Or press *F3* on the keyboard.

Enable Schedule for Captured Frames

Click this button to define a cyclic time interval to store the captured messages. That is, if the user configures a 2-hour interval, for example, **FBView** will store in the *Capture Buffer* the messages captured in the last two hours, from the moment the user clicks the *Stop* button, overwriting old messages stored in the *Buffer*.

Click this button to open the configuration dialog box:

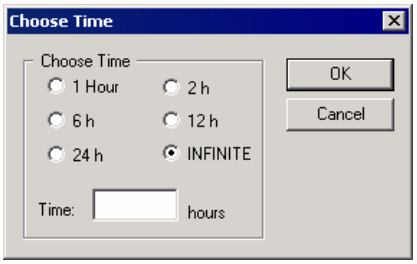


Figure 2.4. Defining the Schedule for Captured Messages

Select one of the pre-defined intervals or type the value in the *Time* box, and click *Ok* to conclude.

NOTE
It is necessary to configure this tool before start capturing frames by clicking the *Start* button.

Disable Red Grid Lines

The vertical grid line is a graphical tool used to locate bytes in a message.

Click this button to remove all grid lines displaying in the message buffer.

See section *2.4 Grid Lines and Interval Lines*.

Disable Blue Interval Lines

Interval lines are automatically drawn according to the interval defined by the user.

Click this button to remove all interval lines displaying in the message buffer.

See section *2.4 Grid Lines and Interval Lines*.

Hexadecimal

Displays the message in the format that it was received, in hexadecimal. In this case, no de-codification is made.

Decimal

Displays the message in the format that it was received, in decimal. In this case, no de-codification is made.

Fieldbus Foundation

Decodes and displays the messages according to the **Fieldbus Foundation** protocol. If the message does not suit the protocol, it will be displayed in hexadecimal format.

Absolute Time

Indicates the *Absolute Time* of the end of the message. The absolute time is measured according to the computer clock. When this button is active, the *Time* column indicates the absolute time for all stored messages.

Relative Time

Indicates the *Relative Time* of the end of the message. The relative time is calculated from the absolute time of each message related to the initial point, that is displayed as 0 (zero) in the *Time* column. When this button is active, the *Time* column displays the relative time for all messages and the message that is currently selected is considered the initial point (*Relative Time* equals to 0).

Message Time

Measures the period of time between two consecutive message endings. When this button is active, the *Time* column displays the elapsed time from the end of one message until the end of the following message.

Idle Time

Measures the period of time that the bus has no activity between two consecutive messages. When this button is active, the *Time* column displays the *Idle Time* between two messages.

About

Open the dialog box with the information about the **FBView** version.

2.2 Preferences

2.2.1 Hiding Messages

When the interval to capture message is too long, the user can temporarily hide the messages being captured.

In the *Preferences* menu, select *Hide Frames*.

To display the messages being captured, go to the *Preferences* menu and select *Show Frames*.

2.2.2 Customizing the Buffer Window

The user can configure the *Reception Buffer*, changing colors. In the *Preferences* menu, click *Change Colors*. The dialog box to configure the colors will open:

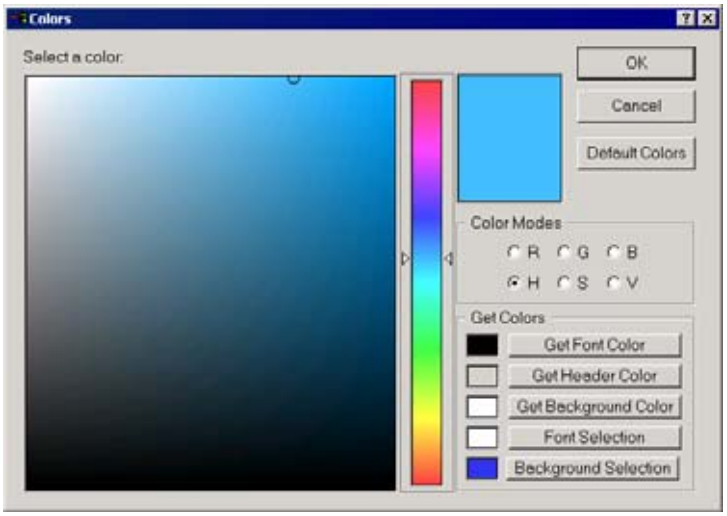


Figure 2.5. Customizing the Buffer Window

Get Font Color	Change the color of the text font of unselected messages.
Get Header Color	Change the fill color of the header in the buffer window.
Get Background Color	Change the background color of the buffer window.
Font Selection	Change the color of the text font of the selected message.
Background Selection	Change the fill color of the line for the selected message.

To change the color:

1. in the *Color Mode* field, select the mode to display colors.
2. in the color scale, click the desired color.
3. Click the button related to the color that will be changed. (See the table above).
4. Click *Ok* to apply the changes or click *Cancel* to exit without altering the colors.

To return to the default color configuration, click the button *Default Color* in the Color dialog box.

2.3 Organizing Buffer Windows

Several buffer windows can be displayed in **FBView**.

To organize and arrange the windows, go to the *Window* menu and select:

Cascade: Arrange all non-minimised windows in cascade.

Tile Horizontally: arrange all non-minimised windows side by side horizontally.

Tile Vertically: arrange all non-minimised windows side by side vertically.

2.4 Grid Lines and Interval Lines

To add a grid line to the buffer, right-click the buffer screen in the desired location. To remove only one specific grid line, right-click the line in the buffer screen.

To remove all grid lines displaying in the message buffer, click the button  , in the toolbar.

To add interval lines to the buffer, press and hold the <*Ctrl*> key on the keyboard and right-click the buffer screen in the desired location. Define the period for the interval and press <*Enter*> to conclude:

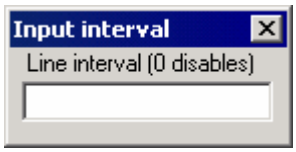


Figure 2.6. Defining the Grid Interval

To remove all interval lines displaying in the message buffer, click the button  , in the toolbar.

2.5 Debugging

To create a message sequence to debug error messages, for example, follow the steps described below:

1. Select a message, clicking the corresponding message line.
2. Press <*Ctrl*> + *F2* on the keyboard.
3. Repeat these steps to select other messages.

To unmark a message, click the message line to select the message and press <*Ctrl*> + *F2* again.

To browse and debug the selected messages, press *F2*.

2.6 Selecting the Communication

To create a message file, go to the *File* menu and click the option *New*. The dialog box showed in the figure below will open:

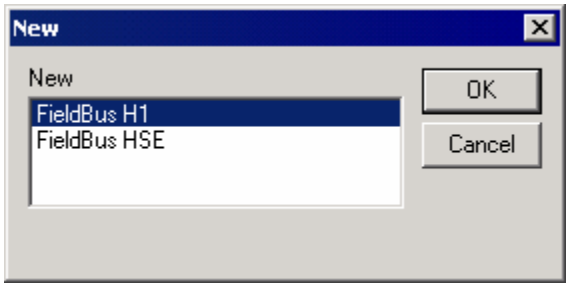


Figure 2.7. Selecting the Communication

In this dialog box, the user must select the type of the communication network where the messages will be captured. For each type, H1 or HSE, different interfaces and filters will be available.

When selecting the H1 communication, the Ethernet and USB interfaces will be available, as showed in the figure below:

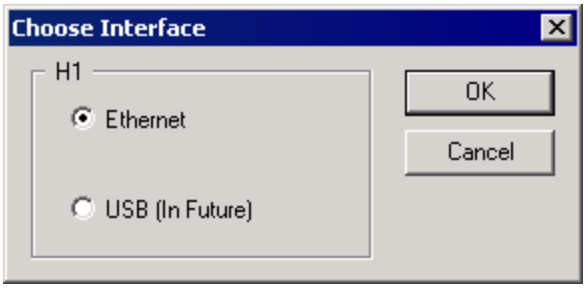


Figure 2.8. H1 Communication Interfaces

Refer to section "***Selecting the Communication Interface***" in the ***FBView-H1*** mode for further information.

When selecting the HSE communication, if the PC has more than one network adapter, click the button **Interface**,  , to open the dialog box and select the adapter that will be used to capture the frames.

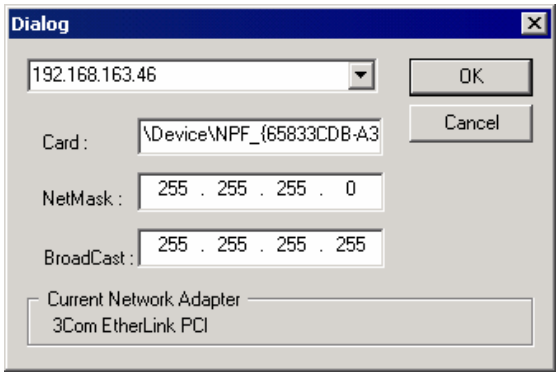


Figure 2.9. HSE Communication Interfaces

Refer to section "***Selecting the Communication Interface***" in the ***FBView-HSE*** mode for further information.

2.7 Defining the Diagnostic Structure

FBView uses a *XML* file to define the structure and the interpret the diagnostic messages. This file is named "*diagnostic.xml*" and it is located in the **FBView** installation folder. The default path is "C:\Program Files\Smar\FBView".

The diagnostic file "*diagnostic.xml*" can be edited in *Windows Notepad* or any HTML and XML editor.

For example, to decode the message *Compel Data 2* showed below:

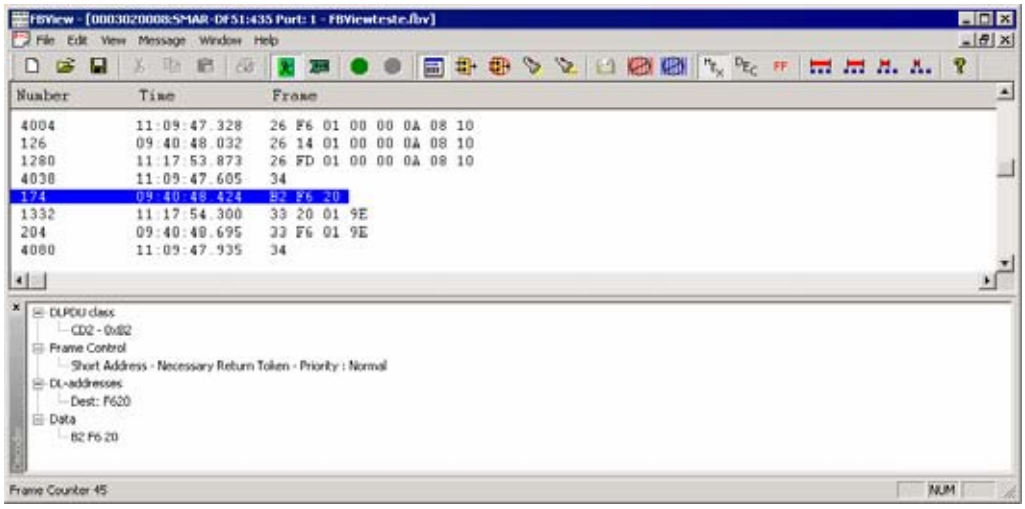


Figure 2.10. Message Compel Data 2

Add the following structure to the XML file:

```
<files_structures>
    <structure name='Compel Data 2' length='3'>
       <Uchar>
           <Name>Message Type</Name>
           <Format>0x%x</Format>
           <Length>1</Length>
           <Opt value='81'>COMPEL DATA 2</Opt>
       </Uchar>
       <Uint>
           <Name>Destination Address</Name>
           <Format>%x</Format>
           <Length>1</Length>
       </Uint>
    </structure>
</files_structures>
```

The structure <*Uchar*> has 1 byte, <*Uint*> has 2 bytes and <*Ulong*> has 4 bytes.

The field <*Format*> indicates the format of the byte value, according to the ANSII C standard: *%d* for decimal, *%l* for long int, *%x* for hexadecimal and *%c* for character.

The field <*Length*> always has the value equals to 1.

The value of the field <*Opt value=''*> must always be in decimal.

After editing and saving the file "*diagnostic.xml*", go to **FBView**, click the message to be interpreted, select the parameter *Data* in the *Decode* window and double right-click the message to open the menu:

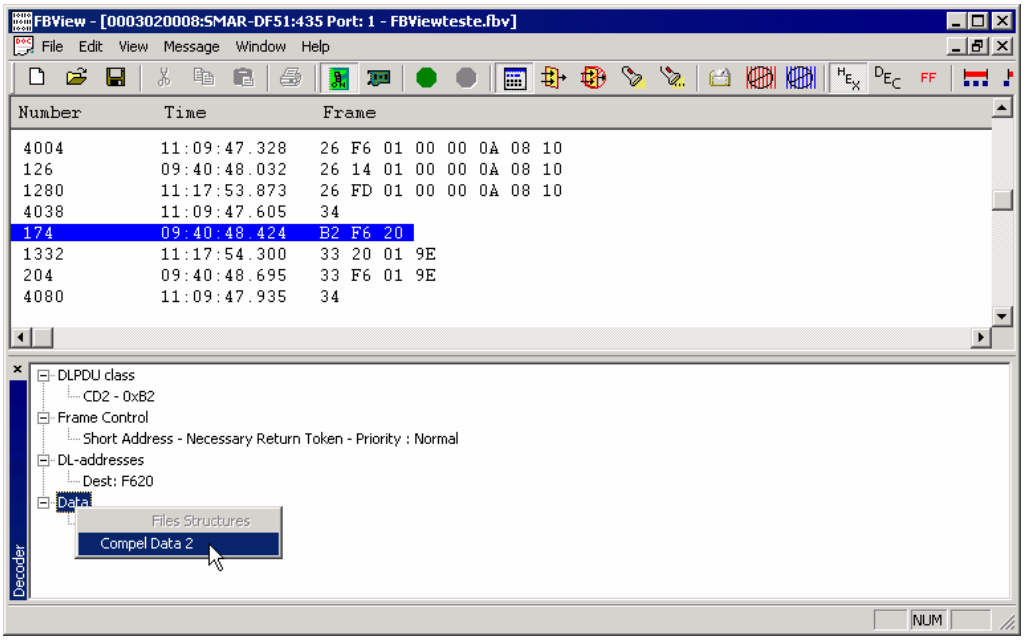


Figure 2.11. Selecting the Data Structure

Select the message format and the interpretation will be displayed according to the *XML* file:

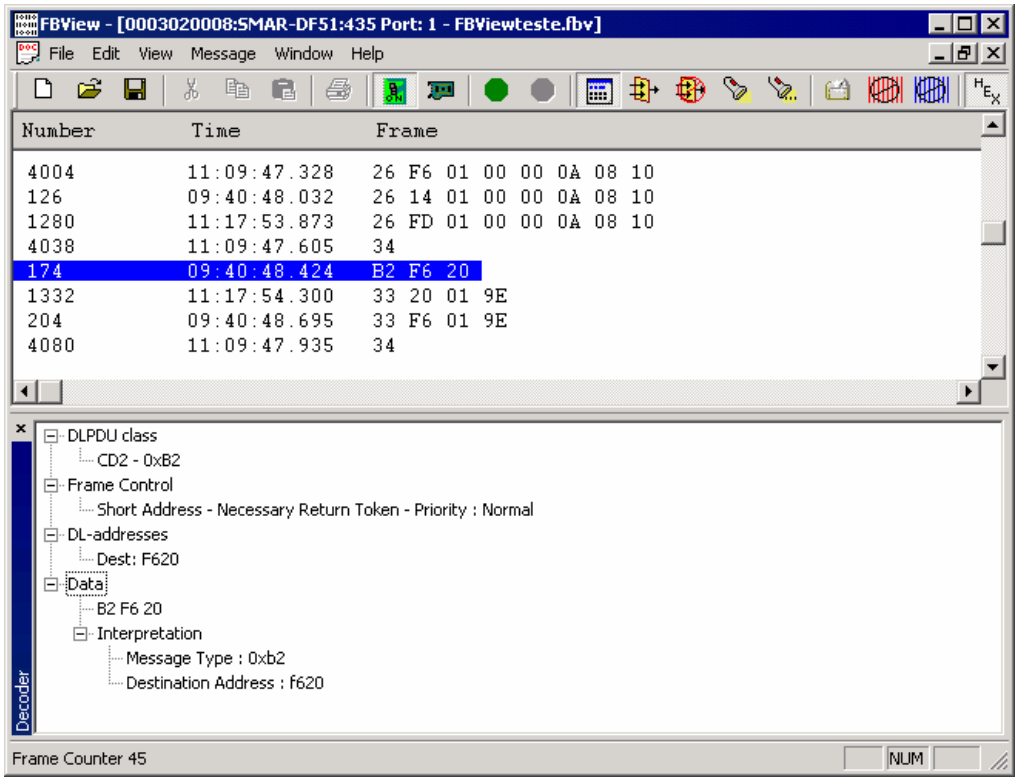


Figure 2.12. Interpretation of Compel Data 2

2.8 Saving a file in text format

The user can export the frame captures from **FBView** to a text file. This way, the user can share the Fieldbus line diagnostics with other machines that don't have **FBView** installed, or even generate diagnostic reports.

To export the messages to the text file, go to the *File* menu and click the option *Save Txt File*. The *Save as* dialog box will open:

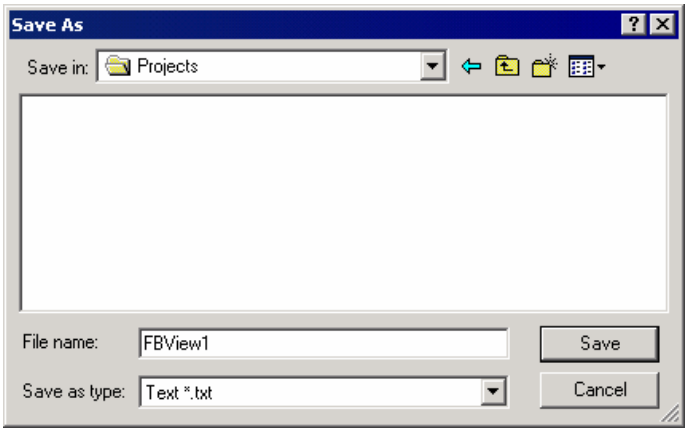


Figure 2.13. Saving as a text file

1. Browse the directories to select the folder where the txt file will be saved.
2. Type the name for the txt file.
3. Click *Ok* to conclude.

2.9 Capturing messages using the DF51

The DF51 can work only as an interface for **FBView**, in the *sniffer* mode, or as a configuration/supervision interface, plus the *sniffer* mode.

In the *sniffer* mode, the DF51 doesn't appear in the *Live List*, it only captures the messages sent by the H1 communication bus.

In the Interface+*sniffer* mode, it is possible to use the same DF51 to operate with *Syscon*, with the supervisory system and **FBView**, at the same time.

2.9.1 When to use each mode?

The Interface+*sniffer* mode is easier to use because the H1 bus analysis can be executed with the interface installed, there is no need to alter the electrical connection. Therefore, it is the most recommended to perform the analysis.

The *sniffer* mode can be used to analyse the time interval between messages or during the LAS tests of the field devices. To analyse the time interval, the Interface+*sniffer* mode doesn't have a good precision for the messages sent by the interface itself. In the LAS test of the field devices, the *Syscon* interface has to be switched off, and then another interface would be necessary to capture the messages from the H1 bus.

3. FBVIEW - H1

3.1 Selecting the Communication Interface

After initializing the communication, the *Interface* dialog box will open:

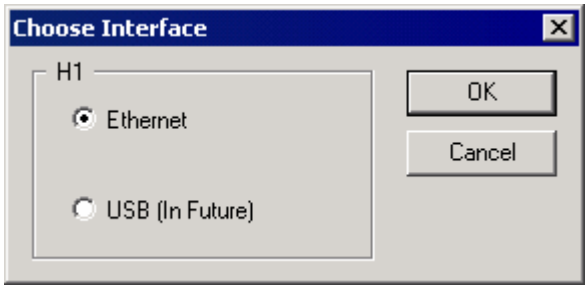


Figure 3.1. H1 Communication Interfaces

The messages file based on the H1 communication network will have the following interfaces available:

- **Ethernet**

1. Select this option and click *Ok*.
2. The *Configuration* dialog box will open:

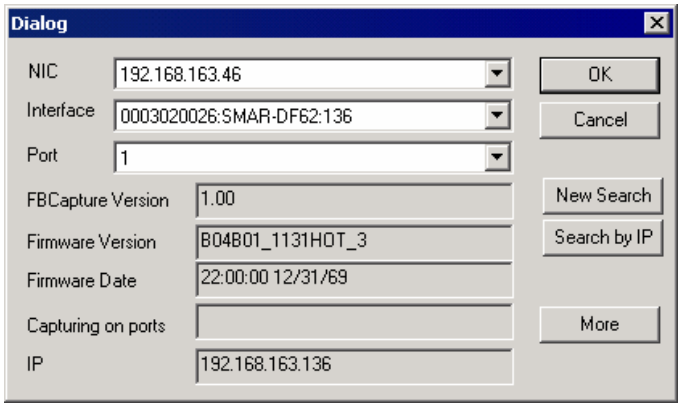


Figure 3.2. Ethernet Configuration

3. The *NIC* box shows the IP address of the machine.
4. In the *Interface* box, select the communication interface from list.
5. In the *Port* box, select the communication port conected to the selected interface.

Click the button *New Search* to automatically search for a network card available in the communication network.

Click the button *Search by IP* to open a new dialog box and type the IP address of the remote machine that has the communication interface desired. See the example below:

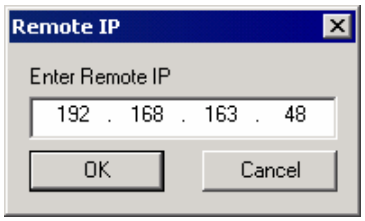


Figure 3.3. Searching for a remote IP address

When the interface is selected, specific information of this interface will be displayed in the dialog box, such as the firmware version and the IP address of the network card. The interface address is displayed instead of its name, to avoid problems with computers that have two or more cards with the same name.

Click *Ok* to conclude the interface configuration.

- **USB**

1. To capture the frames from the serial port of the PC, select this option and click *Ok*.
2. The configuration dialog box will open:

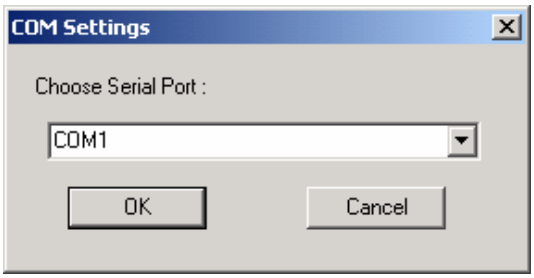


Figure 3.4. Serial Port Configuration

3. Select the serial port from the list.
4. Click *Ok* to conclude.

3.2 Filters

3.2.1 Adding Filters

To configure a filter in the *Hexadecimal* mode, click the button **Enable Filters**,  , in the toolbar. The *Filter Configuration* window showed in the figure below will open:

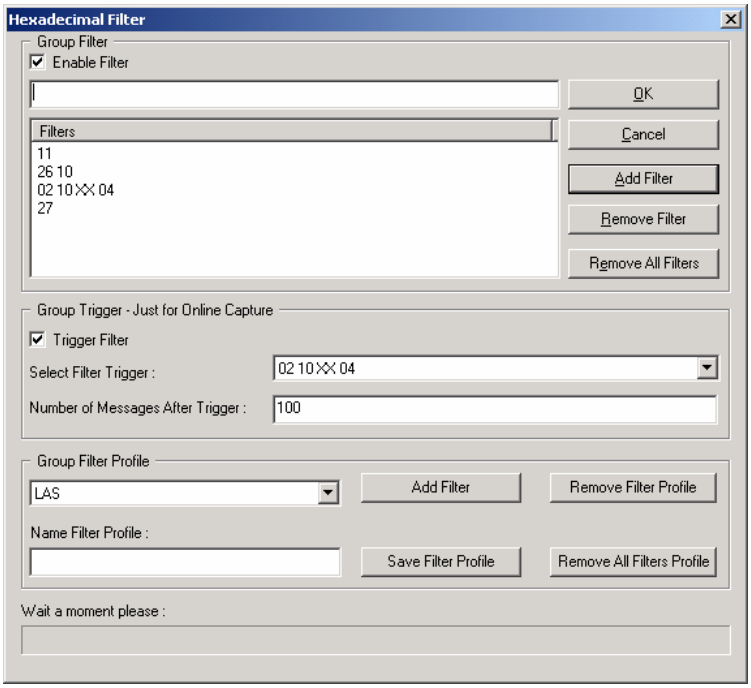


Figure 3.5. Filters Configuration

Select the option *Enable Filter* to apply the filters to the messages.

To add a new filter, type the first bytes of the message in the text box and click the button **Add Filter**.

To select the filter that will be applied to the messages being captured, mark the option *Trigger Filter*, select the filter from the *Select Filter Trigger* menu and type the number of messages to be captured.

There are codes that can be used to improve the filters and the analysis. The codes are listed in the table below:

Code Table	
Code	**Function**
XX	This code indicates that any hexadecimal number can occur in this position.
******	This code searches for the next occurrence of the following hexadecimal in the frame, independent of its position.
!!	This code captures all frames that are different from the following hexadecimal. It works as a filter "*NOT*", only when the code is at the beginning of the filter.

Code Examples	
Example	**Description**
26XX10	Captures the frames that start with 26 and have 10 in the third byte of the frame.
****33**	Captures the frames that have 33 in any position of the frame. (Note: **3310 is an invalid filter.)
!!34	Captures the frames that do not start with 34. (Note: 51!!1B is an invalid filter.)

The user can match the codes in a more coherent way having the codes on hands, to assure a better analysis of the frames and the fieldbus line.

3.2.2 Removing Filters

To remove a filter, select the specification in the *Filters* list and click the button **Remove Filter**. See the example below:

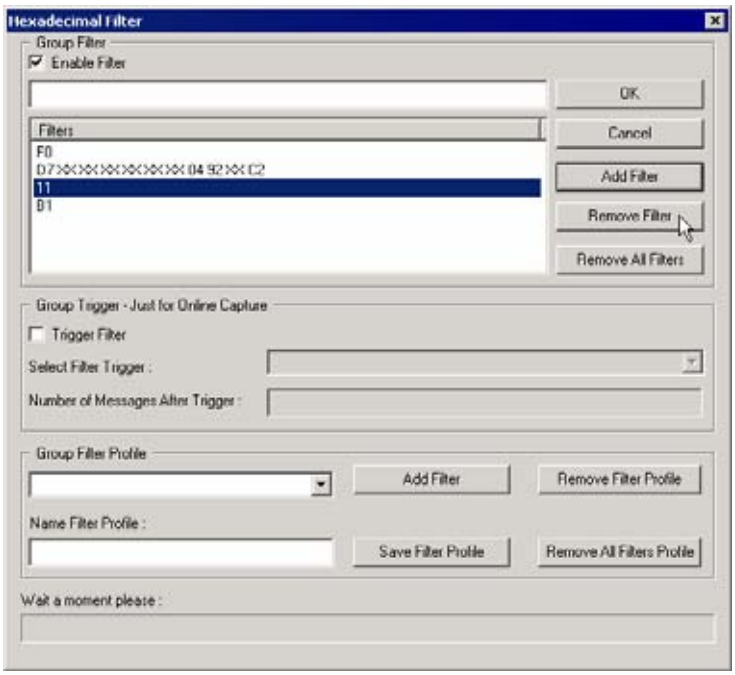


Figure 3.6. Removing a Filter

To remove all filters, click the button **Remove All Filters**.

3.2.3 Creating a Filter Profile

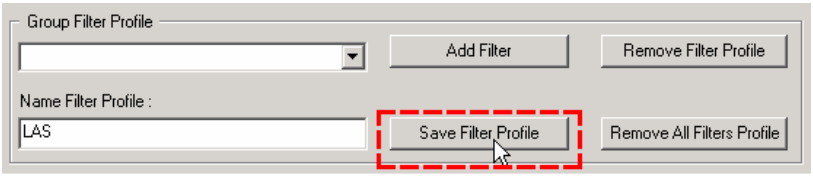
The *Filter Profile* allows the user to save all filters configured with specific names, related to their function.

The user can create a library with configured filters, being able to add/remove one or all filters from the library.

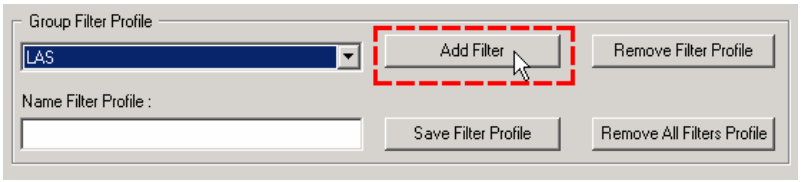
To create a filter profile:

1. Click the button **Enable Filters**,  , in the toolbar, to open the *Filter Configuration* window.
2. Type the first bytes of the message in the text box and click the button **Add Filter**. Repeat this step to add more filters to the filter profile.

3. Type a name for the filter profile and click the button **Save Filter Profile**.



After adding a new filter profile, it will be displayed in the *Group Filter Profile* menu. Use the button **Add Filter** to add the selected filter profile to a group of filters from another message buffer.



3.2.4 Removing Profiles

To remove a filter profile, select the name of the profile in the *Group Filter Profile* list and click the button **Remove Filter Profile**. See the example below:

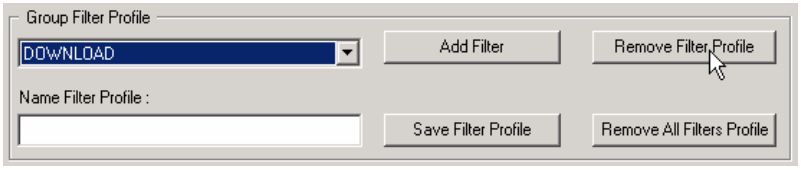


Figure 3.7. Removing a Filter Profile

To remove all filter profiles, click the button **Remove All Filters Profile**.

3.3 Basic Components of the Message

A message is composed by data from different levels of the protocol: *Data Link Layer* (DLL), *Fieldbus Access SubLayer* (FAS), *Fieldbus Message Specification* (FMS) and *System Management* (SM).

The basic composition of a message is:

FC	DA	SA	DLL-p	FAS	FMS

FC	DA	SA	DLL-p	SM

Where:

FC	*Frame Control* – indicates the type of the service, the priority and whether the token has been returned or not.
DA	*Destination Address* – address of the destination device of this message.
SA	*Source Address* – address of the device that transmitted the message.
DLL-p	*Data Link Layer* parameters.
FAS	FAS parameters.
FMS	FMS parameters.
SM	SM parameters.

3.3.1 Simplified De-codification of the Frequently Used Message

This section will describe the most frequently used messages. Some of the components will be described in the section ***"Complete Message De-codification"***. The purpose of this section is to provide the user with the basic idea on how to identify a *Read Request*, a *Write Request* and other commands frequently used for error detection in a *Fieldbus Foundation* system.

FBView suppresses the FCS when displaying a H1 message.

Pass Token (PT)

Used by the LAS to give time to the device in the same node of the DA. The device gives the token back to the LAS, after using the network, with the *Return Token* message type, or sends the last message with the third bit of the frame control set to 1.

FC	*DA*	*XX*	*FCS*

FC	Assumes the following values: 31h, 32h or 33h.
DA	Any acceptable value for the node, from 10h to FFh (16 to 255).
XX	Non-decoded bytes. Sequence with two bytes.
FCS	Two check sum bytes from the bytes of the message.

Example: Pass Token for the device in node 19h.

FC	DA	XX		FCS	
33h	19h	01h	9Fh	44h	80h

Return Token (RT)

This message returns the token to the LAS.

FC	*FCS*

FC	Its value is always 34h.
FCS	Two check sum bytes from the bytes of the message. As this is a 1-byte message and it is always with the same value, the FCS does not change and its value is AF21h.

Example:

FC	FCS	
34h	AFh	21h

Probe Node (PN)

The LAS uses this message to check if there is any device with the same node of the DA.

FC	*DA*	*XX*	*FCS*

FC	Its value is always 26h.
DA	Any acceptable value for the node, from 10h to FFh (16 to 255).
XX	Non-decoded bytes. Sequence with 6 bytes.
FCS	Two check sum bytes from the bytes of the message.

Example: Probe node for the node 19h.

FC	DA	XX						FCS	
26h	19h	01h	00h	00h	0Ah	08h	0Ch	50h	A9h

Probe Response (PR)

When a device receives a *Probe Node*, it replies with a *Probe Response*. The LAS initiates this node when it receives this message.

FC	*XX*	*FCS*

FC	Its value is always 27h.
XX	Non-decoded bytes. Sequence with 8 bytes.
FCS	Two check sum bytes from the bytes of the message.

Example: The *Probe Response* does not contain the information about which node has been transmitting. To know which node transmitted the message, it is necessary to read the *Destination Address* (DA) of the *Probe Node* right before the *Probe Response*.

FC	XX							FCS	
27h	01h	03h	00h	00h	00h	50h	0Ch	ABh	55h	B0h

Node Activation (DT1)

Every time a new node replies to a *Probe Node*, its *Data Link Layer* status is set to offline. In this mode, the node only answers to the *Probe Node*. The LAS sends an activation message to change its *Data Link Layer* to the *Operational* status. This message is sent with a *Data Transfer 1* message (DT1).

FC	*DA*	*SA*	*XX*	*FCS*

FC	Its value is always D2h.
DA	Destination address. It has two bytes: the first one assumes any acceptable value of the node. The second one is always 0.
SA	Source address. It has two bytes with values 4 and 0.
XX	Non-decoded bytes. Sequence with 11 bytes.
FCS	Two check sum bytes from the bytes of the message.

Example: Node19h activation.

FC	DA		SA		XX										FCS	
D2h	19h	00h	04h	00h	09h	00h	02h	00h	00h	00h	00h	00h	00h	DAh	ABh	D8h	22h

Establish Connection 1 (EC1)

All device information is located in the VFD. All configuration parameters of the fieldbus network are located in the management VFD (SMIB), such as: T1, T2, T3, VCRs, FB Schedule, etc. The parameters of the function blocks are located in the application VFD (FBAP). The device must have at least one SMIB. It is common to find devices with two VFDs: 1 SMIB and 1 FBAP. But the protocol allows the device to have more than one application VFD.

The device has to set the connection with the VFD to access the information from the VFD. *EC1* messages are used in this connection. To set a connection the node sends the *EC1* with a connection request and the addressed node sends another *EC1* accepting the connection.

FC	*DA*	*SA*	*XX1*	*DC*	*XX2*	*FCS*

FC	It can assume two values: F0h or F4h. The F4h is the EC1 with *Return Token* flag set at 1.
DA	Destination address. It has two bytes: the first one assumes any acceptable value for the node. The second is always 07.
SA	Source address. It has two bytes: the first one assumes any acceptable value for the node. The second one is the value of the connection's selector of the node that is transmitting the message.
XX1	Non-decoded bytes. Sequence with 19 bytes.
DC	It has two bytes. The first one has the value of the addressed node and the second one has the value of the connection's selector of the addressed node.
XX2	Non-decoded bytes. Sequence with 37 bytes.
FCS	Two check sum bytes from the bytes of the message.

Example: This is a connection request between 1191h (SA) and 18F3h (DC).

FC	DA		SA		XX1 – 13 bytes											
F0h	18h	07h	11h	91h	11h	07h	81h	76h	4Eh	20h	4Eh	20h	E1h	80h	00h	F0h	E1h
XX1 – 6 bytes						DC		XX2 – 10 bytes									
80h	00h	F0h	00h	00h	00h	18h	F3h	00h	00h	11h	91h	E2h	FFh	89h	02h	00h	00h
XX2 – 18 bytes																	
12h	4Dh	47h	21h	00h	32h	00h	00h	41h	EFh	51h	EFh	68h	00h	30h	00h	00h	00h
XX2 – 9 bytes									FCS								
00h	00h	00h	71h	07h	01h	71h	08h	01H	32h	36h							

To establish this connection, the node 18h must send an EC1 as indicated below:

FC	DA	SA	XX	FCS
F0h ou F4h	1191h	18F3h		

Observe that the DA of the confirmation is the same as the SA of the request, and the SA of the confirmation is the same as the DC of the request.

Disconnect 1 (DC1)

When the addressed node rejects a connection request or the connection is broken, the node sends a *DC1* message.

FC	DA	SA	XX	FCS

FC	It can assume two values: 70h or 74h. The 74h value is the DC1 with *Return Token* flag set at 1.
DA	Destination address. It has two bytes: the first one assumes any acceptable value of the node. The second one is the value of the selector for the addressed node connection.
SA	Source address. It has two bytes. The first one assumes any acceptable value of the node. The second one is the value of the selector for the source node connection.
XX	Non-decoded bytes. Sequence with 2 bytes.
FCS	Two check sum bytes from the bytes of the message.

Example: In the following message, the node 11h is breaking its connection 91h with the connection F3h of the node 18h.

FC	DA		SA		XX		FCS	
70h	18h	F3h	11h	91h	01h	46h	91h	E6h

Read Request

A node makes a *Read Request* when it wants to know the parameter value from another node. This message is encapsulated in a *DT1*.

FC	DA	SA	XX1	RQ	FMS PDU	INVOKE ID	READ	XX2	INDEX	FCS

FC	It can assume two values: D3h or D7h. The D7h value is the DT1 with *Return Token* flag set at 1.
DA	Destination address. It has two bytes: the first one assumes the value of the addressed node. The second one is the value of the selector for the addressed node connection.
SA	Source address. It has two bytes: the first one assumes the source node's value. The second has the selector's value of the source node connection.
XX1	Non-decoded bytes. Sequence with 2 bytes.
RQ	Byte that identifies the message as a request. Assumes value 4.
FMS PDU	Byte that identifies the message as a request with confirmation. Assumes value 92h.
INVOKE ID	This byte is a sequence number. It is used to identify the answer. That is, when the addressed node answers this request, it sends the *invoke ID* received in the request.
READ	Byte that identifies the message as a reading. It always assumes the value A1h.
XX2	Non-decoded byte.
INDEX	Two bytes that identify which INDEX of the OD have been read.
FCS	Two check sum bytes from the bytes of the message.

Example: In the following message, the node 1091h is requesting the reading of index 0960h of the node 19F7h.

FC	DA		SA		XX1		RQ	FMS PDU	INVOKE ID	READ	XX2	INDEX		FCS	
D3h	19h	F7h	10h	91h	70h	47h	04h	92h	67h	A1h	02h	09h	60h	B3h	9Ch

Read Response

This message is the response to a *Read Request*. The node must send the same *invoke ID* as was received in the *Request*.

FC	*DA*	*SA*	*XX1*	*RS*	*FMS PDU*	*INVOKE ID*	*LENGTH*	*DATA*	*FCS*

FC	It can assume two values: D3h or D7h. The D7h value is the DT1 with *Return Token* flag set at 1.
DA	Destination address. It has two bytes: the first one assumes the value of the addressed node. The second has the selector's value of the addressed node connection.
SA	Source address. It has two bytes: the first one assumes the value of the source node. The second has the selector's value of the source node connection.
XX1	Non-decoded bytes. Sequence with 2 bytes.
RS	Byte that identifies the message as a response. Assumes value 5.
FMS PDU	Byte that identifies the message as a confirmation. Assumes value A2h.
INVOKE ID	This byte is a sequence number. It is used to identify the answer. That is, the node will send the *invoke ID* received in the request.
LENGTH	Byte that identifies the data length that was read. If the length is less than 15 bytes, the value will be 2Xh, where X is the length. If the length is greater than or equal to 15 bytes, we will have the byte 2Fh and the next byte will be the data length.
FCS	Two check sum bytes from the bytes of the message.

Example: Response message to the *Read Request* example above. In this case, the data length is 8 bytes.

FC	DA		SA		XX1		RS	FMS PDU	INVOKE ID
D7h	10h	91h	19h	F7h	80h	47h	05h	A2h	67h
LENGTH	**DATA**								**FCS**
28h	00h	00h	00h	00h	00h	00h	00h	00h	08C8h

Write Request

A node makes a *Write Request* when it wants to configure a new value of the parameter from another node. This message is encapsulated in a *DT1*.

FC	*DA*	*SA*	*XX1*	*RQ*	*FMS PDU*	*INVOKE ID*
WRITE	*XX2*	*INDEX*	*LENGTH*	*DATA*	*FCS*	

FC	It can assume two values: D3h or D7h. The D7h value is the DT1 with *Return Token* flag set at 1.
DA	Destination address. It has two bytes: the first one assumes the value of the addressed node. The second has the selector's value of the addressed node connection.
SA	Source address. It has two bytes: the first one assumes the value of the source node. The second has the selector's value of the source node connection.
XX1	Non-decoded bytes. Sequence with 2 bytes.
RQ	Byte that identifies the message as a request. Assumes value 4.
FMS PDU	Byte that identifies the message as a request with confirmation. Assumes value 92h.
INVOKE ID	This byte is a sequence number. It is used to identify the answer. That is, when the addressed node answers this request, it sends the *invoke ID* received in the request.
WRITE	Byte that identifies the message as a writing. It always assumes value B2h.
XX2	Non-decoded byte.
INDEX	Two bytes that identify which INDEX of the OD to be written.
LENGTH	Byte that identifies the data length to be written. If the length is less than 15 bytes, the value will be 4Xh, where X is the length. If the length is greater than or equal to 15 bytes, we will have the byte 4Fh and the next byte will be the data length.
DATA	Value to be written.
FCS	Two check sum bytes from the bytes of the message.

Example: In the following message, the node 1091h is writing 7 bytes in the index 01C7h of the node 19F7h.

FC	DA		SA		XX1		RQ	FMS PDU	INVOKE ID
D3h	19h	F7h	10h	91h	80h	58h	04h	92h	58h
WRITE	**XX2**	**INDEX**		**LENGTH**	**DATA – 5 bytes**				
B2h	02h	01h	2Ch	47h	02h	80h	02h	01h	D0h
DATA – 2 bytes		**FCS**							
00h	01h	61h	E2h						

Write Response

This message is sent in response to a *Write Request*. The node must send the same *invoke ID* as received in the *Request*.

FC	DA	SA	XX1	RS	FMS PDU	INVOKE ID	WRITE	FCS

FC	It can assume two values: D3h or D7h. The D7h value is the DT1 with *Return Token* flag set at 1.
DA	Destination address. It has two bytes: the first one assumes the value of the addressed node. The second has the selector's value of the addressed node connection.
SA	Source address. It has two bytes: the first one assumes the value of the source node. The second has the selector's value of the source node connection.
XX1	Non-decoded bytes. Sequence with 2 bytes.
RS	Byte that identifies the message as a response. Assumes value 5.
FMS PDU	Byte that identifies the message as a confirmation. Assumes value A2h.
INVOKE ID	This byte is a sequence number. It is used to identify the answer. That is, the node will send the *invoke ID* received in the request.
WRITE	Byte that identifies the message as positive response.
FCS	Two check sum bytes from the bytes of the message.

Example: Response message from the *Write Request* example above.

DC	DA		SA		XX1	
D7h	10h	91h	19h	F7h	90h	58h
RS	**FMS PDU**	**INVOKE ID**	**WRITE**	**FCS**		
05h	A2h	58h	30h	C0h	56h	

Get OD Request

The node sends this message to request an *OD* description of a specific parameter. This message is encapsulated in *DT1*.

FC	DA	SA	XX1	RQ	FMS PDU	INVOKE ID	GET OD	XX2	INDEX	FCS

FC	It can assume two values: D3h or D7h. The D7h value is the DT1 with *Return Token* flag set at 1.
DA	Destination address. It has two bytes: the first one assumes the value of the addressed node. The second has the selector's value of the addressed node connection.
SA	Source address. It has two bytes: the first one assumes the value of the source node. The second has the selector's value of the source node connection.
XX1	Non-decoded bytes. Sequence with 2 bytes.
RQ	Byte that identifies the message as a request. Assumes value 4.
FMS PDU	Byte that identifies the message as a request with confirmation. Assumes value 92h.
INVOKE ID	This byte is a sequence number. It is used to identify the answer. That is, when the addressed node answers this request it sends the *invoke ID* received in the request.

GET OD	Byte that identifies the message as a Get OD. It always assumes the value C2h.
XX2	Non-decoded byte. Sequence with 3 bytes.
INDEX	Two bytes that identify which INDEX of the OD to be described.
FCS	Two check sum bytes from the bytes of the message.

Example: In the following example, the node 1090h is requesting a *Get OD* to index 02C1h of node 1AF7h.

DC	DA		SA		XX1		RQ	FMS PDU
D3h	1Ah	F7h	10h	90h	30h	43h	04h	92h
INVOKE ID	**GET OD**	**XX2**			**INDEX**		**FCS**	
58h	C2h	01h	00h	12h	02h	C1h	7Eh	64h

Get OD Response

This message is sent in response to a *Get OD Request*. The node must send the same invoke ID as received in the request.

FC	*DA*	*SA*	*XX1*	*RS*	*FMS PDU*	*INVOKE ID*	*GET OD*	*XX2*	*More*	*FCS*

FC	It can assume two values: D3h or D7h. The D7h value is the DT1 with *Return Token* flag set at 1.
DA	Destination address. It has two bytes: the first one assumes the value of the addressed node. The second has the selector's value of the addressed node connection.
SA	Source address. It has two bytes: the first one assumes the value of the source node. The second has the selector's value of the source node connection.
XX1	Non-decoded bytes. Sequence with 2 bytes.
RS	Byte that identifies the message as a response. Assumes value 5.
FMS PDU	Byte that identifies the message as a confirmation. Assumes value A2h.
INVOKE ID	This byte is a sequence number. It is used to identify the answer. That is, the node will send the *invoke ID* received in the request.
Get OD	Byte that identifies the message as positive response.
XX2	Non-decoded bytes; the amount of bytes in this sequence depends on the description type that was asked.
More	This byte indicates there is more information. The value 00h means that there are not, and FFh means that there is more information on this parameter (More Follows).
FCS	Two check sum bytes from the bytes of the message.

Example: Response message from the *Get OD Request* example above.

DC	DA		SA		XX1		RS	FMS PDU
D7h	10h	90h	1Ah	F7h	40h	43h	05h	A2h
INVOKE ID	**GET OD**	**XX2 – 7 bytes**						
58h	C2h	81h	05h	02h	C1h	09h	00h	45h
XX2	**MORE**	**FCS**						
11h	00h	1Ah	E8h					

Compel Data 2 (CD2)

When it is time for a output parameter of a link to be published, the LAS sends a *Compel Data 2* to request the node that produces this output to send it in that moment.

FC	*DA*	*FCS*

FC	B1h, B2h and B3h.
DA	Destination address. It has two bytes: the first one assumes the value of the addressed node. The second has the selector's value of the addressed node connection.
FCS	Two check sum bytes from the bytes of the message.

Example: This is a CD2 to the node 2221h.

FC	DA		FCS	
B1h	22h	21h	A5h	BEh

Information Report

When a node receives the *CD2* it immediately sends the output parameter of the link specified by the *DA* of the *CD2*. This message is encapsulated in a *DT3*.

FC	*SA*	*XX1*	*FMS PDU*	*INVOKE ID*	*INF. REPORT*
XX2	*INDEX*	*LENGHT*	*DATA*	*FCS*	

FC	It can assume the values: 50h, 51h, 52h, 53h or 54h.
SA	Source address. It has two bytes: the first one assumes the value of the source node. The second has the selector's value of the source node connection.
XX1	Non-decoded bytes. Sequence with 2 bytes.
FMS PDU	Byte that identifies the message with no confirmation. Assumes value C2h.
INVOKE ID	Its value is always FFh.
INF. REPORT	Byte that identifies the message as an *information report*. Assumes value 82h.
XX2	Non-decoded byte.
INDEX	OD index of the published parameter.

LENGTH	Byte that identifies the data length to be published. If the length is less than 15 bytes, the value will be 4Xh, where X is the length. If the length is greater than or equal to 15 bytes, we will have the byte 4Fh and the next byte will be the data length.
DATA	Value of the published parameter.
FCS	Two check sum bytes from the bytes of the message.

Example: *Information Report* of the *CD2* from the example above.

FC	SA		XX1		FMS PDU	INVOKE ID	INF. REPORT	XX2	INDEX
51h	22h	21h	0Fh	06h	C2h	FFh	82h	02h	04h
INDEX	**LENGTH**	**DATA**					**FCS**		
54h	45h	80h	3Fh	80h	BFh	B2h	63h	32h	

Multi Variable Contained (MVC)

This view can be configured and used to optimize the bus usage during supervision. The node start sending it periodically after the MVC is configured. The periodicity is measured in macrocycles and it is one of the MVC configuration parameters.

FC	*DA*	*SA*	*XX*	*FCS*

FC	It can assume two values: D3h or D7h. The D7h value is the DT1 with *Return Token* flag set at 1.
DA	Destination address. It has two bytes with the value 0140h.
SA	Source address. It has two bytes: the first one assumes the value of the source node. The second one is 08h.
XX	Non-decoded bytes; the amount of bytes in this sequence depends on the parameters that were configured.
FCS	Two check sum bytes from the bytes of the message.

Example: MVC of the node 18h.

FC	DA		SA		XX – 9 bytes								
D3h	01h	40h	18h	08h	06h	C2h	FFh	82h	02h	0Ah	18h	4Dh	00h
XX – 12 bytes												**FCS**	
D1h	00h	09h	00h	0Dh	80h	00h	80h	3Bh	D1h	E2h	48h	DEh	C3h

3.3.2 Complete Message De-codification

DLPDU Class	FC	DL Address				
	Frame Control	Destination (DA)	Source (SA)	2nd source	Parameters	User Data
EC 1	1111 LF00	[HL.]N.S	[HL.]N.S	[HL.]N.S	EC-p	o-DLSDU
EC 2	1110 LF00	—	[HL.]N.S	[HL.]N.S	EC-p	o-DLSDU
DC 1	0111 LF00	[HL.]N.S	[HL.]N.S	—	DC-p	o-DLSDU
DC 2	0110 LF00	—	[HL.]N.S	—	DC-p	o-DLSDU
CD 1	1111 LFPP	[HL.]N.S	[HL.]N.S	—	—	—
CD 2	1011 LFPP	[HL.]N.S	—	—	—	—
DT 1	1101 LFPP	[HL.]N.S	[HL.]N.S	—	SD-p	o-DLSDU
DT 2	1001 LFPP	[HL.]N.S	—	—	SD-p	o-DLSDU
DT 3	0101 LFPP	—	[HL.]N.S	—	SD-p	o-DLSDU
DT 5	0101 0F00	—	[PDA]	—	SD-p	o-DLSDU
SR	0001 0F11	[PSA]	N	—	o-SR-p	—
CT	0001 0F00	—	—	—	—	—
TD	0001 0F01	—	N	—	TD-p	—
RQ	1100 0F00	N.0	N.0	—	RQ-p	—
RR	1101 0F00	N.0	N.0	—	RR-p	—
PN	0010 0110	N	—	—	PN-p	—
PR	0010 0111	—	—	—	—	SPDU
PT	0011 0FPP	N	—	—	DD-p	—
RT	0011 0100	—	[DTH]	—	—	—
RI	0010 0000	—	[DTH]	—	DD-p	—
CL	0000 0001	—	N	—	—	—
TL	0000 0110	N	—	—	—	SPDU
Idle	0001 0F10	—	—	—	—	o-DLSDU

Table 1 - Data Link Layer Message Structure

LEGEND:

L	Indicates the length of the addresses used in the message (0=Short, 1=Long).
F	Indicates that the token will not be used anymore. The device is returning the token to LAS.
PP	Specifies the priority of the message (urgent, normal, time available).
—	Indicates that the field is not used.
[HL.]N.S	The address bytes. If L is equal to 1 there will be long addresses, which length is 4 bytes (HLNS), but if L is equal to 0 there will be short addresses, which length is 2 bytes (NS).
N	Indicates that the address is 1 byte, only with the node information (node address).
N.0	Indicates the short address where the first byte contains the node (node address) and the second byte is equal to zero.
[PDA]	The source address. In case the message does not have the destination address. This type of addressing is normally used in the information report message.
[PSA]	The destination address.
o-	Indicates that the field is optional.
xx-p	Indicates the parameter class of the DLL (ex: Time Distribution, Probe Node, etc.).
DLSDU	Parameters of the "*DL Service Data Unit*".
SPDU	Parameters of the "*Support Protocol Data Unit*". These are the parameters from the other levels of the protocol (FMS, SM and FAS).

Pass Token (PT = 33)

FC	DA	Duration		FCS	
33h	XX	XX	XX	XX	XX

FC	It can assume different values according to the priority. The valid values are 31h, 32h and 33h, corresponding to the priorities urgent, normal and time available, respectively.
DA	Any valid value of the node from 10h to FFh (16 to 255).
Duration	Two bytes that contain the available time for the token use. Each increase values 256us.
FCS	Two check sum bytes from the bytes of the message.

Return Token (RT = 34)

FC	FCS	
34	AF	21

FC	Its value is always 34h.
FCS	Two check sum bytes from the bytes of the message. As it is a 1-byte message, always has the same value. The FCS does not change and it is equal to AF21h.

Probe Node (PN = 26)

FC	DA	PNp-1	PNp-2	Slot Time		VMID		FCS	
26	XX	01	00	XX	XX	XX	XX	XX	XX

FC	Its value is always 26h.
DA	Any valid value of the node from 10h to FFh (16 to 255).
PNp-1	Currently with value 1. Bit 7 to 4 – maximum inter-channel signal skew. Bit 3 – zero. Bit 2 to 0 – version of the DLL protocol.
PNp-2	Currently with value 0. Bit 7 to 4 – Post transmission Gap extension.
Slot Time	Two bytes that contain the slot time.
VMID	Two bytes that contain the minimum inter PDU delay.
FCS	Two check sum bytes from the bytes of the message.

3.4 Methodology

3.4.1 Signal Quality (CRC Test)

It is possible to quantify the quality of the signal using the Statistics View. Follow the steps below:

i. Disable the filters.

ii. Wait until 32,000 messages are captured.

iii. If the CRC error is less than 0.8%, it means that this bus will not have installation problems.

iv. If the CRC error is greater than or equal to 0.8%, it means that the bus will have a problem.

Here are some tips on where to look for the cause of the problem:

- Problems with terminators (BT): bad connection, lack or excess of BT;
- Bad grounding;
- There is water in the junction boxes or inside the devices;
- Transmitters with low insulation;
- Digital board of any transmitter has a problem;
- Interface with problem (PCI or DF51).

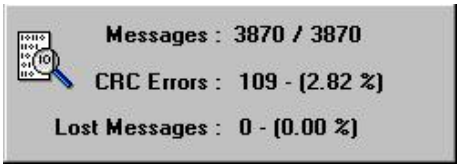


Figure 3.8. Statistics with CRC Error

3.4.2 Live List

The LAS (master) has an internal list with all of the devices that are communicating in that moment. This list is named *Live List*.

The LAS constantly verifies if there is any device in a specific address. The LAS sends *Probe Node* (PN) messages to every address that is not listed in the *Live List*. If a device receives a *PN* for its address it immediately answers with a *Probe Response* (PR), and afterward the LAS sends an activation command and passes the token to this device.

To leave the *Live List* a device must quit answering the token for three consecutive times.

The figure below shows a sequence of a device entering the *Live List* and leaving it. The following filters were used for this capture:

- 26 19
- 27
- D2 19 XX 04
- 33
- 34

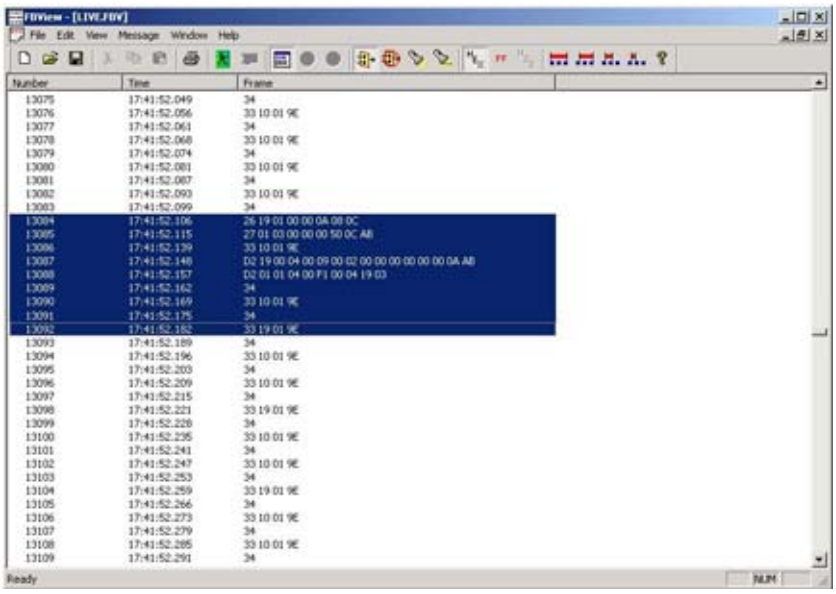


Figure 3.9. Live List Maintenance (Input)

The messages from number 13076 to 13083 shows that the token passing was being made only for node 10H.

The message of number 13084 is a probe node for the node 19H. The device from node 19H answers immediately with a *Probe Response* in the message of number 13085. A little below in the message of number 13087, the LAS is making the activation of that device. After that we can see that the LAS began to pass the token for that node (messages of number 13090 to 13098).

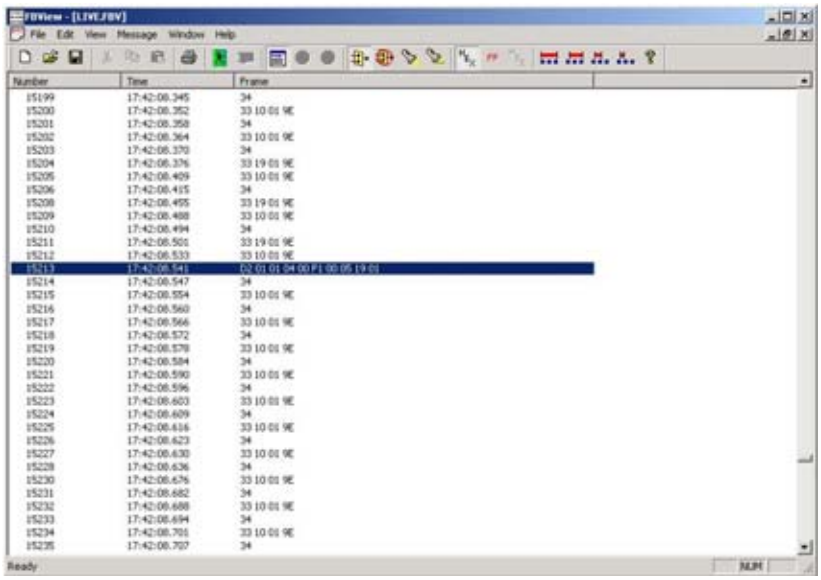


Figure 3.10. Live List Maintenance (Output)

The messages 15204, 15208 and 15211 in Figure 3.10 shows the three attempts of the LAS to pass the token for that node, but with no success. In this case, the LAS removes that node from the *Live List*. This can be verified starting from the message 15215, where we see only the token passing for the node 10H (messages 15215, 15217, 15219, 15221 and 15223).

As we can see in the previous example, it is enough to analyze the *PT* message to know which devices are in the live list at a certain moment, because the LAS is passing the token for all of the nodes that are active.

3.4.3 Link Master - LAS

Each fieldbus has a device that controls the use of the transmission means (bus). That device is named *Link Master*.

It is possible to have more than one device with *Link Master* capability in the same bus, but in a specific moment only one of them will be the *Active Link Master*, also named LAS.

To determine the LAS, search for the *Time Distribution* message (TD), because only the LAS can send this message type.

When the LAS stops communicating, another device takes the control of the bus. The *Link Master* that becomes active sends the message *Claim LAS* to inform that it will assume the control.

Fieldbus has the concept of *Preferential LAS*. The *Link Master* that has the parameter ***PrimaryLinkMasterFlag*** equal to TRUE is the *Preferential LAS*. When the *Preferential LAS* is not the LAS of the network, it always asks to become the LAS. It sends the *DT* message, *Transfer LAS Rolls*, requesting the transfer control of the bus to the LAS. The LAS sends the *Transfer LAS* to transfer the role of LAS to the *Link Master*.

Figure 3.11 shows the messages related with the transfers of control of the bus. The following filters were used:

- 11
- 01
- 06
- 2610
- 27
- D210XX04

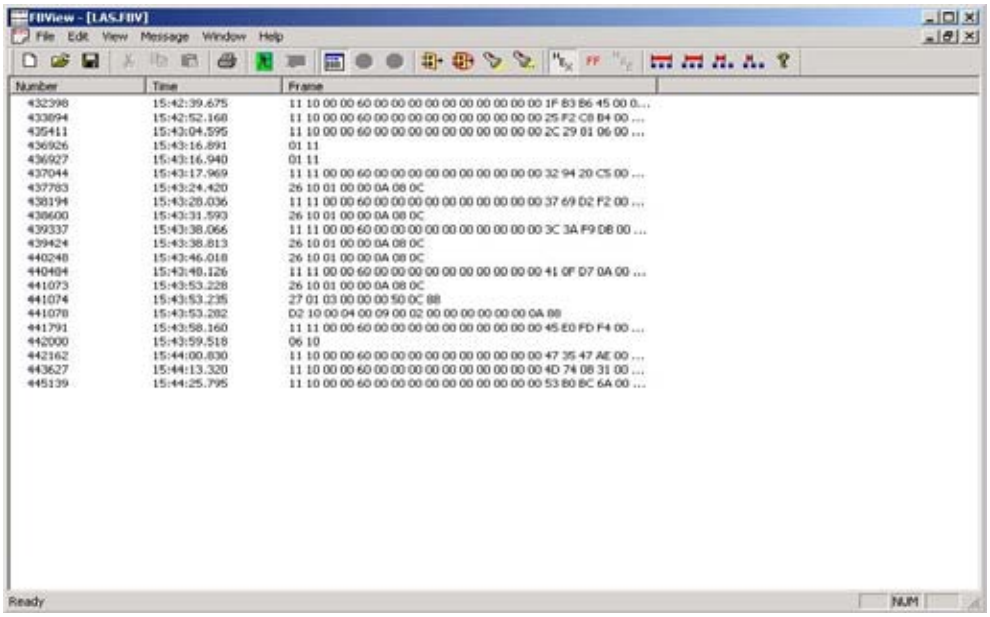


Figure 3.11. Link Active Schedule Mechanism (LAS)

The messages 432398, 433894 and 435411 are the *TD* messages from the LAS of the node 10h. In the message 436926 the LAS of the node 11h assumed the control after the node 10h was removed from the bus. The node 11h starts to send the *TD*. After the node 10h returns to the bus (messages 441073, 441074 and 441078) it asks to assume control. The LAS then sends the *TL* (message 442000) and right after that we can see the node 10h sending the *TD* (messages 442162, 443627 and 445139).

3.4.4 Publishing Control (Traffic Schedule)

The most important task of the LAS is to control the publishing, that is the control of the time of publishing each link output parameter between function blocks from different devices. The period of time to publish the same parameter should be recurrent and equal to the macrocycle.

The LAS transmits the *Compel Data* message (CD) to inform the device that is the time to transmit the specific parameter. When the device receives that message it transmits the value of that parameter using the information report service encapsulated in a *Data Transfer 5* (DT5).

3.4.5 Master Backup

The field device configured as master backup is a LAS with the ***PrimaryLinkMasterFlag*** equal to FALSE. To know that a master backup is correctly configured it is necessary to verify the following characteristics:

- It assumes the network control when the other LAS are removed.
- It is executing the traffic schedule.

The procedure to test a master backup is:

i. Certify that the system interface (PCI/DF51) is connected to the bus.
ii. Starts **FBView** and configure the filters with:
   - 11
   - B1
iii. Disable the Statistics View to see the messages that are being captured.
iv. Initiate the capture. Certify that one of the interfaces is the LAS.
v. Wait for at least 4 macrocycle periods and then stop the capture.
vi. Write down which traffic schedule has been executed and the macrocycle.
vii. Configure the filters with:
   - 11
   - 06
   - 01
viii. -initiate the capture.
ix. Remove all of the devices with LAS capability for the bus, leaving just the device to be tested.
x. Certify that the device to be tested assumed the LAS role. If it did not, remove the LAS device from the bus. Repeat this step until the tested device assumes the LAS role.

   If the bus activity stops at any moment, it will mean that the tested device does not have the LAS capability or this capability is not configured. In this case, the test can be interrupted in this step.
xi. If the tested device assumed the bus control then configure the filters with:
   - 11
   - B1
xii. Re-initiate the capture.
xiii. Wait for at least 4 macrocycle periods and then stop the capture. In this case it is also important to wait for a TD to certify which is the LAS.
xiv. Verify that the traffic schedule and the macrocycle has the same values as written in step ***vi***.
xv. If the traffic schedule and the macrocycle are the same, then the device is correctly configured to work as LAS. If not, the device is not correctly configured to work as LAS.
xvi. Reconnect all the devices and finish the test.

The following figures show a sequence of **FBView** screens for a bus where there are three LAS, which are the nodes 10h, 11h and 19h. The device that will be tested is the node 19h. The node 10h is the *Preferential LAS*.

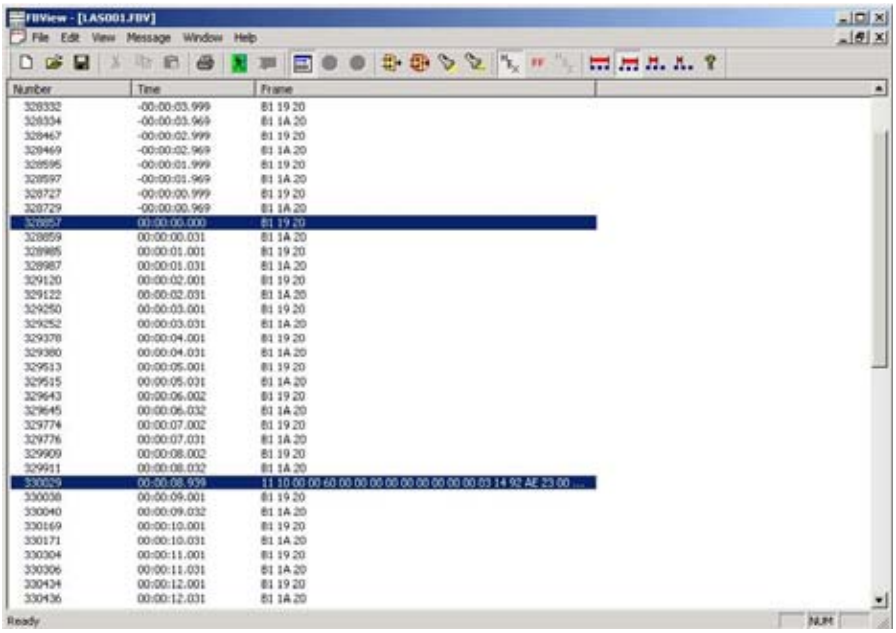


Figure 3.12. Traffic Schedule from preferential LAS

The capture showed in Figure 3.12 had the filters:

- 11
- B1

It shows that the LAS is the node 10h (message 909 – TD) and the messages 0 and 2 shows that the traffic schedule is:

- B1 19 20
- B1 1A 20

Observe that the frame 0 is selected and the option relative time measure is enabled. In this way the macrocycle is the period of time for the next "B1 19 20" message to appear. Therefore, the macrocycle is 1001 ms that is the time for the message 328985.

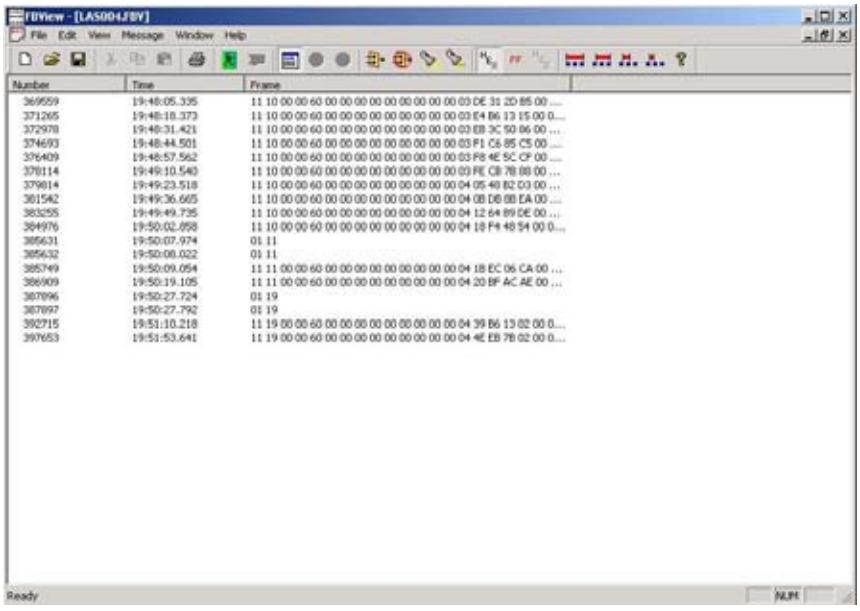


Figure 3.13. Other LAS removed

The capture showed in Figure 3.13 has the filters:

- 11
- 01
- 06

When the node 10h was removed the LAS 11h assumed control (message 385631). Only after the node 11h was removed the LAS that we wanted to test assumed control (message 387896). That indicates the node 19h is configured as LAS. If it was not configured as LAS, after we removed the node 11h, then we would not have had any captured messages. The bus activity would have stopped.

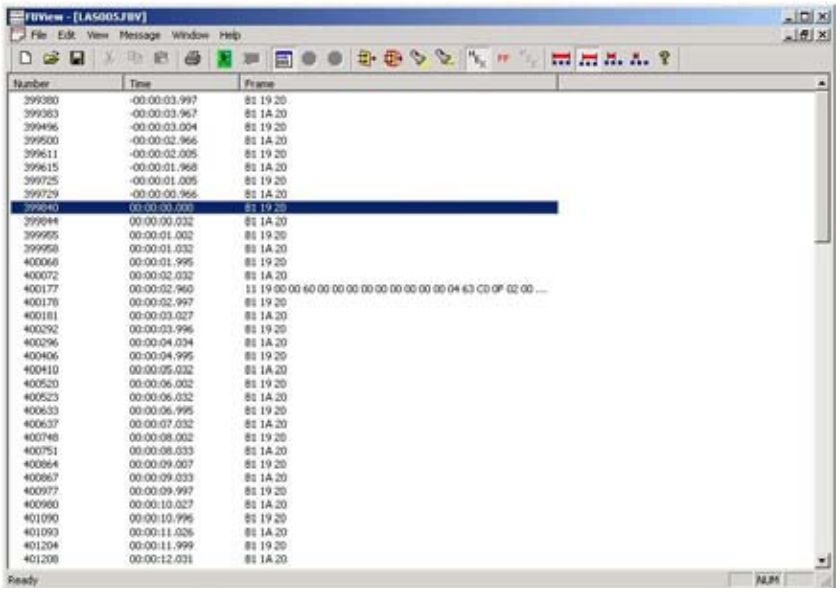


Figure 3.14. Master Backup Traffic Schedule

The capture showed in Figure 3.14 has the filters:

- 11
- B1

It shows that the LAS is the node 19h (message 337 – TD) and the messages 0 and 4 shows that the traffic schedule is:

- B1 19 20
- B1 1A 20

Note that the frame 0 is selected and the option relative time measure is enabled. In this way the macrocycle is the period of time for the next "B1 19 20" message to appear. Therefore, the macrocycle is 1002 ms which is the time for the message 399955.

As the traffic schedule from node 10h and 19h are the same and the macrocycle is closed, we can say that the node 19h has the correct LAS configuration.

3.4.6 Checking Links

It is possible to diagnose the external links with **FBView**, once the internal links are not transmitted in the bus. The external links always have the device that publishes the output parameter (named *publisher*), the device that subscribes the published value (named *subscriber*), and the device that controls the publishing (the LAS described above).

In the example:

- Node 19h is publishing a link in 1920h, which the node 1Ah is subscribing.
- Node 1Ah is publishing a link in 1A20h, which the node 19h is subscribing.

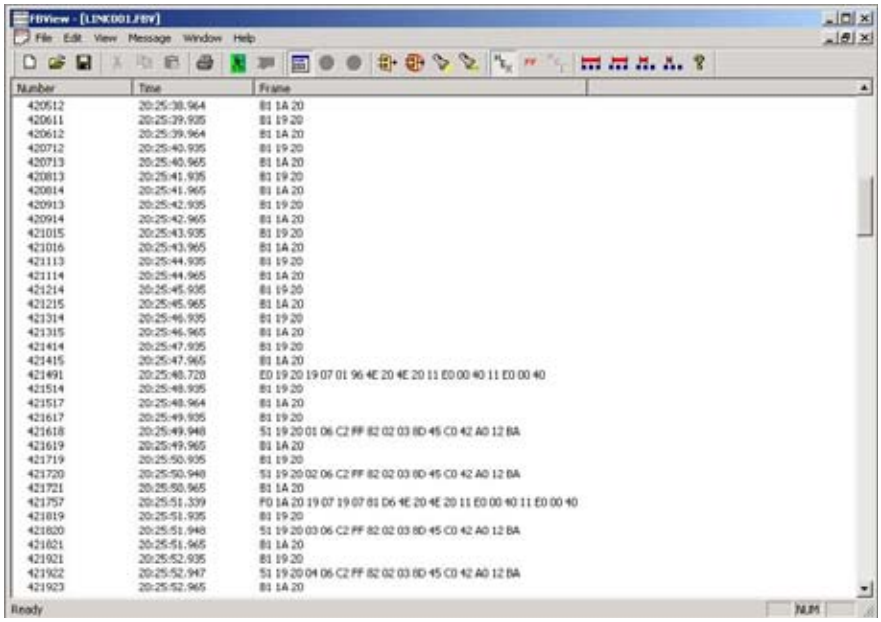


Figure 3.15. Node 19h Publishing

The capture showed in Figure 3.15 had the filters:

- B1
- 51
- EO
- E4
- FO
- F4

In the beginning the messages from 421113 to 421719 show that none of the parameters were published, because after the *Compel Data* (CD) there is no information report (DT5). This is because the two devices were removed from the bus.

The first device connected to the bus was the one from node 19h. Before it starts to publish a parameter the node sends a message to establish the *Publisher* connection. This message is an *Establish Connection 2* (EC2, message 421491) type and afterward the node starts to send the information report (DT5, messages 421618, 421720, 421820 and 421922).

As this node also has a parameter subscribing a value, it sends connection requests (establish connection EC1) to the publisher address of this link, in the example it is 1A20h (message 421757). The node will not subscribe a value until the publisher answers this connection request.

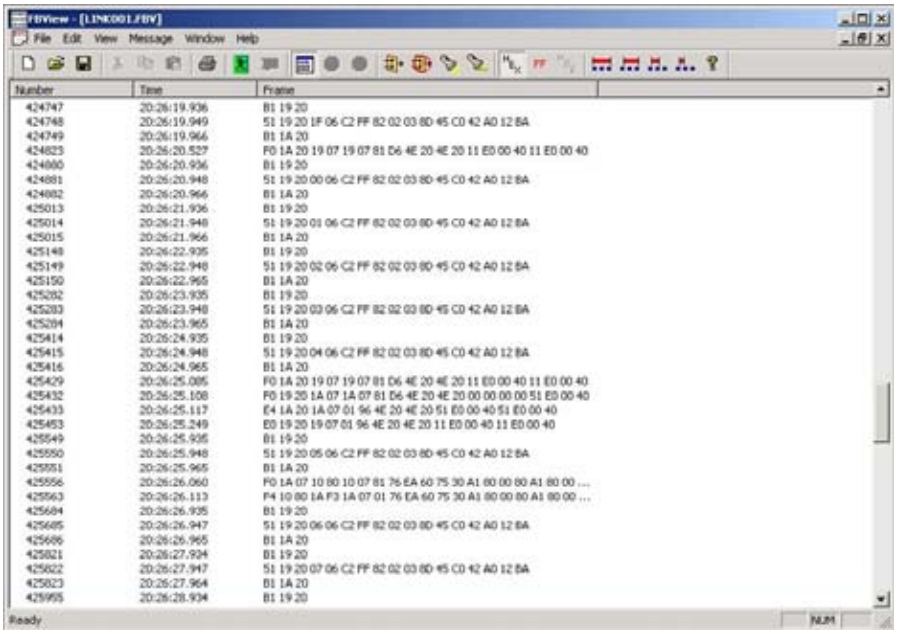


Figure 3.16. Established Links

Figure 3.16 shows two links being established. The message 424823 shows the node 19h trying to establish the connection with the publisher 1A20h again but it still does not have the answer. In the message 425429 the node 19h tries to establish the connection again and now the connection is established, because the publisher answered the connection request (message 425433).

This figure also shows the link where the node 19h is the publisher being established. In the message 425432 the link subscriber is requesting the connection and in the message 425453 the node 19h answers the request.

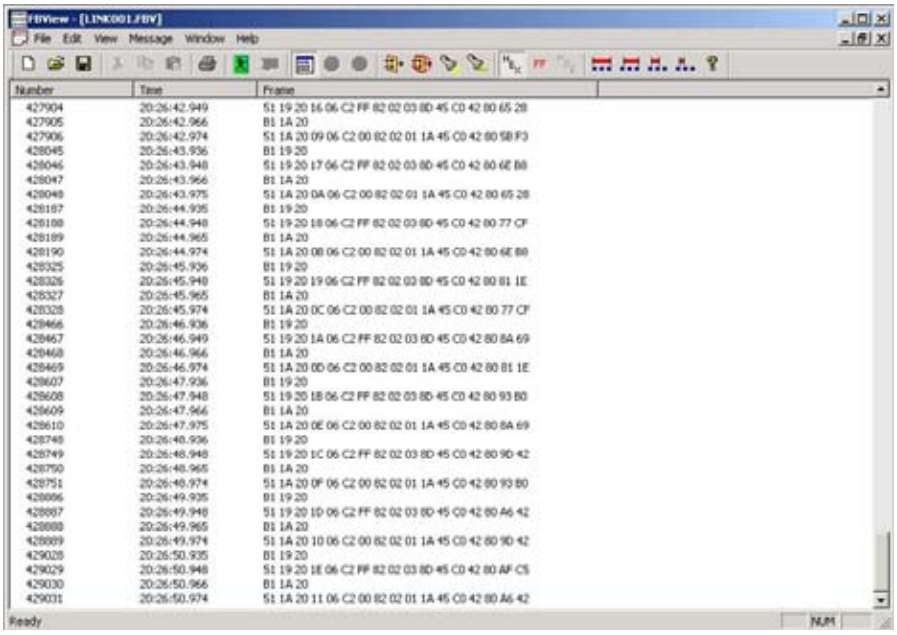


Figure 3.17. Sequence with Established Links

Figure 3.17 shows the *Compel Data* and *Information Report* messages for a bus where all of the links were established and the capture used the filters below.

- B1
- 51
- EO
- E4
- FO
- F4

3.4.7 Supervision and MVC

The simplest way to execute the parameter supervision of the device is by using an object named *View*. Each function block has four *View* types.

View is a pre-configured list of parameters. When the device receives a *Read Request* of a specific *View*, it sends the current values of each parameter of the function block that composes this *View*.

In some cases, there are function block parameters that do not belong to any *View*. Then, if the analyzer system wants to monitor this parameter it must send a *Read Request* to the parameter index.

Some fieldbus devices have an object that optimizes the time spent with supervision. This object is named *MVC* (Multi Variable Contained). The analyzer system is responsible for the configuration of the parameters that will be sent in the *MVC* and the period of time this *MVC* must be transmitted into the bus.

The *MVC* optimizes the supervision for the following reasons:

- One single *MVC* can contain parameters of different function blocks from the same device.
- The analyzer system does not have to send a request and wait for the answer, because the device sends the message without needing a request.

A *MVC* is easily detected. Use a filter to capture the *DT1* and set the destination address to 0140h. In this case, the filters were:

- D30140
- D70140

4. FBVIEW - HSE

4.1 Selecting the Communication Interface

After creating the messages file for the HSE communication, **FBView** automatically initializes the communication, searching for the communication interfaces available in the network.

If the PC executing **FBView** has more than one network adapter, click the button **Interface**,  , to open the dialog box and select the adapter that will be used to capture the frames.

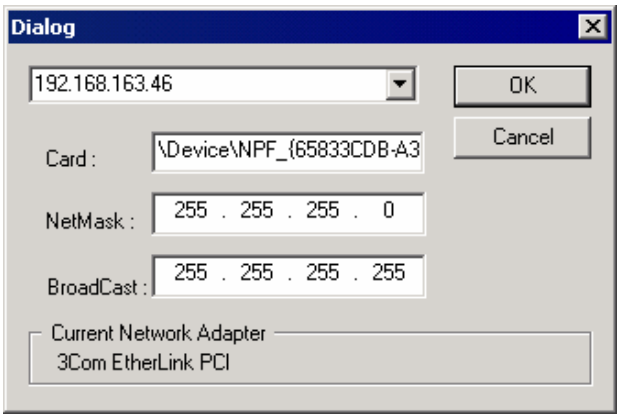


Figure 4.1. HSE Communication Interfaces

Click the button **Start**,  , to start capturing the messages from the bus.

The figure below shows the messages in the HSE network:

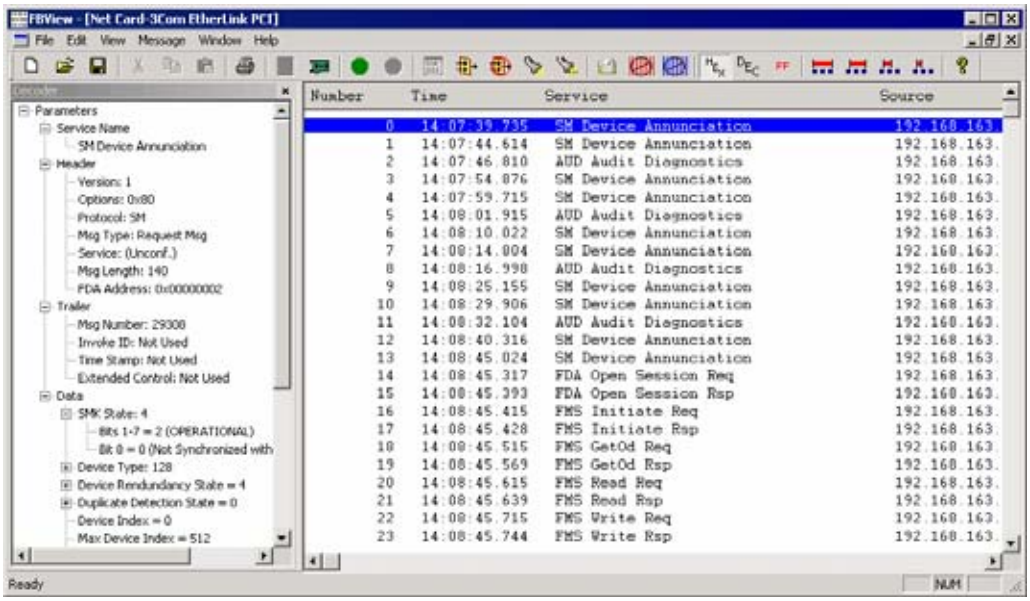


Figure 4.2. FBView Interface

4.2 Filters

4.2.1 Adding Filters

To configure a filter, click the button **Enable Filters**,  , in the toolbar. The *Filter Configuration* window showed in the figure below will open:

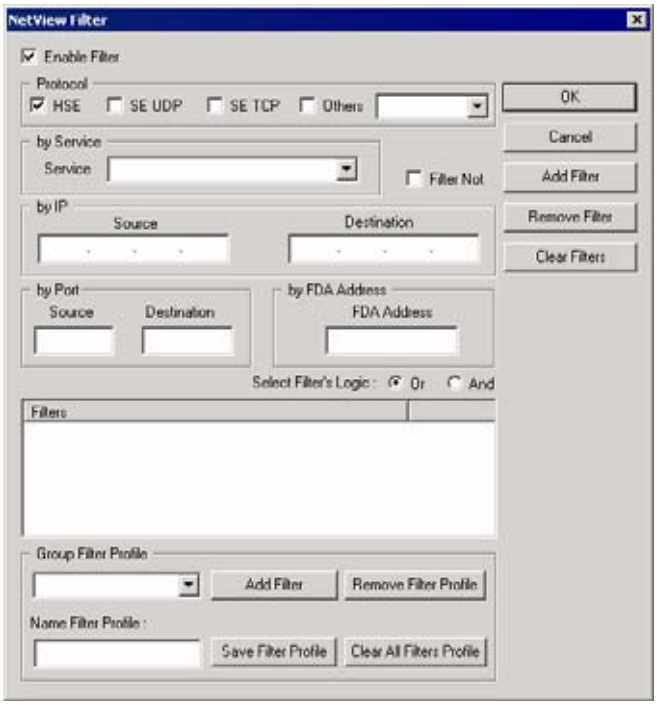
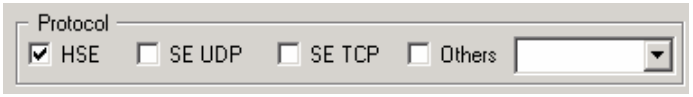


Figure 4.3. Filters Configuration

Select the option *Enable Filter* to apply the filters to the messages.

In the field *Protocol*, select the communication protocol for the filter:



- **HSE**: capture the HSE frames defined in the documentation FDA Agent (FF-588) FS 1.09;
- **SE UDP/SE TCP**: capture the frames from the Smar Ethernet protocol;
- **Others**: filters the frames from a log file of the HSE, DFI or ModBus.

In the field *By Service*, select the type of the service related to the message:

The list with the services depends on the protocol selected in the field *Protocol*. The service showed in the figure above was selected in the HSE protocol.

Mark the option *Filter Not* to filter the messages that do not have the selected service.

Click the button **Add Filter** to add the filter to the *Filters* list. See the example below:

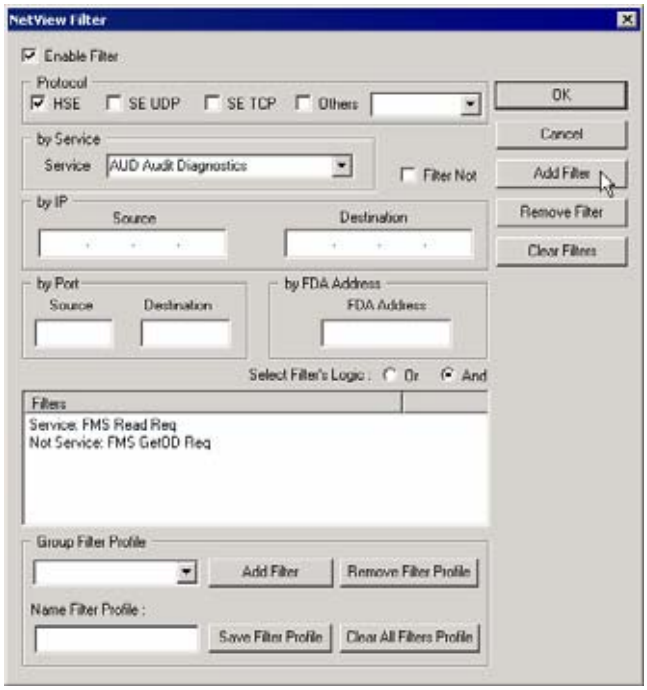


Figure 4.4. Creanting a Filter by Service

In the field *By IP*, type the IP address of the computer from where the messages are being sent (filed *Source*) to filter the received messages, or the IP address of the computer receiving the messages (field *Destination*):

Click the button **Add Filter** to add the filter to the *Filters* list. See the example below:

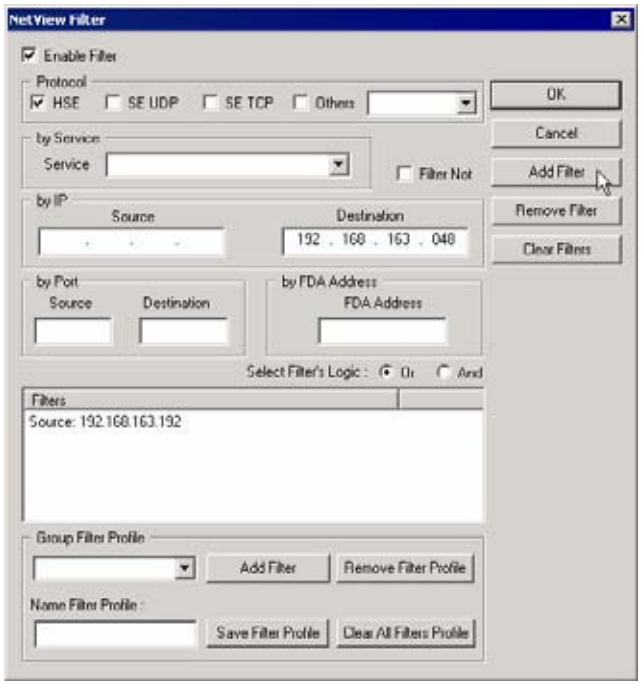
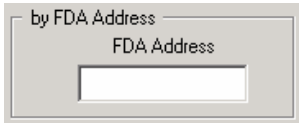


Figure 4.5. Creanting a Filter by IP Address

In the field *By Port*, type the number of the communication port to filter the messages being received (filed *Source*) or being sent (field *Destination*) through that port:

Click the button **Add Filter** to add the filter to the *Filters* list.

In the field *By FDA Address*, type the FDA address to filter the received messages:



Click the button **Add Filter** to add the filter to the *Filters* list.

If the user add two or more filters to the list of filters, select the logic operation:

- **Or**: applyes the logic "or" when capturing the frames - the message must fill at least one criterion defined by the filters to be captured.
- **And**: applyes the logic "and" when capturing the frames - the message must fill all criteria defined by the filters to be captured.

4.2.2 Removing Filters

To remove a filter, select the specification in the *Filters* list and click the button **Remove Filter**. See the example below:

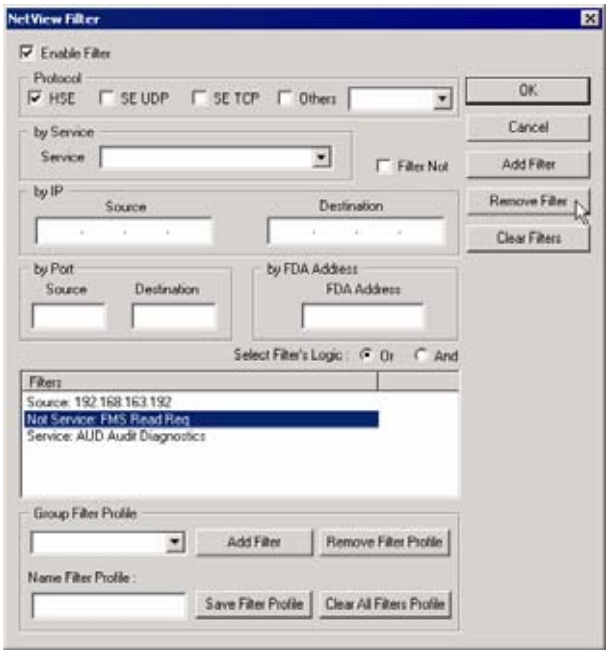


Figure 4.6. Removing a Filter

To remove all filters, click the button **Remove All Filters**.

4.2.3 Creating a Filter Profile

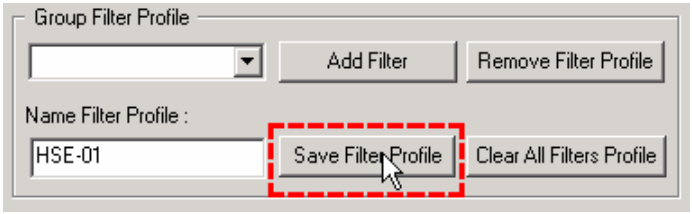
The *Filter Profile* allows the user to save all filters configured with specific names related to their function.

The user can create a library with configured filters, being able to add/remove one or all filters from the library.

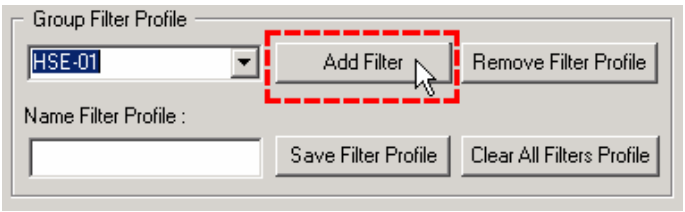
To create a filter profile:

1. Click the button **Enable Filters**,  , in the toolbar, to open the *Filter Configuration* window.
2. Add the filters as indicated in the figure below.

3. Type a name for the filter profile and click the button **Save Filter Profile**.



After adding a new filter profile, it will be displayed in the *Group Filter Profile* menu. Use the button **Add Filter** to add the selected filter profile to a group of filters from another message buffer.



4.2.4 Removing Profiles

To remove a filter profile, select the name of the profile in the *Group Filter Profile* list and click the button **Remove Filter Profile**. See the example below:

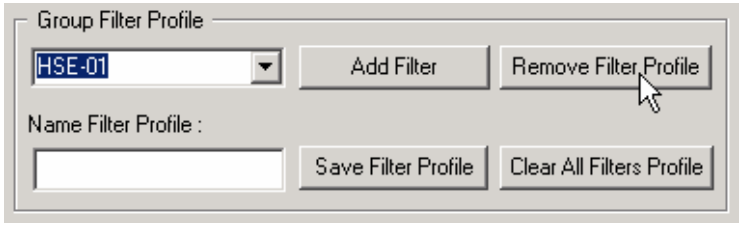


Figure 4.7. Removing a Filter Profile

To remove all filter profiles, click the button **Remove All Filters Profile**.

4.3 Message De-codification in the HSE Mode

To decode a message captured in the HSE mode, click the message in the list. The details of the selected message will be displayed in the *Decoder* window. See the example below:

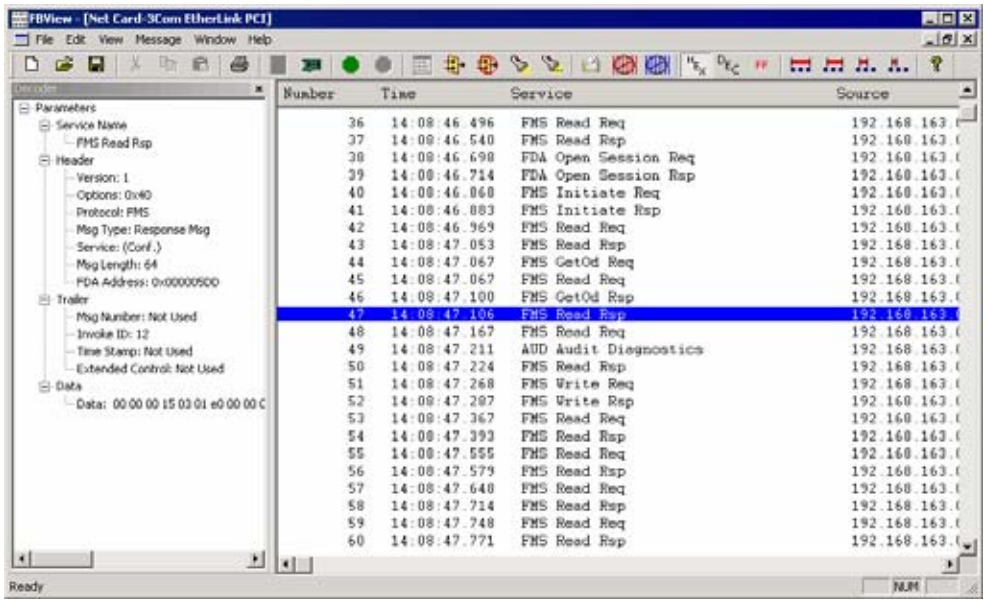


Figure 4.8. Displaying Message Information

The figure above shows the interpretation of a HSE message, type *FMS Read Rsp*. A HSE message is composed by three sets of information: *Header*, *Trailer* and *Data*.

The *Header* contains all common and mandatory fields for each HSE message.

The *Trailer* contains optional fields for HSE messages.

The *Data* contains object data, therefore the length and the content of this field depends on the object that the message is accessing. This field doesn't appear in all HSE messages. If a message doesn't have an associated object or error, then this field is not necessary.

Every message has the type of the service, that can *request*, *response* or *error*, and an option that indicates whether the message requires a confirmation or not.

It is important to notice that some messages have reserved fields, that must have value 0 (zero).

Header Details

Version: specifies the version number of the message. Currently in version 1.

Option: each bit in this field has a meaning, described below:

Bit 8:	message number is in the trailer.
Bit 7:	the *invoke ID* is in the trailer. (for client/server sessions, this bit will always be 1)
Bit 6:	*Time Stamp* is in the trailer.
Bit 5:	reserved.
Bit 4:	extended control field is in the trailer.
Bits 1-3:	number of bytes sent between the trailer and data to align the message.

Protocol ID and Confirmation Msg: bits 3 to 8 indicate the protocol of the message, that can be: *FDA Session Management*, *SM*, *FMS* or *Lan Redundancy*. Bits 1 to 3 indicate the type of the message: *request*, *response* or *error*.

Service: bit 8 indicates if a confirmation was requested, and bits 1 to 7 show the identification of the service.

Message Length: indicates the total number of bytes in the message, including the *Header* and the *Trailer*.

FDA Address: the usage of this field depends on the type of the message and the VCR used. For further information, see table 14 in norm FF-588 FS 1.3 of the *Fieldbus Foundation* (*Field Device Access Agent*).

Trailer Details

Message Number: sequential number of the messages transmitted by a specific VCR.

Invoke ID: identification of request and response. This means it is used to associate a response to a request, and vice versa.

Time Stamp: indicates the time, in the system, that the message was created. The device receiving the message can use this field to determine the transmission time of the message.

Extended Control Field: to be used in the future.

For some HSE-type services, such as *FMS Read Rsp*, *FMS Write Req* and *FMS Get OD Rsp*, it is possible to decode the parameter *Data*, since this parameter has several types of data structure.

For example, to decode the HSE service named *FMS Read Rsp*, showed in the figure below, double right-click the parameter *Data*. The popup menu will open showing the list of options to decode the service.

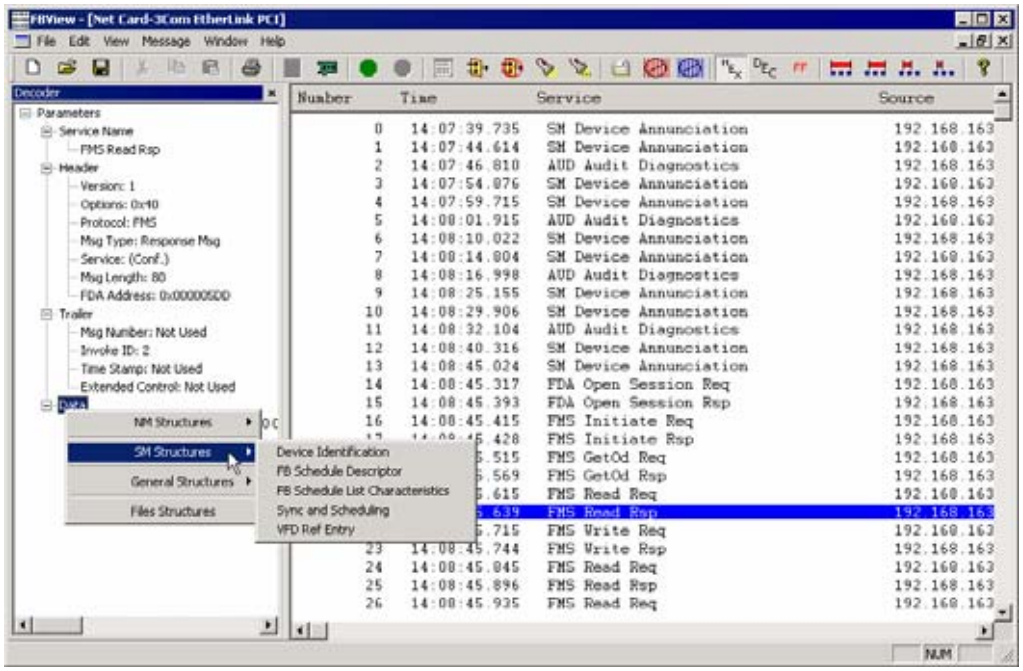


Figure 4.9. Decoding the HSE Service

The user must know which type of data structure is included in the parameter *Data* of the service *FMS Read Rsp* (in the example above, the user has the *index* information sent in the service *FMS Read Req*), to select the correct data structure in the options menu.

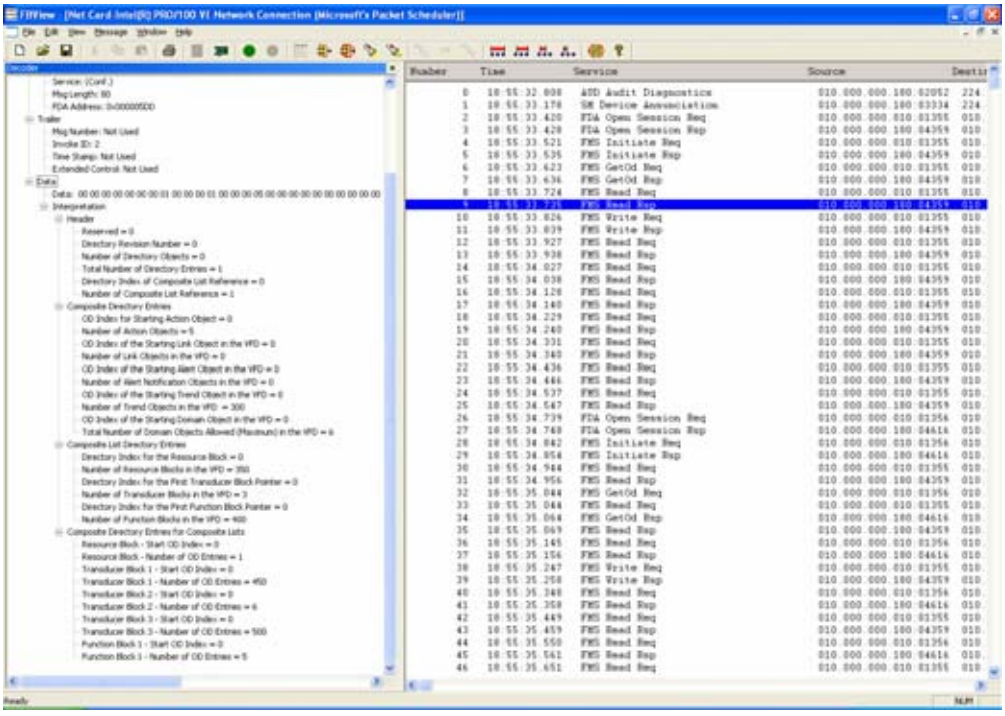


Figure 4.10. Service De-codification

Figure 4.10 shows the de-codification for reading the directory of the VFD application.

4.4 Importing Log Files

To import a log file generated by **Smar NetView**, click the button **New**,  , in the toolbar, to create a new file and select the HSE communication.

Go to the *File* menu and click the option *Import NetView Old Version Files*. The *Open* dialog box will appear. Select the log file and click *Open*.

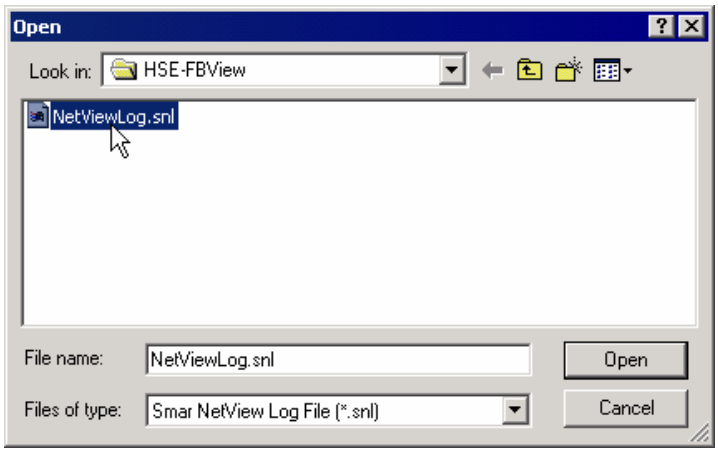


Figure 4.11. Importing NetView Log Files