FarSync Flex Monitor User Guide

1. Introduction

The *FarSync Flex* not only provides support for access to Wide Area Network resources (e.g. via TCP/IP) but also can be run in a passive (external) monitoring mode. In this case, the device is passively connected to the network and can capture both data and signal information from the network connection in real-time. The data can be displayed by monitoring application such as the FarSync Line Monitor application provided, or the standard Wireshark application. Alternatively, a sample Windows application is provided in the FarSync SDK that demonstrates how easy it is to programmatically capture data and signal information via standard Win32 I/O commands.

The *FarSync Flex* supports monitoring in the following interface modes

- V.24
- X.21
- RS530/449

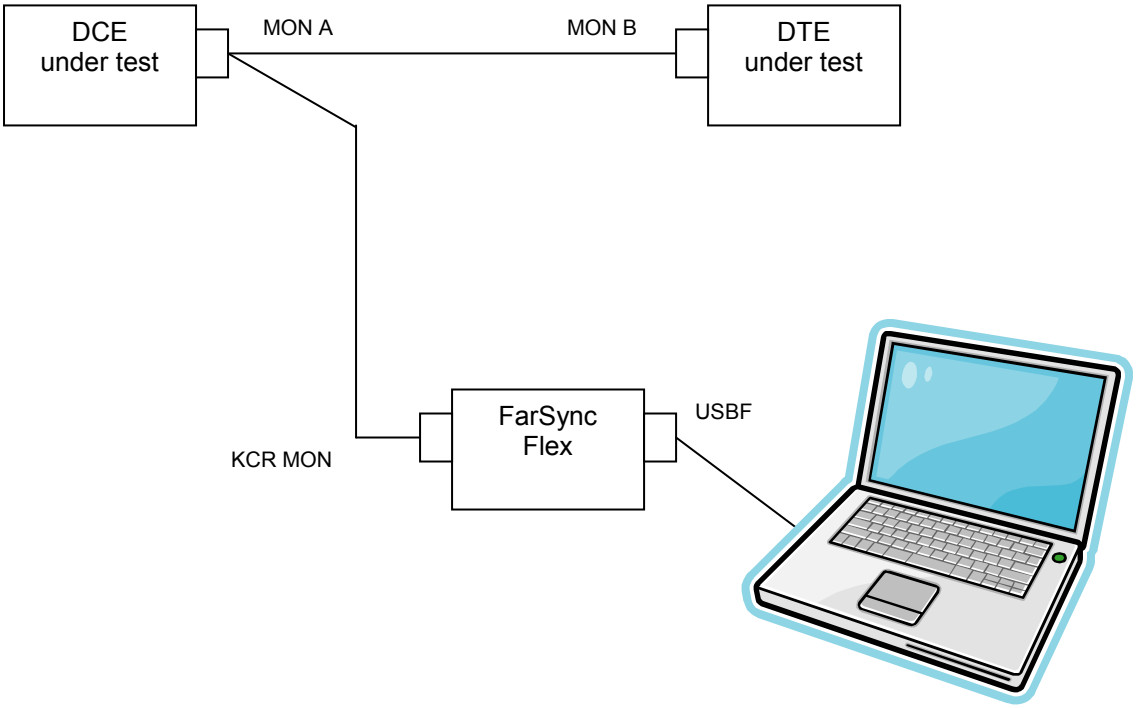
Line rates of up to 2Mbps are supported.

2. Test Environment

When the *FarSync Flex* is used to monitor an external line, a special monitor cable is required. The type of cable to be used is dependant upon the interface type being monitored. Two cable types are available:

KCR-MON – used for V.24/RS530
KCX-MON – used for X.21

The cable connects the test environment together as follows:



There are two ways of using the passive monitoring support offered by the *FarSync Flex:*

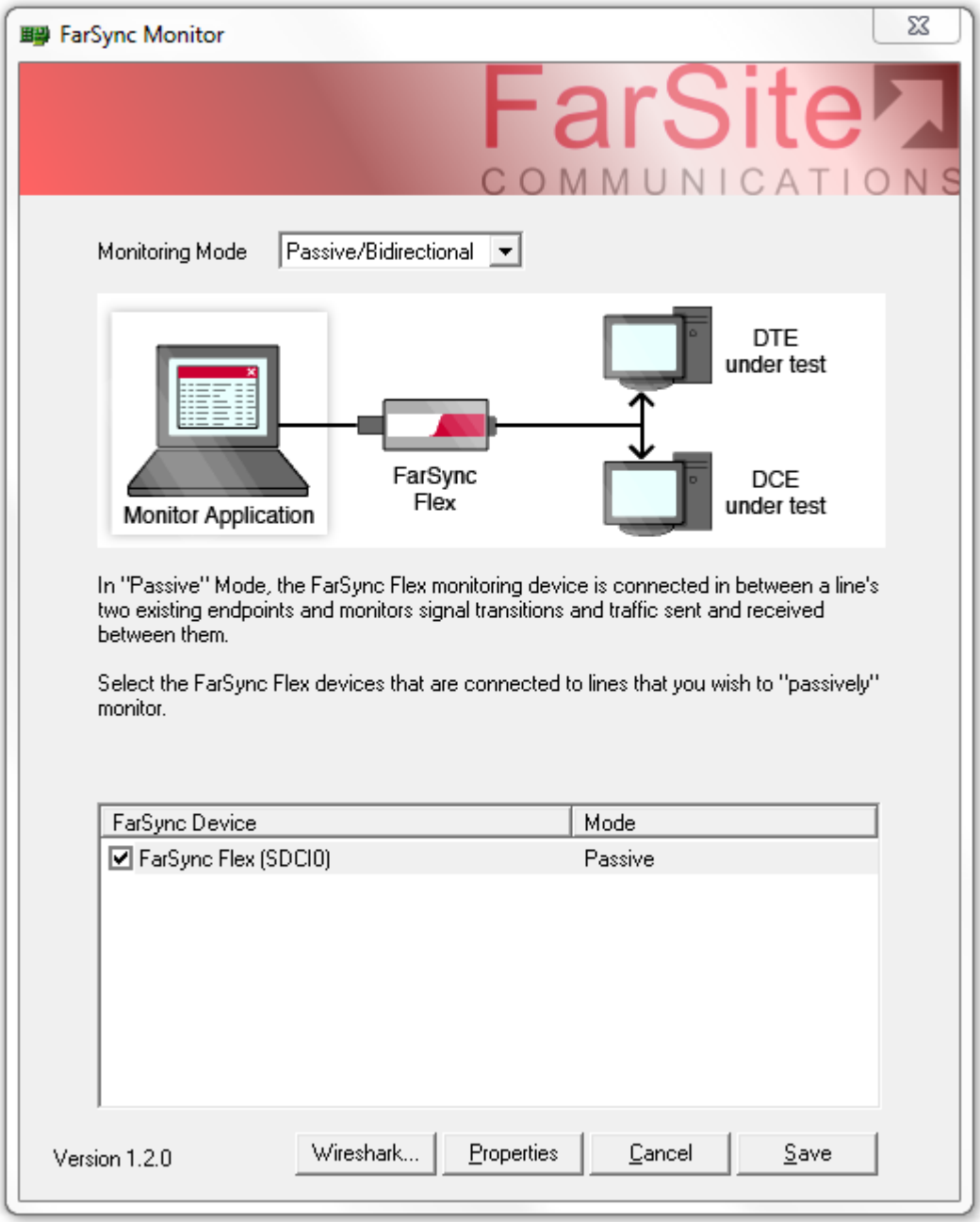
1) a plug-and-play/turnkey solution using a simple graphical user interface-based application, *FarSync* Line Monitor (*fsmon*)
2) programmatically using the Windows-based *FarSync Flex* API.

3. Plug-and-Play Passive Monitoring

Once the *FarSync Flex* drivers are installed, you should next install the *FarSync* Line Monitor component.

Next, connect the monitor cable into the system under test as shown in the previous section.

Run the FarSync Line Monitor from the Windows Start menu and select File – Recording Mode… This will display the configuration dialog shown below.



Select the Passive/Bidirectional monitoring mode.

Now select the *FarSync Flex* being used for monitoring:

The settings for the monitoring port may be configured via the properties button. Note that the clocking setting for a port used for passive monitoring is always treated as “external” regardless of the setting shown in the port’s properties dialog.

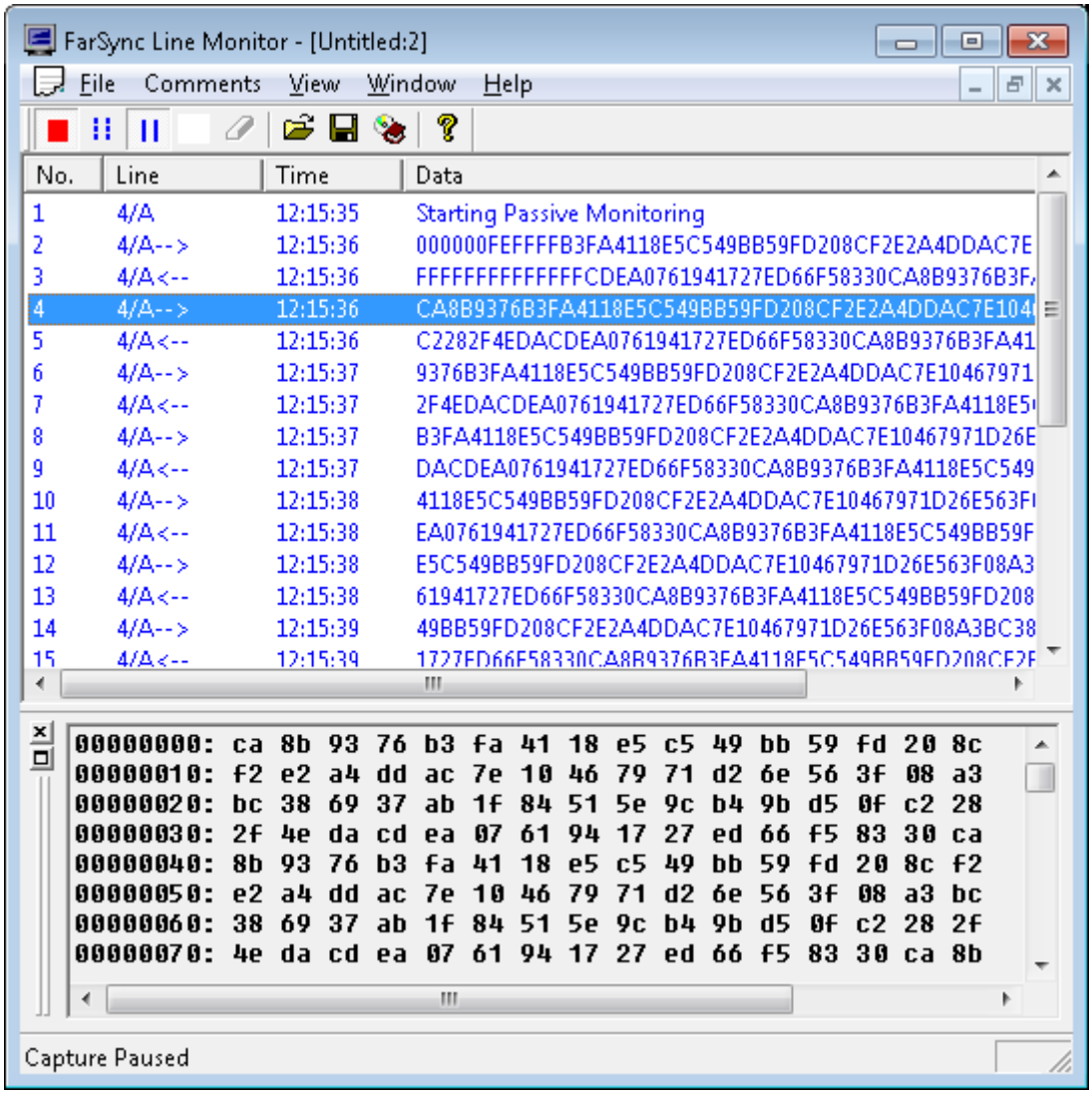
It is important to ensure that you configure the port’s Interface, Mode and Encoding settings to match the line being monitored e.g. if you were monitoring an X.25 or HDLC line then the monitoring port’s Mode parameter would have to be set to HDLC whereas if you were monitoring a line carrying a transparent bitstream, e.g. when running a Bit Error Rate test, then the Mode should be set to Transparent.

Then press “Save”.

Press the FarSync Line Monitor’s Record button twice to restart capturing using the new configuration to starting viewing the network traffic in real-time – this includes both data and control signal information.

Data flowing in the DCE to DTE (MON A to MON B) direction is denoted by a “→” and DTE to DCE (MON B to MON A) by a “←”.

RTS and DTR signals originate from the DTE (MON B), CTS and DCD signals originate from the DCE (MON A).



Note that real-time monitoring at high speed can take a large amount of CPU time just refreshing and scrolling the display. To minimize this effect, the display can be frozen, without affecting data capture. To select this feature, press the  button.

Note that the *FarSync* Line Monitor supports **both** external (passive) and internal monitoring. The latter is used where the Flex is being run in access mode i.e. it is the actual DTE/DCE of the network link rather than being connected passively into a line using a KCR-MON/KCX-MON cable.

Signal Support

The following signals can be monitored by *FarSync Flex*:

Interface Mode	Supported Signals
V.24	DTR, RTS, CTS, DCD
X.21	INDICATE
RS530/449	CTS, DCD

Decode Support

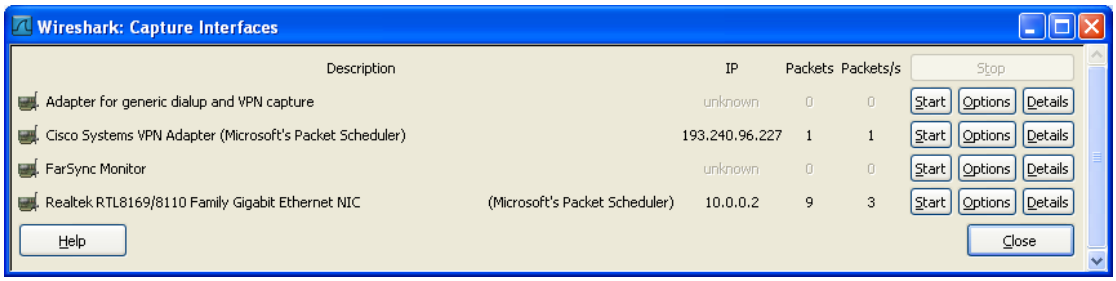
The *FarSync* Line Monitor decodes both PPP and X.25 packets/frames. The payload of PPP/X.25 data packets is not decoded. Instead these are displayed by *fsmon* as raw hex/ASCII dumps. However, the data captured by *fsmon* can be saved as a file in libpcap format and subsequently read, for instance, by 3rd party monitor applications, e.g. Wireshark which does provide decoding of the payload portion of the packets as well.

Wireshark Run-Time Support

It is also possible to use the 32-bit version of Wireshark instead of the *FarSync* Line Monitor to display/decode the PPP/X.25 network traffic in real-time. This is supported in both 32-bit and 64-bit versions of Windows.

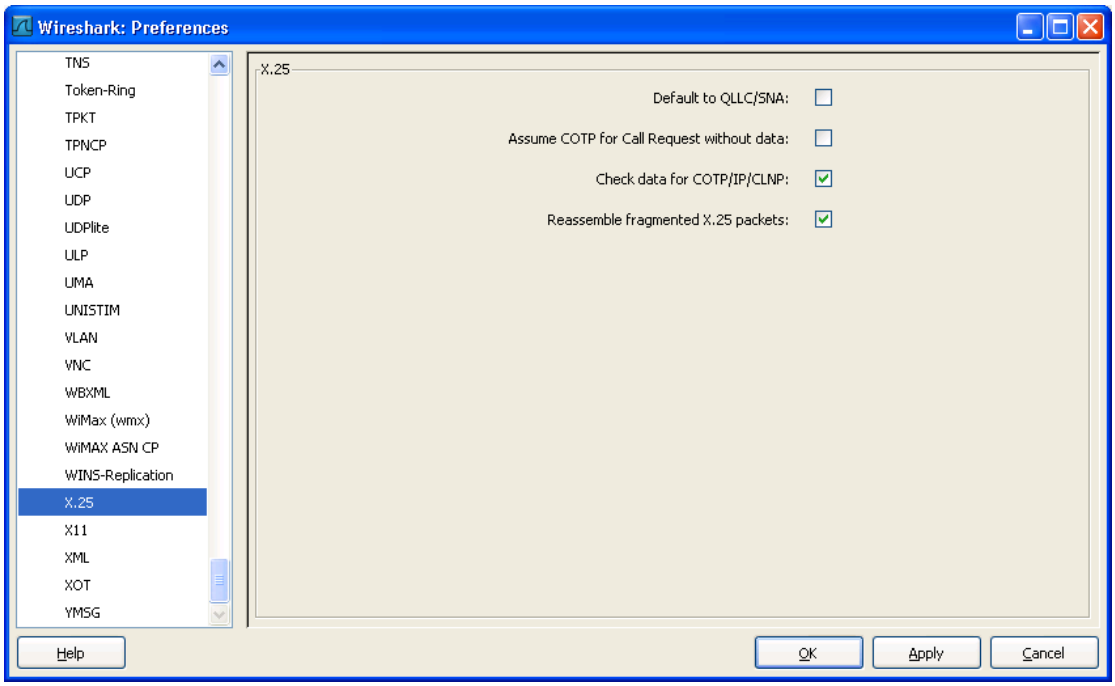
The version of Wireshark (32-bit) currently supported is 1.10.0. The following steps enable Wireshark to display data captured by a *FarSync Flex* which is running in passive monitoring mode:

1) install Wireshark (1.10.0)
2) install the *FarSync* Line Monitor
3) use FarSync Line Monitor's File - Recording Mode menu item to display the monitor configuration dialog
4) configure the monitoring mode to Passive/Bidirectional and use the Wireshark button to add FarSync support to Wireshark
5) close FarSync Line Monitor
6) run Wireshark (in versions of Windows Vista and later this must be **run as administrator**) and select Capture-Interfaces
7) select the "FarSync Monitor" interface to start capturing network traffic being monitored at the *FarSync Flex*

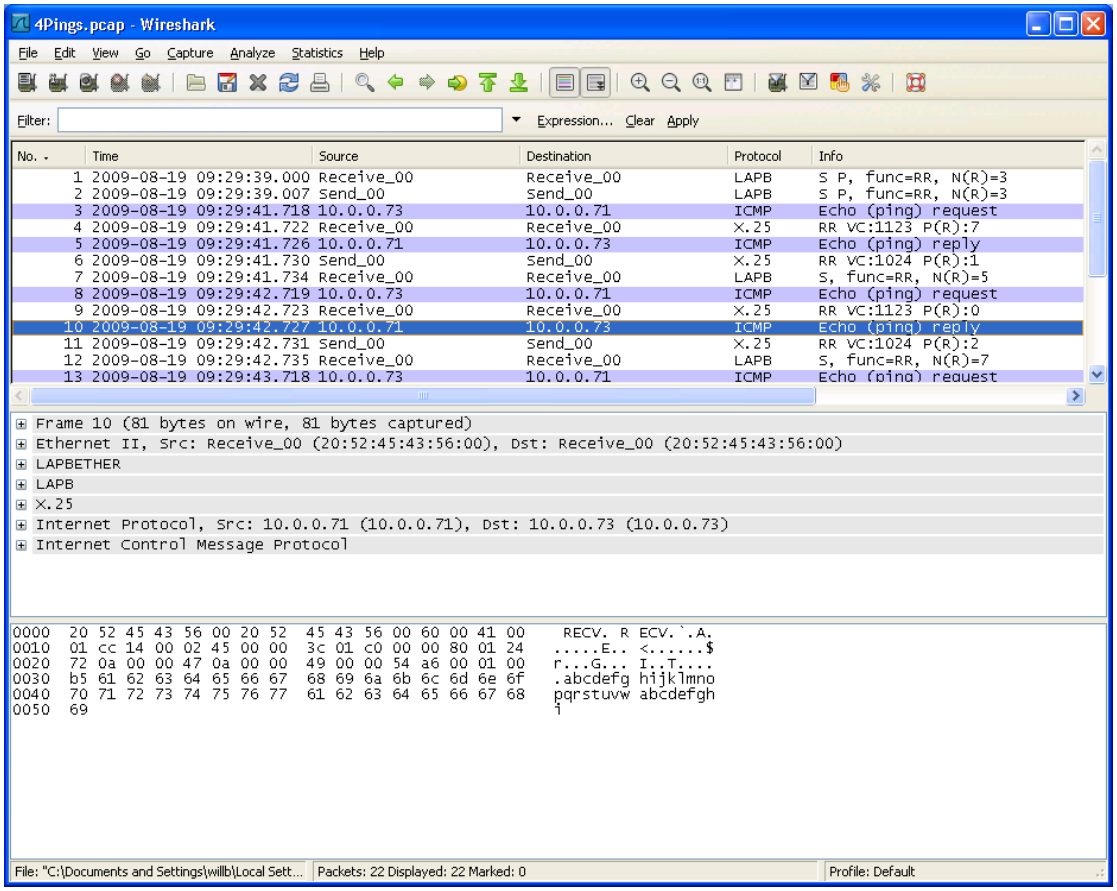


Wireshark Decode Support

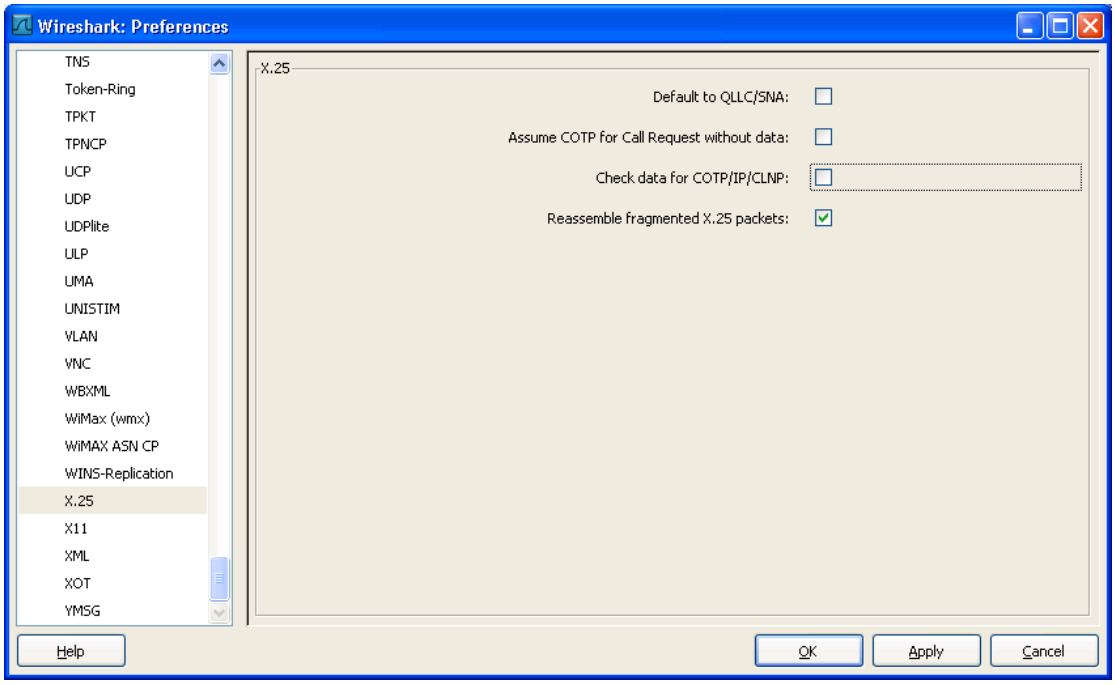
Wireshark offers a wide range of protocol decoders; you should refer to the Wireshark documentation for more information regarding these. However, it is particularly important to note that, the protocol-specific decode options can be found under Edit-Preferences-Protocols; then select the protocol of interest e.g. in the case of X.25:



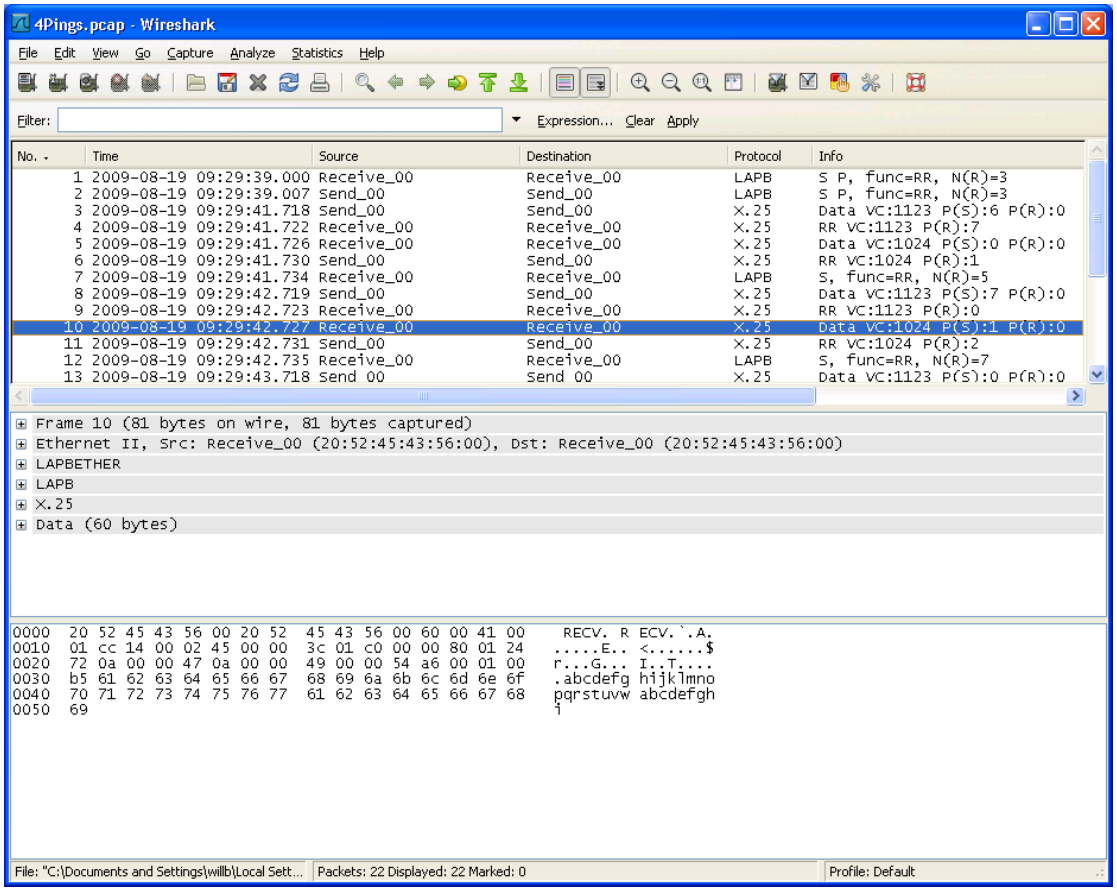
having **Check data for COTP/IP/CLNP** will instruct the decoder to continue further level decoding of the X.25 payload e.g.



Without that option enabled i.e.



the X.25 payload itself is not decoded e.g.



Port Identification

When multiple FarSync card/devices/ports are in use they can be identified using the Source and Destination fields as displayed by Wireshark. These values correspond to the pseudo MAC addresses associated with data sent/received on serial ports. The identification scheme used is as follows:

Direction

Data sent from a FarSync port have **both** Source and Destination fields set to **Send_**XX. Data received by a FarSync port have **both** Source and Destination fields set to **Receive_**XX.

Card/Device

The FarSync card/device instance/number is identified by the numerical value shown in the Source field e.g. Send_00 represents a frame sent via FarSync card/device #0; Receive_04 represents a frame received by FarSync card/device #4.

Port

The FarSync port is identified by the Destination field e.g. Send_0a represents a frame sent via port A; Receive_0c represents a frame received on port C.

4. Programmatic Passive Monitoring

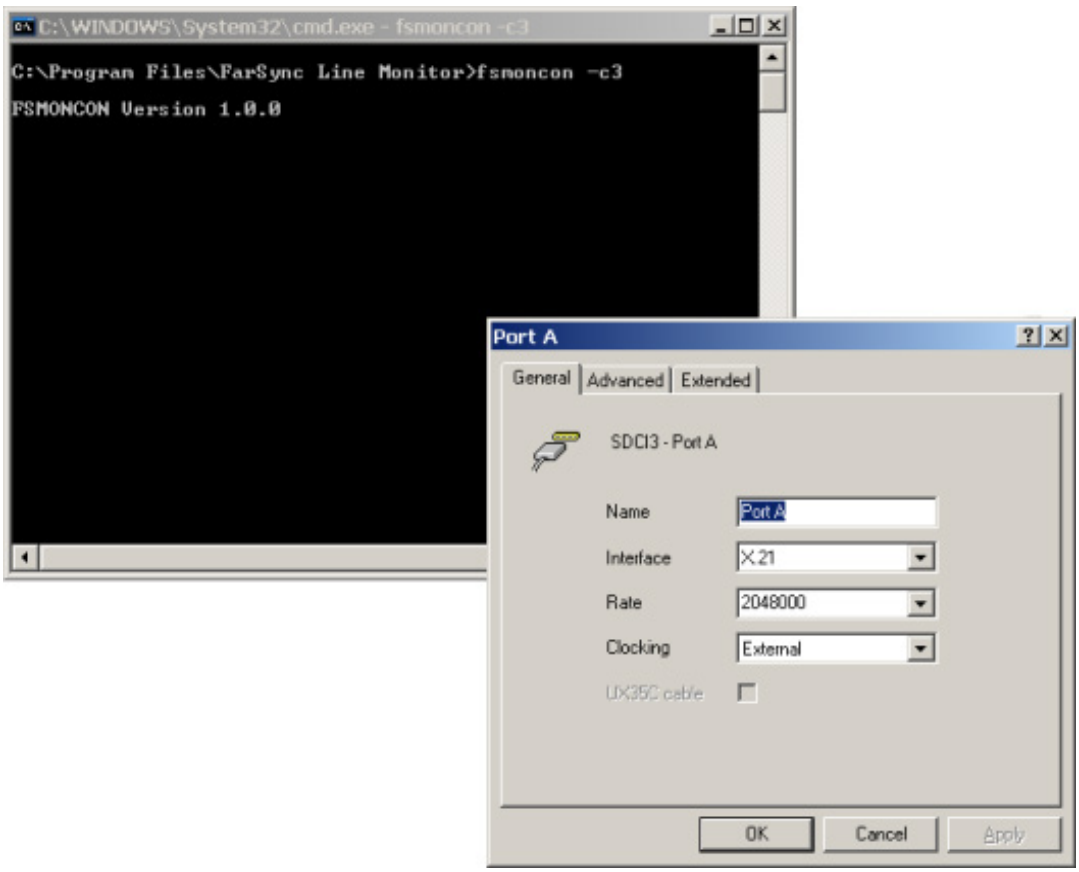
In order to demonstrate how a developer can programmatically configure and control the *Flex* in passive monitor mode, *FarSite* provides a sample application, *fsmoncon*. This application is included in the FarSync SDK

The command-line *fsmoncon* application is pre-built and can be run as is (from \Program Files\FarSync Line Monitor\fsmoncon). In order to specify a specific device for monitoring, the program can be started from the command-line by entering, for instance:

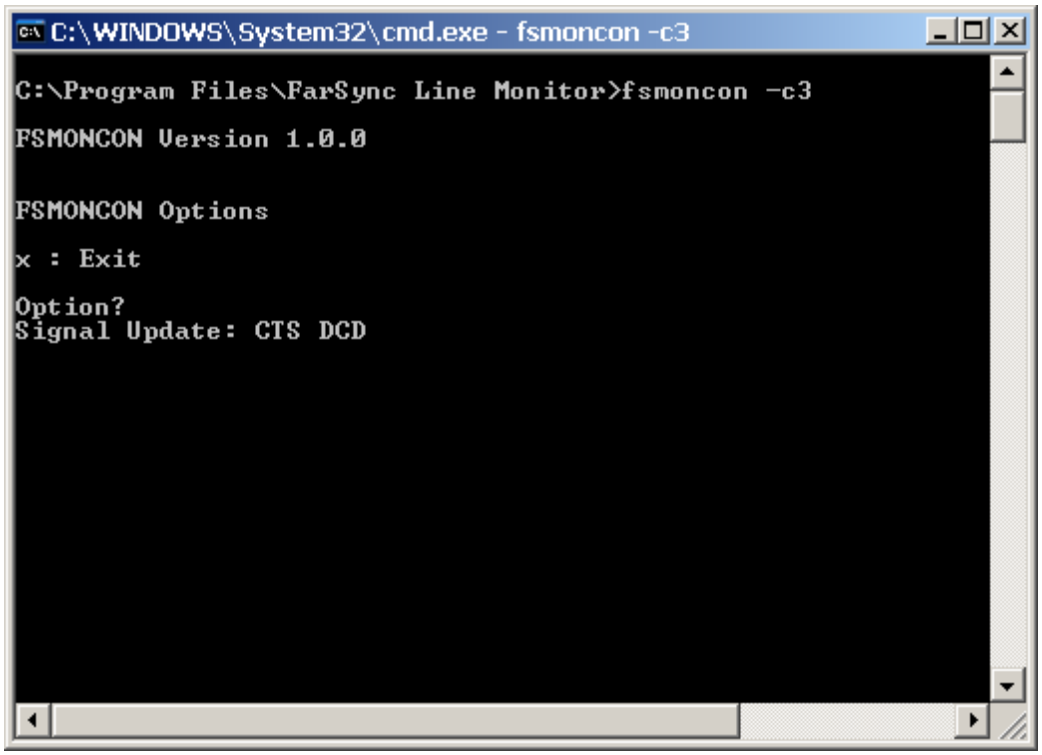
fsmoncon –c3

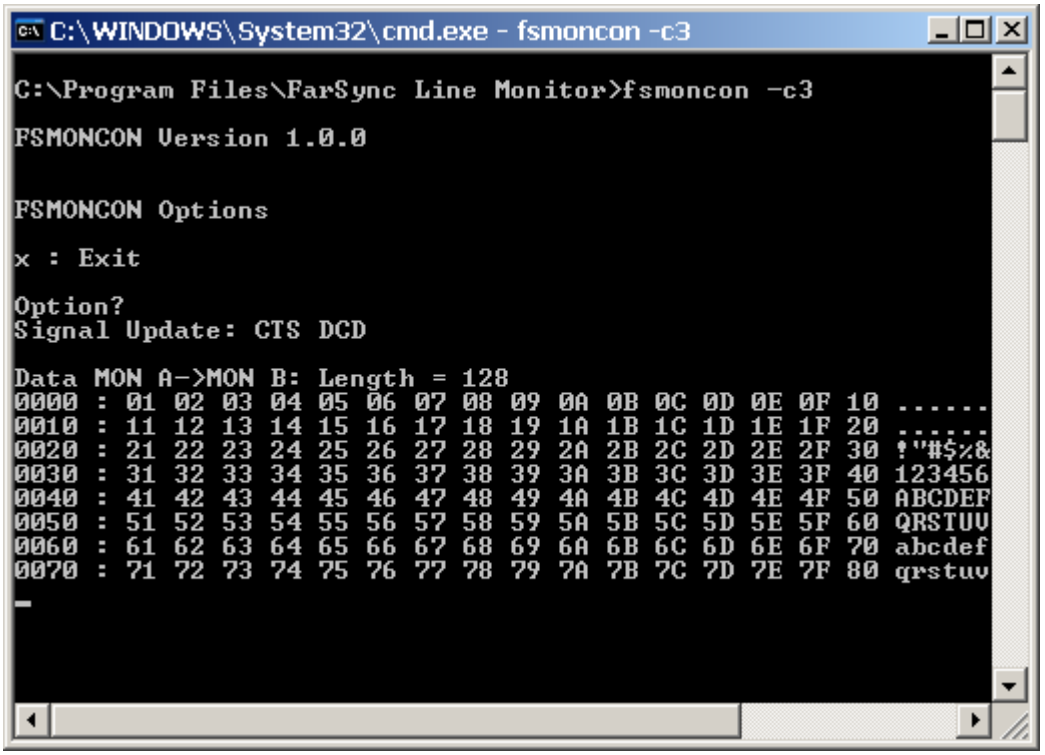
This would make use of the *Flex* identified as SDCI3 instead of the default SDCI0.

When first opened, the application prompts the user to configure the *Flex* appropriately. Check the interface type is correct for the device under test and that the clocking mode is set to external.



Once the application is running it will trace transmit and receive data and indicate when signal states have changed.





FarSite provides full source for the *fsmoncon* test application in a Visual Studio Application. It should provide a good base for any developers wishing to add *FarSync Flex* monitoring support to their own projects. This can, for example, enable developers to carry out their integration of the Flex with other 3rd party monitoring applications.

Monitoring API

As demonstrated by the *fsmoncon* sample application, the passive monitoring support of the *FarSync Flex* is achieved on Windows using the *FarSync* Win32-based interface (FsWinAPI). Details of this API are available in the FsWinAPI User Guide available in the FarSync SDK.

Note that when running in monitor mode, the *Flex* can be considered, as far as the client application is concerned, as having two ports, Port A and Port B.

The following table illustrates how the signals of the network connection are connected to the *Flex*:

DCE/DTE Signal	Signal Flow	Flex Signal
TxD	DTE ➔ DCE	Port B RxD
RxD	DCE ➔ DTE	Port A RxD
RTS	DTE ➔ DCE	Port B CTS [1]
CTS	DCE ➔ DTE	Port A CTS
DCD	DCE ➔ DTE	Port A DCD
TxC	DCE ➔ DTE	Port A TxC *
RxC	DCE ➔ DTE	not used
DTR	DTE ➔ DCE	Port B DCD [1]

All captured frames are passed up to the client application via the handle associated with PortA. However, each frame is prefixed with a FSCMN_TXRX_PREAMBLE header which can be examined to determine the direction of the frame (i.e. tx or rx.). Signal changes and errors can be reported on either port.

In async mode, the Flex can be configured to capture additional information along with the rx data. Using fsmoncon's –m option enables you to choose between:

```
-m0 ==> FSCMN_ASYNC_RX_CHAR
[rx buffer contains rx chars only]

-m1 ==> FSCMN_ASYNC_RX_STATUS_CHAR
[rx buffer contains LSR+char pairs for each rx event]

-m2 ==> FSCMN_ASYNC_RX_STATUS_CHAR_TIME
[rx buffer contains LSR+char+4-byte field containing a 16-bit
timestamp for each rx event]

The timestamp field is in little-endian format.

Note that the LSR format is as follows: [VAL|CD |RI
|CTS|BRK|FRM|PAR|OVR] (VAL indicates whether the associated 'char' is
valid or not)
```

fsmoncon uses the SetSerialConfig IOCTL to configure the AsyncReceiveMode option – see the FsWinAPI User Guide for more information regarding use of the SetSerialConfig IOCTL.

Please refer to the fsmoncon sample for more information.

* All timing is relative to this clock in monitor mode, for all interface types
[1] RTS and DTR (DCE/DTE Signal) are currently not supported in X.21 and RS530 modes.