# **Ellisys USB Analysis Software**

## **User Manual**

🔹 🐁 🏷 🚅 🚅 🖬 🗠 🗐 Tem	⊙ 🚁 🗳 Device	Endpoint		Status	Time		^	Details						ņ
nter text here	Y E Y		Ent Y			. 7					,		09.38	-
GetDescriptor (Device)	0(1)	0		ок	0.060 153		_			age (Wheel				_
SetAddress (1)	0(1)	0		OK	0.082 650				Log	gical minim	um (-127)		15 81	
GetDescriptor (Device)	1	0		OK	0.145 161				Log	gical maxim	um (127)		25 7F	
GetDescriptor (Configuration)	1	0		ок	0.146 467	117			Re	port Size (8	·)		75.08	
GetDescriptor (Configuration)	1	0		ок	0.147 536	450							95.03	-
GetDescriptor (String lang IDs)	1	0		ОК	0.149 581	450			ке	port Count	(3)			_
GetDescriptor (String iProduct)	1	0		ок	0.150 464	950			In	put (Data,V	'alue,Relative,Bit	Field)	81 06	
GetDescriptor (String lang IDs)	1	0		OK	0.152 478	450			End (	Collection			C0	
GetDescriptor (String iProduct)	1	0		OK	0.153 463	783		E	nd Col	lection			CO	
🙀 GetDescriptor (Device)	1	0		ок	0.161 209	450				lection			00	
GetDescriptor (Configuration)	1	0		OK	0.162 484	283								
GetDescriptor (Configuration)	1	0		ок	0.163 464			1	nput R	eport 0				
SetConfiguration (1)	1	0		ок	0.165 462			Bit of	ffset	Bit count	Descriptio	n		
🐚 SetIdle (All, Indefinite)	1	0	0	ОК	0.192 100			0		1	Button 1			
GetDescriptor (Report)	1	0	0	ОК	0.193 540			1		1	Dutton 2			-
E Left button click					4.456 816			-		-				_
Dinput Report	1	1	0	OK	5.464 885 5.584 893			2		1	Button 3			
Input Report     Right button click	1	1	0	ОК	5.584 893			3		5	<li>(Not used)</li>			
Right button dick           Imput Report	1	1	0	ок	14.377 493			8		8	€ x			
The second	1	1	0	OK OK	14.377 493		~				7			_
mmary							Ψ×	16		8	ФY			
,							-	24		8	🗼 Wheel			
🕲 🕲 Find next 🛛 🙀														
tem			Displaye	3		Total	^	🗾 Deta	ils 🔎	Search   🎯 B	export   🚱 Options			
🖶 Transactions		Count	Byte	5 0	ount	Bytes		Data						
Σ Total		3'040	592 byte	5 3	3'040	592 bytes			0 :	1 2 3	4 5 6 7	01234567		_
🖃 🧼 By Device Address								0:	D5 0		A1 01 09 01			
Addr 0		13	34 byte		13	34 bytes	<b>E</b>				19 01 29 03	).		
<ul> <li>Addr 0</li> <li>Addr 1</li> </ul>		3'027	558 byte		3'027	558 bytes					95 03 75 01	*u.		
By Endpoint		3027	550 Dyte		5527	550 bytes					75 05 81 03 09 31 09 38	u 0.1.8		
•											75 08 95 03	*[u		
Addr 0 - 0 IN		10	18 byte		10	18 bytes		48:	81 0	6 CO CO				
Addr 0 - 0 OUT		3	16 byte	5	3	16 bytes								
Addr 1 - 0 IN		109	254 byte	5	109	254 bytes								
Addr 1 - 0 OUT		24	104 byte		24	104 bytes	~							

Version 3.0.2 May 30, 2013



Ellisys • Chemin du Grand-Puits 38 • CH-1217 Meyrin Geneva • Switzerland www.ellisys.com • support@ellisys.com

## **Table of Contents**

Chapter	1: Software overview	6
	Overview	6
	Dockable panes	6
	Toolbar	8
	Protocol display	9
	Grouping of consecutive NAKs and SOFs	10
	Details display	11
	Color-coding transactions	13
	Options	13
Chapter	2: Recording options	15
	General tab	15
	Filter tab	
	Trigger tab	17
		• •
Chapter	3: Real-time Monitoring	30
	Overview	30
	General tab	31
	Devices tab	32
	Frames tab	32
Chapter	4: Protocol Summary	34
	Introduction	34
	Categories	35
	Find Summarized Items	35
	Total and Displayed items Summary	35
	Report export	36
Chapter	5: Searching and Post-capture Filtering	37
	Search	37
	Instant filters	43
Chapter	6: Data Export	45
	Introduction	45
	Exporting USB transactions	46
	Exporting endpoints data	52
	Exporting Setup requests	53
	Exporting descriptors	54

	Exporting generator scripts	56
Chapter	7: Software and drivers installation5	8
	Install or update the analysis software	58
	Automatically install or update the device driver	58
	Manually install or update the device driver	58
	Install an optional functionality	53
Chapter	8: Frequently Asked Questions	7
Chapter	9: Troubleshooting	9

# Table of Figures

Figure 1 - Overview of the analysis software
Figure 2 - Grouping of consecutive Start Of Frames10
Figure 3 - Grouping of consecutive NAKed transactions10
Figure 4 - Default display of tables11
Figure 5 – Tables display with full details11
Figure 6 - Tables display with full details and low level values12
Figure 7 - Color-coding of transactions13
Figure 8 - Protocol Summary view
Figure 9 - Protocol Summary view with filtered items
Figure 10 - Search panel
Figure 11 – Search for protocol elements
Figure 12 - Search for USB transactions
Figure 13 - Search for endpoints data40
Figure 14 - Search for setup requests41
Figure 15 - Search for columns strings42
Figure 16 - Search of protocol errors43
Figure 17 - Export panel45
Figure 18 - Export USB transactions
Figure 19 - XML export sample
Figure 20 - Export endpoints data
Figure 21 - Export setup transferts53
Figure 22 - Export descriptors
Figure 22 - Export generator script
Figure 23 - Updating driver from the Device Manager
Figure 24 - Install from a list or specific location
Figure 25 - Search for the best driver61
Figure 26 - Driver installation
Figure 27 – Driver installation completed63
Figure 28 - Icon of the unlocking software
Figure 29 - Extracting compressed software
Figure 30 - Software waiting for an analyzer connection
Figure 31 - Software ready to activate optional functionality
Figure 32 - Wrong serial number detected65
Figure 33 - Optional functionality properly installed

# Chapter 1: Software overview

## Overview

The Ellisys USB analysis software comprises the following elements:

- A list displaying USB protocol elements. This list contains editable fields at the top, called 'Instant Filters', which enable you to filter out elements easily.
- Dockable panes. You may move the panels as you wish in accordance with your preferences.
- A toolbar with shortcuts to the most frequently activated functions.

Item			dp Statu			Time	O Details		
Ivendor	7 E	82	9 E	YE Y	Enter text here	Enter Inv 8	0.0		
Reset (781.5 us)						0.004 789 899	GetDescriptor (Conf	iguration)	
High speed Detection Hands			OK	100		0.005 331 166			2251
GetDescriptor (Device)	0(2)	0	OK	HS	18 bytes (12 01 00 02		Configuration description	tor	8
High speed Detection Hands			OK			0.147 991 349 0.140 715 533	bNumInterface	1	
High speed Detection Hands     SetAddress (2)	0 (2)	0	OK	HS	No data	0.258 124 216	bConfigurationValue	1	
GetDescriptor (Device)	2	0	OK	HS	18 bytes (12 01 00 02	Contraction of the second s		-	
GetDescriptor (Device)     GetDescriptor (Configuration		0	OK	HS	9 bytes (09 02 20 00		bmAttributes.	Not supported	
GetDescriptor (Configuration		0	OK	HS	32 bytes (09 02 20 00		RemoteWakeup		
GetDescriptor (String lang 10		0	OK	HS	4 bytes (04 03 09 04)	0.321 516 216	🐺 bmAttributes. SelfPower	ed No, Bus Powered	
GetDescriptor (String Produ		0	OK	HS	34 bytes (22 03 55 00		1 bMaxPower	400 mA	
I GetDescriptor (String lang ID		0	OK	HS	4 bytes (04 03 09 04)	0.321 962 483	Landard		
+ C GetDescriptor (String Produ		0	OK	HS	34 bytes (22 03 55 00				
GetDescriptor (Device)	2	0	OK	HS	18 bytes (12 01 00 02		Interface descriptor	š.	¥
- GetDescriptor (Configuration		0	OK	HS	32 bytes (09 02 20 00		🤣 bInterfaceNumber	0	
- + SETUP transaction	2	0	ACK	HS	8 bytes (80 06 00 02		Ju bAlternateSetting	0	_
-+ SETUP packet	2	0		HS	0 0)100 (00 00 00 00 m	0.431 971 749		2	
→ DATA0 packet	-	-		HS	8 bytes (80 06 00 02		bNumEndpoints		
+- ACK packet			ACK	HS		0.431 972 583	↓ bInterfaceClass	Vendor-specific	
IF + IN transaction	2	0	NAK	HS	No data	0.431 981 133	-		
E + IN transaction	2	0	ACK	HS	32 bytes (09 02 20 00		Endpoint descriptor		
→ IN packet	2	0		HS		0.432 004 483		1 Particular	
+ DATA1 packet				HS	32 bytes (09 02 20 00	0.432 004 883	bEndpointAddress	1 IN	
→ ACK packet			ACK	HS		0.432 005 816	Details 🐨 Export 🕞 Optio	is 🔑 Search	
E > OUT transaction	2	0	ACK	HS	No data	0.432 014 983	Data		
SetConfiguration (1)	2	0	OK	HS	No data	0.432 253 533	(10000)		
	2	1	ACK	HS	4 bytes (8A F7 8F EF)	0.928 534 083	0 1 2 3 4		012345
E + IN transaction (478)	2	1	NAK	HS	No data	0.929 535 050		00 00 07 05 81 03	
	2	1	ACK	HS	4 bytes (82 F7 0F EF)	1.407 567 049	22: 04 00 04 07 05		
IN transaction (160)	2	1	NAK	HS	No data	1.408 566 949			
⊕ ←; IN transaction	2	1	ACK	HS	4 bytes (8A F7 8F EF)	1.568 577 983			
	2	1	NAK	HS	No data	1,569 570 099			
📧 🗱 ClearFeature (Endpoint 2 IN	H 2	0	OK	HS	No data	2.224 299 533			
🗉 🛶 🛛 IN transaction	2	1	NAK	HS	No data	2.224 622 066			
	2	2	ACK	HS	512 bytes (0F 14 11 1	2.224 677 300			
🕀 🕂 IN transaction	2	2	ACK.	HS	S12 bytes (13 15 15 1	2.224 690 766			
🗉 🕂 IN transaction	2	2	ACK	HS	S12 bytes (10 1E 19 1				
	2	2	ACK	H5	512 bytes (18 22 20 2	2.224 753 699			
Ready									

Figure 1 - Overview of the analysis software

## **Dockable panes**

Software panes can be moved to adapt to your needs. For example, the Data pane can be placed under the detail pane as in the screen shot below.

To move a pane, drag the pane and slide it to the desired place, release the mouse button and the pane will fix itself at the desired destination. In the screen shot below, the user dragged the Search pane and positioned it under the protocol element's list.

-	004-01-01] Mouse.ufo - El		ual USB						2
File	<u>View S</u> earch <u>R</u> ecord <u>H</u> e	lp							
•	🍬 🖙 😹 🔛 🤫 🚺	9 🖨 🕇	i iii d	à 📒	?.				
Item		Device	End	Status	Speed	Time	^	Search	μ
Enter	text here 🏼 🍸	E 🍸	E 🍸	E 🍸	E 🍸	Enter text 🍸			ŀ
- <mark>6</mark> 別	GetDescriptor (Device)	0(1)	0	ОК	LS	0.060 153 583		Search	
8	SetAddress (1)	0(1)	0	ОК	LS	0.082 650 283			
- <b>-</b> 8	GetDescriptor (Device)	1	0	ОК	LS	0.145 161 116		What kind of information are you looking for?	
- 22	GetDescriptor (Configuration)	1	0	ОК	LS	0.146 467 116		Protocol elements	
- 12	GetDescriptor (Configuration)	1	0	ок	LS	0.147 536 450		» Transactions	
8	GetDescriptor (String lang IDs)	1	0	ок	LS	0.149 581 450		» Endpoints data	
-22	GetDescriptor (String iProduct)	1	0	ОК	LS	0.150 464 950		» Setup requests	
-29	GetDescriptor (String lang IDs)	1	0	ок	LS	0.152 478 450		» Columns strings	
8	GetDescriptor (String iProduct)	1	0	ок	LS	0.153 463 783	-	» Errors	
<b>1</b> 22	GetDescriptor (Device)	1	0	ОК	LS	0.161 209 450			
<b>6</b> 8	GetDescriptor (Configuration)	1	0	ОК	LS	0.162 484 283			
-80	GetDescriptor (Configuration)	1	0	ОК	LS	0.163 464 783			_
	SetConfiguration (1)	1	0	OK	LS	0.165 462 616		🛛 🔽 Details 💫 Search 🚱 Options 🕋 Trigger	
3	SetIdle (All, Indefinite)	1	0	ОК	LS	0.192 100 783		Data	ц.
1	GetDescriptor (Report)	1	0	ОК	LS	0.193 540 116		0 1 2 3 4 5 6 0123456	Ē
	Left button click					4.456 816 950		0: 24 03 55 00 53 00 42 \$.U.S.B	-
8	HID Report	1	1	ОК	LS	5.464 885 783		7: 00 20 00 4F 00 70 00O.p.	
3	HID Report	1	1	ОК	LS	5.584 893 950		14: 74 00 69 00 63 00 61 t.i.c.a	
Ē	Right button click					13.185 412 616		21: 00 6C 00 20 00 4D 00 .1M.	
-	HID Report	1	1	ОК	LS	14.377 493 783		28: 6F 00 75 00 73 00 65 o.u.s.e	
3	HID Report	1	1	ОК	LS	14.529 504 283		35:00 .	
Ē	Several moves					21.922 009 450			
8	HID Report	1	1	ок	LS	22.018 015 283			
3	HID Report	1	1	ОК	LS	22.346 037 949			
8	HID Report	1	1	ОК	LS	22.354 038 116			
-	.HID.Report	1	.1	OK	15	22.362.038.783	~		

## Toolbar

•	Record (Ctrl+R)
	Starts recording USB traffic (an asterisk appears when the recording
	options are modified).
٠	Record and Append
	Starts recording USB traffic and append new traffic to the current
	document (an asterisk appears when the recording options are modified).
-	Recording options
	Opens the dialog permitting to configure all options related to recording like
	circular buffer, hard-disk streaming, limit, filters and triggers.
<b>2</b>	Open (Ctrl+O)
	Opens a previously saved file.
i 🚅	Open and Append
	Appends a previously saved file at the end of the current document.
	Save (Ctrl+S)
	Saves a data acquisition in a file.
€	Group transactions
$\rightarrow$	Displays transactions without grouping them into transfers.
<b>F</b>	Group transfers
	Displays transactions grouped into transfers.
153	Show/Hide SOFs
$\odot$	Displays or hides Start Of Frames.
-> <b>b</b>	Show/Hide NAKs
₽.	Displays or hides NAKs.
	Show/Hide Split transactions
⇒\$	Displays or hides Split transactions.
560	Show/Hide bus states
550	Displays or hides the bus and power supply states.
¢۹.	Find Shows the display enabling users to start searches.
	Automatic Coloring
	Color-codes transactions based on the device address and endpoint
	number.
	Set Time Reference
	Set the zero on a specific protocol element.
TRANET.	Reset Time Reference
	Reset the zero to the original value.

## **Protocol display**

The most basic USB element is the data packet, but this is not always the most relevant element. The transaction, generally comprising several data packets, is what ultimately conveys information. The analysis software lists transactions on the left side of the display. Each transaction includes the address of the device involved, the endpoint number, the transaction status, payload and absolute time expressed in seconds. All of this information is presented in a convenient column format.

The software analyses and decodes descriptors and requests so that they can be displayed in a format that is easier to follow. In just a few seconds you can record and display all the characteristics of a device being analyzed.

Every element displayed in the USB list protocol holds an easily identifiable icon as follows:

- ☑ Start-of-frame
- Setup transaction
- → Isochronous OUT transaction
- Bulk or Interrupt OUT transaction
- NAKed Bulk or Interrupt OUT transaction
- ← Isochronous IN transaction
- Bulk or Interrupt IN transaction
- NAKed Bulk or Interrupt IN transaction
- Ping ACK
- 롿 🛛 Ping NAK
- <table-of-contents> Start Split transaction
- Start Split transaction (erroneous)
- Complete Split transaction
- Complete Split transaction (erroneous)
- Control transfer OUT
- Control transfer IN
- High Speed Detection Handshake
- Extended SE0 state for more than 2 microseconds
- No activity detected for more than 3 milliseconds
- Power supply detected on DUT connector
- Loss of power supply on DUT connector
- User comment
- The element contains a warning
- The element contains an error
- X The element contains a low-level error

## Grouping of consecutive NAKs and SOFs

*Start Of Frames* are fundamental protocol elements, but they make the display too cumbersome if all SOFs are displayed. To prevent this inconvenience, the software automatically groups consecutive *Start Of Frames* and indicates the count of grouped elements. The following example shows an open group of *Start Of Frames* containing four elements.

Item	Device	Endp	Status	Speed	Comment	Time
Enter text here	E 🍸	E 🍸	E 🍸	E 🍸	Enter te 🍸	Enter text 🍸
🕀 🙀 GetDescriptor (Device)	1	0	ОК	FS	18 bytes (1	0.292 524 783
😑 😒 Start of Frame (4)				FS	1'001 -> 1'004	0.292 861 050
😇 Start of Frame				FS	1'001	0.292 861 050
😇 Start of Frame				FS	1'002	0.293 861 116
😇 Start of Frame				FS	1'003	0.294 861 183
😇 Start of Frame				FS	1'004	0.295 861 266
	1	0	ОК	FS	No data	0.295 911 433
				FS	1'005 -> 1'677	0.296 861 333
🛨 🕎 SetInterface (0, AltSet=1)	1	0	ОК	FS	No data	15.307 088 483
				FS	1'678 -> 486	15.307 982 383
	1	1		FS	No data	16.164 163 150
😇 Start of Frame				FS	487	16.165 041 133
	1	1		FS	No data	16.165 163 966
😇 Start of Frame				FS	488	16.166 041 199

Figure 2 - Grouping of consecutive Start Of Frames

Consecutive NAK transactions are also grouped together in the same way. Figure 3 shows a Bulk transmission for a storage device with relatively slow access. Each valid transaction is preceded by several NAKs, which reduces the reading performance of this device.

Item		Device	Endp	Status	Speed	Comment	Time
Enter text here	7	E 🍸	E 🍸	E 🍸	E 🍸	Enter te 🍸	Enter text 🍸
🗉 🚭 🛛 IN transaction		2	1	ACK	FS	64 bytes (E	1.873 215 483
🕀 <table-cell-rows> IN transaction (3)</table-cell-rows>		2	1	NAK	FS	No data	1.873 275 983
🕀 🗲 🛛 IN transaction		2	1	ACK	FS	64 bytes (E	1.873 335 233
		2	1	NAK	FS	No data	1.873 395 566
🕀 🗲 🛛 IN transaction		2	1	ACK	FS	64 bytes (E	1.873 483 166
		2	1	NAK	FS	No data	1.873 543 499
🕀 🗲 🛛 IN transaction		2	1	ACK	FS	64 bytes (E	1.873 586 416
		2	1	NAK	FS	No data	1.873 648 350
🕀 🚭 🛛 IN transaction		2	1	ACK	FS	64 bytes (E	1.873 706 099
🕀 <table-cell-rows> IN transaction (17)</table-cell-rows>		2	1	NAK	FS	No data	1.873 766 849

Figure 3 - Grouping of consecutive NAKed transactions

## Details display

The details display allows you to show more information about the selected protocol element. In addition to other tasks, this display allows you to visualize USB descriptors, requests details, etc.

By default, the software displays only high-level values in the tables. The least important elements are hidden to prevent the display from being overloaded (see Figure 4).

Device descriptor	× ×
🔱 bDeviceClass	No class Show full details
🔱 bMaxPacketSize0	64
🍁 idVendor	Ellisys
🌵 idProduct	0×8001
🌵 bodDevice	1.0

Figure 4 - Default display of tables

When you click on the *Show full details* button, the fields used less frequently appear in a slightly darker color (see Figure 5).

🔍 Device descriptor		* 꽏
😲 bLength	Valid	Show low level
🔱 bDescriptorType	DEVICE	
🗘 bcdUSB	2.0	
DeviceClass	No class	
🔱 bDeviceSubClass	No sub-class	
🔱 bDeviceProtocol	None	
🔱 bMaxPacketSize0	64	
🔱 idVendor	Ellisys	
🗼 idProduct	0×8001	
🔱 bodDevice	1.0	
🔍 iManufacturer	1	
🔍 iProduct	2	
🔍 iSerialNumber	3	
DNumConfigurations	1	

Figure 5 – Tables display with full details

😲 Device descriptor	* *			
Name	¥alue	Dec	Hex	Bin
🗼 bLength	Valid	18	0×12	00010010
🗼 bDescriptorType	DEVICE	1	0×01	00000001
🗘 bodUSB	2.0	512	0×0200	00000010 00000000
🔱 bDeviceClass	No class	0	0×00	00000000
DeviceSubClass	No sub-class	0	0×00	00000000
<ul> <li>bDeviceProtocol</li> </ul>	None	0	0×00	00000000
🗼 bMaxPacketSize0	64	64	0x40	01000000
🌵 idVendor	Ellisys	2746	0×0ABA	00001010 10111010
🌵 idProduct	0x8001	32769	0×8001	10000000 00000001
bcdDevice	1.0	256	0×0100	00000001 00000000
🗼 iManufacturer	1	1	0×01	00000001
<ul> <li>iProduct</li> </ul>	2 "USB Tracker 110"	2	0×02	00000010
🗼 iSerialNumber	3	3	0×03	00000011
bNumConfigurations	1	1	0×01	00000001

Clicking once on the *Show low level* button opens up an additional level that displays the digital equivalent of high-level values in decimal, hexadecimal, and binary (see Figure 6).

Figure 6 - Tables display with full details and low level values

## **Color-coding transactions**

When several devices are connected at the same time, the quantity of transactions being displayed can make it difficult to identify which elements belong to what transaction. To prevent you from having to perform a tiresome search, the analysis software allows you to color-code transactions that belong to a specific device. You can thus show information sent on a single device in one color whilst displaying all other traffic in another color. The device address defines the background color of a transaction and the endpoint number defines the text color (see Figure 7).

Item	Device	Endpoint	Speed	Status	Comment	Time
Enter text here	E 🍸	En 🍸	E 🍸	E 🍸	Enter 🍸	Ente 🍸
🕀 💼 GetPortStatus (1)	1	0	FS	OK		1.713 342
🕂 IN transaction (4 NAKs)	1	1	FS	NAK		1.732 341
	1	0	FS	ОК		1.828 348
	1	0	FS	ОК		1.832 335
📢 IN transaction	1	1	FS	ACK	1 byte (02)	1.860 334
⊕ GetPortStatus (1)	1	0	FS	ОК		1.862 334
	1	0	FS	ОК		1.866 333
	1	0	FS	ОК		1.890 332
🕂 IN transaction (NAK)	1	1	FS	NAK		1.924 330
🕀 🗊 GetDescriptor (Device)	0 (2)	0	FS	OK		1.922 330
	1	0	FS	OK		1.927 329
	1	0	FS	OK		1.931 329
📢 IN transaction	1	1	FS	ACK	1 byte (02)	1.956 327
	1	0	FS	OK		1.958 327
	1	0	FS	ОК		1.962 327
	0 (2)	0	FS	ОК		1.984 326
🕂 IN transaction (2 NAKs)	1	1	FS	NAK		1.988 325
🕀 🗐 GetDescriptor (Device)	2	0	FS	ОК		2.047 322
🕂 IN transaction (NAK)	1	1	FS	NAK		2.052 321
	2	0	FS	ОК		2.053 321
⊕	2	0	FS	ОК		2.058 321

Figure 7 - Cold	or-coding of	transactions
-----------------	--------------	--------------

Color-coding transactions for didactic purposes can be very interesting. For example, this approach enables you to compare the traffic of a hub to that of a device during the listing process, as is shown in the illustration above.

## Options

#### Display

#### Expand full-details

Select this check box to display full-details in the Details view by default. If this check box is cleared, only the most frequently used fields will be displayed.

#### • Expand low-level

Select this check box to display hexadecimal, decimal and binary values in the Details view by default. If this check box is cleared, only the most readable information will be displayed.

#### • Expand tree items

Select this check box to open all tree branches by default. If this check box is cleared, all tree items will stay closed until manually opened by the user.

#### Show transactions packets

Select this check box to show the packets contained in the transactions. This enables you to watch details of each packet but consumes much more memory.

#### • Show consecutive elements subitems

Select this check box to show the subitems of consecutive elements like Start of Frames. This enables you to see details of each consecutive element but consumes a lot more memory.

#### Advanced

#### Show wrongly connected devices

Select this check box to display all packets, even those that belong to devices connected above the analyzer. If this check box is cleared, many wrong packets will be hidden if a device is not properly placed on the analyzed bus. Refer to the Getting Started Manual for more information about the correct connection to your analyzer.

# Chapter 2: Recording options

## General tab

The recording options of the Ellisys USB analysis software can accommodate simple and complex recording scenarios. The General tab let the user configure the circular buffer, the hard-disk streaming, the limits and the split transactions combination.

The hard-disk streaming is a unique feature of the Ellisys analyzer. The analyzed data is streamed directly from the hardware to the hard-disk while recording in real-time through the high-performance USB 2.0 high-speed connection. This technique allows recording a huge quantity of data during a quite long period of time. Developers use this technique to debug sporadic issues that only appear once every 24 hours or after several gigabytes of data.

Circular Buffer			
Keep max		0	elements
Keep max		0	MBytes payload
Hard-disk streaming	1		
Split traces in parts of	64		MBytes
Save trace to			
_			
Open last trace when	recording stops		
Open last trace when imits Stop recording after Stop recording after Stop recording after	recording stops	0	sec MBytes elements
	recording stops		

### Filter tab

Please note that the filters are only available with the Ellisys USB Explorer 260 Analyzer or the Ellisys USB Explorer 200 Standard and Professional Editions. All references to the Ellisys USB Explorer 200 Professional Edition in this section also apply to the Ellisys USB Explorer 200 Standard Edition and the Ellisys USB Explorer 260, unless otherwise noted.

The Filter tab let the user specify basic and advanced transactions filters. This is useful to reduce the quantity of recorded data in order to save file size or processing time. The advanced filter features the unique partial payload filter.

Recording options	X
$\checkmark$ The connected unit supports advanced triggers and filters	
Record Second and append	
General Filter* Trigger Front-End	
Basic	
Drop Start of Frames	
✓ Drop Nak transactions	
Advanced	
Keep traffic from Devices 1 (e.g. 2,3)	
Keep traffic from Endpoints 2 (e.g. 2 IN, 3, 4 O	UT)
Custom rules:	
▼Rule 1 - Drop when Hdshk == 'NYET'	<b></b>
Rule 2 - Keep when Token == 'OUT' and truncate to 8 bytes	3
Add Delete Edit	
Load Save Load from Default Save as Default Reset to Factor	ry Default

The screen shot above defines a filter that:

- Drop all transactions with NAK handshake
- Keep only the traffic from Device 1
- Drop all transactions with NYET handshake
- Truncate OUT transactions to 8 bytes of payload

Please note that the custom rules override the "Keep traffic from" rules. In the example above, only traffic from Device 1 will be kept, and the custom rules will be applied on the remaining traffic. The custom rules are applied in the display order.

The custom rules are defined using the following dialog. Samples values and rule summary are displayed to ease the rule creation. If the specified criteria match, the rule action will be applied. In case of Keep transaction, the transaction will be kept and stored by the software. In case of Drop, the transaction will be deleted and will not appear in the trace.

Advanced Transaction C	riteria	
Rule name	Rule 1	
Criteria		
Rule action	○ Keep transaction	
Device address	!2	(e.g. !2,3)
Endpoint		(e.g. 23)
Token	IN, OUT	(e.g. !IN, OUT)
Data		(e.g. DATA1)
Handshake	NAK	(e.g. ACK, NAK)
Truncate payload to	0	bytes
Rule: Drop when Dev != '2' and T	oken == 'IN, OUT' and Hdshk ==	'NAK'
	ОК	Cancel

## Trigger tab

Please note that the hardware trigger is only available with the Ellisys USB Explorer 260 or the Ellisys USB Explorer 200 Professional Edition. All references to the Ellisys USB Explorer 200 Professional Edition in this section also apply to the Ellisys USB Explorer 260, unless otherwise noted.

#### Introduction

The Ellisys USB Explorer 260 and Ellisys USB Explorer 200 Professional Edition are fitted with a powerful and flexible hardware trigger. It enables one to trigger actions when external events happen, in case of repeated events or when specific sequences of events occur. The user interface handling is very quick. A text mode summary is offering a glimpse of events and defined actions.

#### **External events**

The hardware trigger can detect numerous conditions happening on the USB bus, specifically various USB packet types, electrical states, error states and trigger external states.

Split Packets  Token Packets Data Packets Handshake Packets Bus Conditions Trigger Conditions Counters Sequencers Configuration En	Token IN PID Token OUT PID Token SETUP PID Token PING PID Device Address Endpoint Number CRC Actions Recording control Trigger out Restart counter 0 Restart counter 1  dpoint Number ecifies the endpoint nur	Match Don't Care Don't Care Don't Care Match if Greater than 3 Match if Greater than 3 Don't Care No Action No No No No No
--	--	--

#### **Events Counters**

Powerful and flexible event counters are available. They allow counting simple external events as well as events' sequences. Counters can be reset by any event.

External Events	Criteria	
Counters	Event	Packet Error Event
🌳 Counter 0	Counter Match Value	0
Counter 1	Actions	
Sequencers	Recording control	Stop Recording 🗸 🗸 🗸
Configuration	Trigger out	No
	Restart counter 0	No
	Restart counter 1	No
	Recording control Controls the recording pro	ocess when the criteria matches.
When Packet Error matches the When Counter 0 matches the	Controls the recording protection	ocess when the criteria matches.

#### **Events Sequencers**

Events can be sequenced to trigger actions in intricate conditions.

General Filter Trigger		
🚊 External Events	Sequence	
Counters	1st event	Token Packet Event
Sequencers	2nd event	Data Packet Event
Sequencer 0	3rd event	None
Sequencer 1 Sequencer 2	4st event	None
Sequencer 3	Actions	
Configuration	Recording control	No Action
	Trigger out	Yes 🔻
	Restart counter 0	No
	Restart counter 1	No
	<b>Trigger out</b> Generate an event on the Tr	igger BNC connector if the
		igger BNC connector if the

#### **Trigger examples**

# Example 1: To generate an impulse on the BNC connector upon a High Speed Handshake event

In the "External Events > Bus Conditions" setup page, activate the "High Speed Handshake Success". Then set up the "Trigger Out" state on the Yes position. A warning is signaled when the BNC connector is set up on input.

Trigger		×		
Triggering is only available on the <u>USB Explorer 200 Professional Edition</u>				
🔄 External Events	Keep Alive	Don't Care 🛛 🔨		
Packet Errors	High Speed Handshake Success	Match 🔶		
Frame Packets	High Speed Handshake Failure	Don't Care		
Split Packets	Power On	Don't Care 🔤		
Token Packets	Power Off	Don't Care		
Data Packets	Actions	=		
Handshake Packets	Recording control	No Action		
Bus Conditions	Trigger out	Yes 😔 🛨		
Trigger Conditions	Restart counter 0	No 💌		
Counters	Trigger out			
Sequencers	Generate an event on the Trigger B	3NC connector		
Configuration	if the criteria matches.			
When Bus Condition matches then Warnings	from Default Save as Default	able it.		

In the "Configuration > Trigger Connector" setup page, check the connector is activated on output. It is possible to set up the output signal form depending on one's specific needs.

rigger		×			
Triggering is only available on the $\underline{L}$	Triggering is only available on the <u>USB Explorer 200 Professional Edition</u>				
🚞 External Events	🖃 Output Trigger				
Counters	Output Enable	Yes 🔂			
Sequencers	Polarity	Active Low			
Configuration	Mode	Pulse			
Recording					
Trigger Connector					
	Output Enable				
		n of the BNC connector.			
Load Save Load	from Default Save	as Default			
When Bus Condition matches then	Trigger out				
1					

#### Example 2: To stop recording after detecting a certain amount of low level errors

In this example, we'll set up Packet Error events criterion, then we'll set up a counter to stop recording.

In the "External Events > Packet Errors" setup page, activate the "Low Level Error" state. Do not define any actions; this event is going to be used to trigger a counter in the next step.

Trigger 🛛				
Triggering is only available on the <u>USB Explorer 200 Professional Edition</u>				
🔄 External Events	🖃 Criteria			
Packet Errors	PID Error	Don't Care		
Frame Packets	Low Level Error	Match 🕣		
Split Packets	Actions			
Token Packets	Recording control	No Action		
Data Packets	Trigger out	No		
Handshake Packets	Restart counter 0	No		
Bus Conditions	Restart counter 1	No		
Trigger Conditions				
Counters	Low Level Error			
Sequencers	Matches when a Low Le	vel Error occurs.		
Configuration				
Load Save Load	from Default Save as	Default		

In the "Counters > Counter 0" setup page, specify "Packet Error Event" as counter event. We shall use a triggering value of 10 in this example, however a value included between 1 and 65535 can be defined depending on one's requirements. Define the "Stop Recording" action in order to stop recording when the counter will reach its triggering value.

Trigger		×		
Triggering is only available on the USB Explorer 200 Professional Edition				
📄 External Events	🖃 Criteria			
🔄 Counters	Event	Packet Error Event 🚽		
🔷 Counter 0	Counter Match Value	10		
Counter 1	Actions			
Sequencers	Recording control	Stop Recording 🛛 🔂 🕇		
Configuration	Trigger out	No		
	Restart counter 0	No		
	Restart counter 1	No		
	Recording control Controls the recording pr matches.	ocess when the criteria		
Load Save Load f	rom Default Save as D	Default		
When Packet Error matches then N When Counter 0 matches then Sto				

# Example 3: Starting to record after detecting a certain amount of transactions containing specific data pattern

In this example, we shall set up Token and Data packets criterion, we'll set up a transaction sequencer, and then we'll set up a counter to start up recording.

Go to "External Events > Token Packets" and define Token criterion. We'll choose to trigger a Token IN for this example. It is possible to define a peripheral (or device) address and an endpoint number if several peripherals (devices) are actually on the bus.

Trigger 🔀					
Triggering is only available on the USB Explorer 200 Professional Edition					
🔄 External Events		Criteria			
Packet Errors		Token IN PID	Match 🔁		
Frame Packets		Token OUT PID	Don't Care		
Split Packets		Token SETUP PID	Don't Care		
🔷 Token Packets		Token PING PID	Don't Care		
Data Packets	Ð	Device Address	Don't Care		
Handshake Packets	Ð	Endpoint Number	Don't Care		
Bus Conditions		CRC	Don't Care		
Trigger Conditions		Actions			
Counters		Recording control	No Action		
Sequencers		Trigger out	No		
Configuration		Restart counter 0	No		
		Restart counter 1	No		
	1 -	<b>oken IN PID</b> pecifies if matches IN p	ackets.		
Load Save Load from Default Save as Default					
When Token Packet matches then No action					

Go to "External Events > Data Packets" and define criterion. We'll choose to trigger on a 4 bytes data packet with the pattern "00;00 C0;F0 0A;0F 4C;FF", that is at "XXXXXXXX 1100XXXX XXX1010 01001100" in binary. This pattern signifies that the Byte 0 can have any value, that the high part of the Byte 1 must be 0xC (the low part is masked and is not relevant), that the low part of the Byte 2 must be 0xA and that Byte 3 must be 0x4C.

Tr	Trigger 🛛 🛛					
	Triggering is only available on the USB Explorer 200 Professional Edition					
	🔁 External Events		Data DATA2 PID	Don't Care	^	
	Packet Errors		Data MDATA PID	Don't Care		
	Frame Packets	Ð	) Data Size	Match if Equal to 4	=	
	Split Packets	E	) Data Pattern	00;00 C0;F0 0A;0F 4C;FF		
	Token Packets		Byte 0	Value=00 Mask=00 🛛 🔶		t
	🌳 Data Packets		Byte 1	Value=C0_Mask=F0		
	Handshake Packets		Byte 2	Value=0A Mask=0F		
	Bus Conditions		Byte 3	Value=4C_Mask=FF		
	Trigger Conditions		Byte 4	Value=00 Mask=00	~	
	Counters	F	Byte O			
	E Sequencers		ipecifies a byte value	and mask criteria in		
	Configuration		nexadecimal.			
	Load Save Load from Default Save as Default					
	When Token Packet matches then No action When Data Packet matches then No action					

In the "Sequencers > Sequencer 0" setup page, specify the transaction's sequence, that is to say a Token Packet followed by a Data Packet in the example below.

Triggering is only available on the USB Explorer 200 Professional Edition	Trigger 🛛 🔀					
Ist event       Token Packet Even         Ist event       Data Packet Even         Ist event       Data Packet Even         Ist event       None         Ist event       No         Ist event       No         Ist event       No	Triggering is only available on the USB Explorer 200 Professional Edition					
Sequencers       2nd event       Data Packet Event                ◆ Sequencer 0 Sequencer 1 Sequencer 2 Sequencer 3 Configuration        2nd event       Data Packet Event                Packet Event        3rd event       None                Packet Event        4st event       None                Packet Event        Actions       Image: Configuration                Trigger out        No       No                Ist event        Image: Configuration       Image: Configuration	📄 External Events	Sequence		^		
<ul> <li>Sequencer 0</li> <li>Sequencer 1</li> <li>Sequencer 2</li> <li>Sequencer 3</li> <li>Configuration</li> <li>Actions</li> <li>Recording control</li> <li>No Action</li> <li>Trigger out</li> <li>No</li> <li>Restart counter 0</li> <li>No</li> </ul>	Counters	1st event	Token Packet Even 🗲			
Sequencer 1 Sequencer 2 Sequencer 3 Configuration Sequencer 0 No Sequencer 3 Sequencer	🔁 Sequencers	2nd event	Data Packet Event 🗲			
Sequencer 2 Sequencer 3 Configuration Sequencer 3 Configuration Configuration Sequencer 0 Configuration Configur	Sequencer 0	3rd event	None			
Sequencer 3 Configuration Recording control No Action Trigger out No Restart counter 0 No Ist event	Sequencer 1	4st event	None			
Configuration Trigger out No Restart counter 0 No Ist event	Sequencer 2	Actions				
Restart counter 0 No   Ist event	Sequencer 3	Recording control	No Action			
1st event	Configuration	Trigger out	No			
		Restart counter 0	No	~		
Specifies the first event of the sequence.         Load       Save         Load from Default       Save as Default         When Token Packet matches then No action         When Data Packet matches then No action         When Sequencer 0 detects Token Packet followed by Data Packet then No action						

In the "Counters > Counter 0" setup page, specify "Sequencer 0 Event" as a counter event. We'll us a triggering value of 32 in this example, however a value included between 1 and 65535 can be defined depending on one's requirements. Define the "Start Recording" action to stop recording when the counter will hit its triggering value.

Trigger 🛛 🔀						
Triggering is only available on the USB Explorer 200 Professional Edition						
📄 External Events	🖃 Criteria					
🔄 Counters	Event	Sequencer O Event				
🜩 Counter 0	Counter Match Value	32				
Counter 1	Actions					
Sequencers	Recording control	Start Recording 🛛 😔 🚽				
Configuration	Trigger out	No				
	Restart counter 0	No				
	Restart counter 1	No				
<b>Recording control</b> Controls the recording process when the criteria matches.						
Load Save Load from Default Save as Default						
When Token Packet matches then No action When Data Packet matches then No action When Counter 0 matches then Start recording When Sequencer 0 detects Token Packet followed by Data Packet then No action						

One must still make sure that the initial recording state is deactivated, which will allow to start the recording solely when the wished for sequences' count is reached.

Frigger	X			
Triggering is only available on the US	B Explorer 200 Professional Edition			
<ul> <li>External Events</li> <li>Counters</li> <li>Sequencers</li> <li>Configuration</li> <li>Recording Trigger Connector</li> </ul>	Recording Configuration     Initial Recording State Disabled			
	Initial Recording State Specifies the initial recording state. If the initial recording state is disabled, the recording will not start until a Start Recording action is fired.			
Load Save Load fr	om Default Save as Default			
When Token Packet matches then No action When Data Packet matches then No action When Counter 0 matches then Start recording When Sequencer 0 detects Token Packet followed by Data Packet then Start recording				

# Chapter 3: Real-time Monitoring

Please note that the real-time monitoring is only available with the Ellisys USB Explorer 260 Analyzer or the Ellisys USB Explorer 200 Standard and Professional Editions. All references to the Ellisys USB Explorer 200 Professional Edition in this section also apply to the Ellisys USB Explorer 200 Standard Edition and the Ellisys USB Explorer 260, unless otherwise noted.

## Overview

The Ellisys USB analysis software displays powerful real-time information while recording. The top of the dialog displays the serial number of the unit used for recording, the duration of the recording and the total recorded data. Tabs provide then detailed information about the recording session.

A text box at the bottom of the dialog can be used to insert comments in the trace. This is useful to put markers to easily find the position in post-capture analysis. For example the user can enter a text before doing a particular command on his device under test, and search later this comment to see the traffic generated by the command.

## General tab

The General tab displays the most common statistics as the quantity of transactions, the NAKed transactions, the invalid transactions, the detected devices and the bus usage.

USB traffic recording	
Analyzer Serial Number: EX200-1 Duration: 11:30	
Total recorded: 92.8 MB	
General Devices Frame	
Statistics	
Useful transactions:	315'728
NAKed transactions:	6'796
Invalid transactions:	1
Frames:	26'087
Detected devices:	4
Detected enumerations:	0
Bus usage ratio:	
Comment:	
Always On Top	Save Restart Cancel

## **Devices tab**

The Devices tab displays the input, output and total throughput of every detected device. This tab also displays the total link throughput and detected low-level errors.

USB traffic recor	ding		
Analyzer			
Serial Number:	EX200-18795		
Duration:	11:30		
Total recorded:	92.8 MB		
General Devices	Frame		
Device	Total	In	Out
1 2 3 4	0 bytes/s 172 kB/s 775 kB/s 496 bytes/s	0 bytes/s 0 bytes/s 775 kB/s 496 bytes/s	0 bytes/s 172 kB/s 496 bytes/s 0 bytes/s
Total Throughtput:	948 kB/s	Error: <b>1</b>	
Comment:			<u>I</u> nsert
🗌 Always On Top		Save <u>R</u> est	art <u>C</u> ancel

#### Frames tab

The Frame tab displays a graphical representation of the traffic in a frame. The horizontal axis is the time, with the origin set at the Start-of-Frame element. The NAKed transactions, the IN transactions and the OUT transactions are displayed on three separate lines. The vertical axis of these three graphs is the transactions occurrence. The height of the graph represents the occurrence. The Frame composition graph is a combination of the three other graphs. The colors are combined to have a quick understanding of the transactions nature.

The example below shows the traffic of a mouse, an audio speaker and a flash disk connected to an OHCI host controller. The Audio speaker is using isochronous out streaming. The mouse is not moved and is thus NAKing the interrupt endpoint. The host is reading a file on the flash disk with a bulk endpoint. We can see that the OHCI host leave some space at the beginning of the frame for the bulk traffic, then schedule interrupt and isochronous traffic, and then fill the space left with bulk traffic again.

USB traffic recording	
Analyzer Serial Number: EX200-18795 Duration: 11:30	
Total recorded: 92.8 MB	
General Devices Frame	
Frame composition	
NAKed transactions within a frame	
IN transactions within a frame	
OUT transactions within a frame	
Comment:	Insert
Always On Top	Save Restart Cancel

# Chapter 4: Protocol Summary

Please note that the Protocol Summary view is only available with the Ellisys USB Explorer 260 Analyzer or the Ellisys USB Explorer 200 Standard and Professional Editions. All references to the Ellisys USB Explorer 200 Professional Edition in this section also apply to the Ellisys USB Explorer 200 Standard Edition and Ellisys USB Explorer 260, unless otherwise noted.

## Introduction

The Summary pane shows a summary of the traffic being recorded or loaded by reporting the type of communications that occurred and the amount of bytes transferred. The Summary pane can be used to get a quick overview of the number of devices transmitting, the communication between the devices or the amount of errors received:

Summary 🛛						
🕲 🕲 Find next 📑						
Item		Total	^			
NYET		0 bytes				
• ERR		0 bytes				
🗆 🖨 Transactions	Count	Bytes				
Σ Total	1'774	1.32 kB				
🖃 🧼 By Device Address						
<ul> <li>Addr 0</li> </ul>	18	58 bytes				
Addr 1	497	592 bytes				
Addr 2	1'259	706 bytes	_			
🖃 🐗 By Endpoint						
Addr 0 - 0 IN	12	26 bytes				
Addr 0 - 0 OUT	6	32 bytes				
<ul> <li>Addr 1 - 0 IN</li> </ul>	65	294 bytes				
Addr 1 - 0 OUT	61	296 bytes				
<ul> <li>Addr 1 - 1 IN</li> </ul>	371	2 bytes				
<ul> <li>Addr 2 - 0 IN</li> </ul>	136	334 bytes				
Addr 2 - 0 OUT	32	136 bytes				
Addr 2 - 1 IN	1'091	236 bytes				
🖃 🕋 By Payload Length						
<ul> <li>0 to 7 bytes</li> </ul>	1'649	356 bytes				
<ul> <li>8 to 15 bytes</li> </ul>	125	1000 bytes				
16 to 31 bytes	0	0 bytes	~			

Figure 8 - Proto	col Summary view
------------------	------------------

The Summary pane is organized in several sections, each organized in several subsections. In addition, most main section also totalizes the count of all subsections.

#### Categories

The following categories are available in the Summary:

- Bus States: This category contains all protocol items that are not directly based on USB packets like electrical states, electrical protocols, etc.
- **Packets**: This category shows all packets listed by packets types like Token, Data, Handshake, etc.
- **Transactions**: This category shows the transactions listed by various criteria like the device address, the endpoint number, the payload size, etc.
- **Split Transactions**: This category shows the split transactions listed by various criteria like the split type, the hub port, etc.
- **Setup Transfers**: This category shows the setup transfers listed by various criteria like the recipient, the type, the request, etc.
- **Errors**: This category shows the errors on packets, on start-of-frames, by transactions, by split transactions and by LPM transactions.

### Find Summarized Items

Each event counted in a given section can be found in other panes using the Find Next and the Find Previous buttons.

To find a specific item, select it in the Summary list and click the Find Next button in the Summary view toolbar. You can also right click on the line and select Find next in the popup menu. The next item that produced a count in the selected line is highlighted and displayed in the Overview pane on the left side.

The same procedure applies to find the previous item matching the selected summary line. The Find Previous button on the toolbar is identified by a left arrow.

## **Total and Displayed Items Summary**

The Search is applied on the displayed items, but the summary offers both the total and the displayed items count. The Total column counts all the items that are available in the trace, while the Displayed column only counts the items that are displayed on the screen.

The Displayed column can be used to count the items matching a particular post-capture filter. Please refer to the chapter *Searching and Post-capture Filtering* for more information about post-capture filtering. When there is a difference between the Total and the Displayed count, the line is highlighted with a light background color. In the following screen-shot all bus states are filtered-out except the user comments. We can see that all counts are at 0 in the Displayed column:

Summary				×
🔄 🔄 Find next 📑				
Item		Displayed		Total 🔼
□ 🔐 Bus States	Count	Bytes	Count	Bytes
Σ <mark>Total</mark>	3		23'362	_
See Reset	0		4	
Keep Alive	0		23'350	
<sup>IDLE</sup> Suspend	0		1	
□ <sup>50+0</sup> / <sub>2</sub> Power changes				
• ON	0		1	
OFF				
🖃 🏦 Highspeed changes				
<ul> <li>Successful</li> </ul>				
<ul> <li>Timeout</li> </ul>	0		3	
Failed				
Trigger Conditions	0		0	
🖺 User comment	3		3	~

Figure 9 - Protocol Summary view with filtered items

#### Report export

The Summary data can be exported to several formats. Three formats are available:

- **Text**: The summary data will be exported to a plain text file. This is useful to integrate the data into a report for example.
- **CSV**: The summary data will be exported to a Comma Separated Values file. This is useful to open the data in spreadsheet applications like Microsoft Excel.
- XML: The summary data will be exported to a XML file. This is post-process the data using a custom script. XML parsers are available in all major scripting languages. The XML files can also be open in Internet Explorer to review its content.
## Chapter 5: Searching and Post-capture Filtering

## Search

The search function is used in conjunction with the software's instant filters. A search for the information is only conducted in the displayed elements. Several search modes are available as shown in Figure 10.

Search	×
Search	~
What do you want to search for?	
» Protocol elements	
» Transactions	
» Endpoints data	
» Setup requests	
» Columns strings	
» Errors	
	$\sim$
📴 Details 🔎 Search 🎯 Export 🦻 Options	

Figure 10 - Search panel

## Protocol elements search

The protocol elements search allows you to easily find USB elements like transactions, *Start Of Frames*, Resets, etc.

Figure 11 demonstrates a search for *Start Of Frames* and transactions.

Search	X
Search for protocol elements	^
Elements type:	
🗖 Invalid Packets	
✓ Start-of-Frames	
Transactions	
Split Transactions	
Transfers	
Resets	
🗖 Suspends	
🗖 Keep Alives	
Power Changes	
🗖 High Speed Hanshakes	
User Comments	
Options:	
Search up	
Back Search	
Dack Search	~
Details 🔑 Search 🎯 Export 🚱 Options	

Figure 11 – Search for protocol elements

### Transactions search

The transactions search enables you to easily find transactions according to their USB token (IN, OUT, or SETUP), their handshake (ACK, NAK or STALL) and their destination (device address and endpoint number).

Figure 12 demonstrates a search for all NAKed OUT transactions for device 2 on any endpoint.

Search	X
Search for transactions	~
Token type:	
🔽 оцт	
E SETUP	
Handshake type:	
ACK	
✓ NAK ✓ STALL	
I STALL	
Destination:	
Device 2	
Endpoint	
Options: Search up	
L Search up	
Back Search	
	$\sim$
💯 Details 🔎 Search 🞯 Export 📴 Options	

Figure 12 - Search for USB transactions

## Data search

The data search enables you to find data in transfers and transactions. You can perform a data search by entering a sequence of hex bytes or a chain of characters. This method may be used with instant filters for maximum versatility.

Figure 13 demonstrates a search for hex bytes '12 01 10 01' in all data packets containing 64 bytes that was not addressed to device 1.

Search		
Search	for endpoints data	~
Data:		
Pattern	12 01 10 01	
Size	64	
Destinati	on:	
Device	11	
Endpoint		
Options: Match   Searc		
	Search	~
🔊 Details	🔑 Search 🞯 Export 📴 Options	

Figure 13 - Search for endpoints data

### Setup requests search

The Setup requests search allows you to quickly find Setup requests according to their direction (Host-to-Device or Device-to-Host), type (Standard, Class or Vendor), recipient (Device, Interface, Endpoint or Others), bRequest, wValue, wIndex and wLength fields.

Figure 14 shows a search for vendor specific Setup requests 0xA0 in the device's direction. In addition, the *wValue* field must match the mask 0x8FFF.

Search		×								
Search f	or setup requests	~								
Setup data	:									
Direction	O Don't care 💿 To device O To host									
Туре	O Don't care O Standard O Class 🛛 🔍 Vendor									
Recipient	${}^{\bullet}$ Don't care ${}^{\circ}$ Device ${}^{\circ}$ ${}^{\circ}$ Interface ${}^{\circ}$ Endpoint ${}^{\circ}$ Other									
bRequest	0xA0 Use mask									
wValue	0xFFFF Use mask									
Mask	O Lo-byte O Hi-byte 💿 Custom 🛛 🛛 🛛 🗠									
wIndex	Use mask									
wLength	Use mask									
Destinatio	n:									
Device	1									
Endpoint										
Options:										
Back Se	earch	~								
🧊 Details 🍃	Search 🎯 Export   📴 Options									

Figure 14 - Search for setup requests

### Search by column

Searching for elements by column is an advanced search mode that allows you to find elements in accordance with the text displayed in the various columns. This mode uses the same syntax that the instant filters employ. Refer to the section **Instant filters** on page 43 for further information.

Figure 15 demonstrates a search for all NAKed OUT transactions coming from device 1 on endpoint 6 with 36 bytes of data:

Search		X
Search f	or columns strings	>
Columns:		
Item	out	
Device	1	
Endpoint	6	
Status	nak	
Speed		
Comment	36 bytes	
Time		
Options:		
🗖 Match o	ase	
🗖 Search	up	
Back S	earch	~
🔊 Details 🍃	🗅 Search 🎯 Export   📴 Options	

Figure 15 - Search for columns strings

## **Errors search**

The erroneous elements can be easily searched thanks to this mode.

Figure 16 shows a search for all types of errors.

Search	
Search for errors	~
Elements type:	
🔽 Invalid Packet Identifiers	
🔽 Invalid Start-of-Frames	
Invalid Transactions	
🔽 Invalid Split Transactions	
✓ Invalid Transfers	
Options: Search up	
Back Search	~
🗊 Details 🔎 Search 😵 Export 🦻 Options	

Figure 16 - Search of protocol errors

## **Instant filters**

The instant filters give you an easy way to filter undesirable protocol elements. The filters are composed of a sequence of character chains separated by commas. Wildcards can be specified in the Item, Speed, Status and Comment columns. Ranges can be specified for the Device, Endpoint and Time columns. If an exclamation mark appears in front of the filter, the corresponding elements will be excluded. The filters are not case-sensitive.

Start-of-Frames, NAKs and bus states cannot be filtered using the Instant Filters. Please use the buttons available on the toolbar buttons to filter out these elements.

The syntax of the filters is as follows:

filters = [!]filter[, filter, ...]

filter = string or wildcard or range

wildcard = *string containing* \* *and* ? *characters* (only for Item, Speed, Status and Comment columns)

range = min..max (only for Device, Endpoint and Time columns)

### Examples

The filter illustrated below saves all protocol elements intended for the device with address 1 or 2:

Item		Device		Endpoint		Speed		Status		Comment		Time	
Enter text here	7	1,2	7	Ent	Y	E	7	E	Y	Enter t	$\mathbb{Y}$	Ente	$\mathbb{Y}$

The following filter saves all elements belonging to device 1. It also excludes all endpoint 0 elements pertaining to this device:

Item		Device	Endpoint	Speed	Status	Comment	Time
Enter text here	Y	1 🍸	!0 7	E 🍸	E 🍸	Enter t 🍸	Ente 🍸

The following filter enables you to display all *GetDescriptor* requests that have taken place on *Low Speed* devices:

Item	Device	Endpoint	Speed	Status	Comment	Time
GetDescriptor	'En 🍸	Ent 🍸	LS 🍸	E 🍸	Enter t 🍸	Ente 🍸

The following filter saves all transactions containing 64 bytes of data that do not belong to device 1:

Item	Device	Endpoint	Speed	Status	Comment	Time
* transaction	r !1 🛛 🍸	' Ent 🍸	E 🍸	E 🍸	64 bytes 🍸	Ente 🍸

The filter illustrated below enables you to display all transactions that have taken place between 0 and 200 milliseconds:

Item		Device		Endpoint		Speed		Status		Comment		Time	
Enter text here	7	E	7	Ent	$\mathbb{Y}$	En	7	E	7	Enter t	$\mathbb{Y}$	00.2	$\mathbb{Y}$

## Chapter 6: Data Export

## Introduction

This software allows recorded USB transactions, endpoints data, Setup requests or descriptors to be exported. The exporting function can be used in conjunction with instant filters for maximum versatility.

Export	×
Export	^
Export	
What do you want to export?	
» USB transactions	
» Endpoints data	
» Setup transfers	
» Descriptors	
	$\checkmark$
💯 Details 🔎 Search 😵 Export 🚱 Options	

Figure 17 - Export panel

## **Exporting USB transactions**

The exportation of USB transactions allows USB traffic to be recorded in a text or XML file. The text format is often used for printing, whereas the XML format may be simply scanned using a script language in order to extract very specific information.

You can define whether all USB transactions in a file (*All*) or only filtered transactions (*Only filtered*) must be exported (see Figure 18).

Advanced options enable you to extend data packets that include transactions, SOF data packets clustered together in a single group, as well as grouped NAK transactions. The file that is generated in this way may be larger, but it also offers the advantage of providing maximum details about the elements.

Export	X
Export USB transactions	^
Export format: • XML • Text	
Exported items: O All O only filtered	
Advanced options: Don't export time Expand transactions packets Expand consecutive elements	
Back Export Details Pearch Search Coptions	~

Figure 18 - Export USB transactions

## XML format specifications

```
<Reset
  duration = time_expression
  time = time_expression />
<Suspended
  duration = time_expression
  time = time_expression />
<Power
  state = "ON" | "OFF"
  time = time_expression />
<!-- Packet category: Start of Frame -->
<Packet
   id = "SOF"
   speed = "LS" | "FS"
   time = time_expression>
  <frame>decimal</frame>
   <crc5>hexadecimal</crc5>
</Packet>
<!-- Packet category: Token -->
<Packet
   id = "IN " | "OUT" | "SETUP"
   speed = "LS" | "FS"
   time = time_expression>
  <deviceAddress>decimal</deviceAddress>
  <endpointNumber>decimal</endpointNumber>
   <crc5>hexadecimal</crc5>
</Packet>
<!-- Packet category: Data -->
<Packet
   id = "DATA0" | "DATA1"
   speed = "LS" | "FS"
   time = time_expression>
   <length>decimal</length>
  <data>hexadecimal_data</data>
  <crc16>hexadecimal</crc16>
</Packet>
<!-- Packet category: Handshake and special -->
<Packet
   id = "ACK" | "NAK" | "STALL" | "PRE"
   speed = "LS" | "FS"
   time = time_expression />
<Transaction
  type = "IN " | "OUT" | "SETUP"
  device = decimal
      endpoint = decimal
      status = "ACK" | "NAK" | "STALL" | "ERROR"
      speed = "LS" | "FS"
      time = time_expression>
   <!-- Content: Packet -->
</Transaction>
<ConsecutiveStartOfFrame
  count = decimal
  time = time expression>
   <!-- Content: Packet id="SOF" -->
</ConsecutiveStartOfFrame>
```

```
<ConsecutiveTransaction
  count = decimal
  time = time_expression>
  <!-- Content: Transaction status="NAK" -->
</ConsecutiveTransaction>
<Container
  title = string
  device = decimal
      endpoint = decimal
      status = "ACK" | "NAK" | "STALL" | "ERROR"
      speed = "LS" | "FS"
      time = time_expression>
  <!-- Content: Transaction, ConsecutiveStartOfFrame, ConsecutiveTransaction
-->
</Container>
<Document
  title = string
  date = string>
  <!-- Content: Transaction, ConsecutiveStartOfFrame,
         ConsecutiveTransaction, Container, Reset, Suspended, Power -->
</Document>
```

## XML file sample

The Figure 19 shows the enumeration of a USB mouse exported as a XML file in Microsoft Internet Explorer.

🗿 C:\mouse.xml - Microsoft Internet Explorer
Eile Edit View Favorites Iools Help
<pre>Ele Edit View Favorites Tools Help  </pre> <pre> </pre>
<pre>+ <container device="1" endpoint="0" speed="LS" status="0K" time="0.587 794" title="GetDescriptor (Configuration)"> + <container device="1" endpoint="0" speed="LS" status="0K" time="0.590 775" title="GetDescriptor (Configuration)"> + <container device="1" endpoint="0" speed="LS" status="0K" time="0.597 740" title="GetDescriptor (Configuration)"> + <container device="1" endpoint="0" speed="LS" status="0K" time="0.597 740" title="GetDescriptor (String)"> + <container device="1" endpoint="0" speed="LS" status="0K" time="0.609 650" title="GetDescriptor (String)"> + <container device="1" endpoint="0" speed="LS" status="0K" time="0.609 650" title="GetDescriptor (String)"> + <container device="1" endpoint="0" speed="LS" status="0K" time="0.618 589" title="GetDescriptor (String)"> + <container device="1" endpoint="0" speed="LS" status="0K" time="0.618 589" title="GetDescriptor (String)"> + <container device="1" endpoint="0" speed="LS" status="0K" time="0.623 560" title="GetDescriptor (String)"> + <container device="1" endpoint="0" speed="LS" status="0K" time="0.668 368" title="GetDescriptor (Device)"> + <container device="1" endpoint="0" speed="LS" status="0K" time="0.668 368" title="GetDescriptor (Configuration)"> + <container device="1" endpoint="0" speed="LS" status="0K" time="0.668 368" title="GetDescriptor (Configuration)"> + <container device="1" endpoint="0" speed="LS" status="0K" time="0.668 368" title="GetDescriptor (Configuration)"> + <container device="1" endpoint="0" speed="LS" status="0K" time="0.668 368" title="GetDescriptor (Configuration)"> + <container device="1" endpoint="0" speed="LS" status="0K" time="0.668 368" title="GetDescriptor (Configuration)"> + <container device="1" endpoint="0" speed="LS" status="0K" time="0.668 333" title="GetDescriptor (Configuration)"> + <container device="1" endpoint="0" speed="LS" status="0K" time="0.694 274" title="SetConfiguration(1)"> + <container device="1" endpoint="0" speed="LS" status="0K" time="0.724 188" title="GetDescriptor (Report)"> &lt; <container title="&lt;/td"></container></container></container></container></container></container></container></container></container></container></container></container></container></container></container></container></container></container></container></pre>
l 🔮 My Computer

Figure 19 - XML export sample

### Text file sample

The Figure 19 shows the enumeration of a USB mouse exported as a XML file in Microsoft Internet Explorer.

The sample below shows the enumeration of a USB mouse exported as a text file.

```
Filename: Mouse.ufo
Date: 23.04.2003 11:50:13

Power state<ON> time<0.000 000>

Reset duration<28.8 milliseconds> time<0.011 041>

Suspended duration<319.7 milliseconds> time<0.042 809>

Reset duration<31.1 milliseconds> time<0.362 537>
```

Container title<GetDescriptor (Device)> device<0 (1)> endpoint<0> status<OK>
speed<LS> time<0.457 092>
Transaction type<SETUP> device<0 (1)> endpoint<0> status<ACK> speed<LS>
 time<0.457 092>
Transaction type<IN> device<0 (1)> endpoint<0> status<ACK> speed<LS>
 time<0.458 084>
Transaction type<IN> device<0 (1)> endpoint<0> status<ACK> speed<LS>
 time<0.459 076>
Transaction type<IN> device<0 (1)> endpoint<0> status<ACK> speed<LS>
 time<0.460 068>
Transaction type<OUT> device<0 (1)> endpoint<0> status<ACK> speed<LS>
 time<0.462 057>

Reset duration<24.1 milliseconds> time<0.463 144>

```
Container title<SetAddress (1)> device<0 (1)> endpoint<0> status<OK> speed<LS>
time<0.518 978>
   Transaction type<SETUP> device<0 (1)> endpoint<0> status<ACK> speed<LS>
    time<0.518 978>
   Transaction type<IN> device<0 (1)> endpoint<0> status<ACK> speed<LS>
    time<0.519 970>
Container title<GetDescriptor (Device)> device<1> endpoint<0> status<OK>
 speed<LS> time<0.581 837>
   Transaction type<SETUP> device<1> endpoint<0> status<ACK> speed<LS>
    time<0.581 837>
        Packet id<SETUP> devAddr<1> epNum<0> crc5<0x1D> speed<LS>
        time<0.581 837>
       Packet id<DATA0> length<8> data<80 06 00 01 00 00 12 00> crc16<0xF4E0>
        speed<LS> time<0.581 860>
        Packet id<ACK> speed<LS> time<0.581 928>
   Transaction type<IN> device<1> endpoint<0> status<ACK> speed<LS>
     time<0.582 828>
        Packet id<IN> devAddr<1> epNum<0> crc5<0x1D> speed<LS> time<0.582 828>
        Packet id<DATA1> length<8> data<12 01 10 01 00 00 00 08> crc16<0x7711>
        speed<LS> time<0.582 852>
        Packet id<ACK> speed<LS> time<0.582 921>
   Transaction type<IN> device<1> endpoint<0> status<ACK> speed<LS>
     time<0.583 820>
        Packet id<IN> devAddr<1> epNum<0> crc5<0x1D> speed<LS> time<0.583 820>
        Packet id<DATA0> length<8> data<6D 04 0C C0 10 21 01 02> crc16<0xC3E9>
        speed<LS> time<0.583 844>
       Packet id<ACK> speed<LS> time<0.583 912>
   Transaction type<IN> device<1> endpoint<0> status<ACK> speed<LS>
     time<0.584 812>
        Packet id<IN> devAddr<1> epNum<0> crc5<0x1D> speed<LS> time<0.584 812>
        Packet id<DATA1> length<2> data<00 01> crc16<0x8F3F> speed<LS>
         time<0.584 836>
       Packet id<ACK> speed<LS> time<0.584 873>
   Transaction type<OUT> device<1> endpoint<0> status<ACK> speed<LS>
     time<0.585 804>
       Packet id<OUT> devAddr<1> epNum<0> crc5<0x1D> speed<LS>
        time<0.585 804>
       Packet id<DATA1> length<0> crc16<0x0000> speed<LS> time<0.585 828>
       Packet id<ACK> speed<LS> time<0.585 852>
```

Container title<GetDescriptor (Configuration)> device<1> endpoint<0> status<OK> speed<LS> time<0.595 740> Transaction type<SETUP> device<1> endpoint<0> status<ACK> speed<LS> time<0.595 740> Transaction type<IN> device<1> endpoint<0> status<ACK> speed<LS> time<0.596 732> Transaction type<IN> device<1> endpoint<0> status<ACK> speed<LS> time<0.597 724> Transaction type<IN> device<1> endpoint<0> status<ACK> speed<LS> time<0.598 716> Transaction type<IN> device<1> endpoint<0> status<ACK> speed<LS> time<0.599 708> Transaction type<IN> device<1> endpoint<0> status<ACK> speed<LS> time<0.600 700> Transaction type<OUT> device<1> endpoint<0> status<ACK> speed<LS> time<0.602 689> Container title<SetConfiguration (1)> device<1> endpoint<0> status<OK> speed<LS> time<0.694 274> Transaction type<SETUP> device<1> endpoint<0> status<ACK> speed<LS> time<0.694 274> Packet id<SETUP> devAddr<1> epNum<0> crc5<0x1D> speed<LS> time<0.694 274> Packet id<DATA0> length<8> data<00 09 01 00 00 00 00 00> crc16<0x2527> speed<LS> time<0.694 298> Packet id<ACK> speed<LS> time<0.694 365> Transaction type<IN> device<1> endpoint<0> status<ACK> speed<LS> time<0.695 266> Packet id<IN> devAddr<1> epNum<0> crc5<0x1D> speed<LS> time<0.695 266> Packet id<DATA1> length<0> crc16<0x0000> speed<LS> time<0.695 289> Packet id<ACK> speed<LS> time<0.695 315> Container title<GetDescriptor (Report)> device<1> endpoint<0> status<OK> speed<LS> time<0.724 188> Transaction type<SETUP> device<1> endpoint<0> status<ACK> speed<LS> time<0.724 188> Transaction type<IN> device<1> endpoint<0> status<ACK> speed<LS> time<0.725 179> Transaction type<IN> device<1> endpoint<0> status<ACK> speed<LS> time<0.726 171> Transaction type<IN> device<1> endpoint<0> status<ACK> speed<LS> time<0.727 163> Transaction type<IN> device<1> endpoint<0> status<ACK> speed<LS> time<0.728 155> Transaction type<IN> device<1> endpoint<0> status<ACK> speed<LS> time<0.729 147> Transaction type<IN> device<1> endpoint<0> status<ACK> speed<LS> time<0.730 139> Transaction type<IN> device<1> endpoint<0> status<ACK> speed<LS> time<0.731 130> Transaction type<OUT> device<1> endpoint<0> status<ACK> speed<LS> time<0.733 120>

ConsecutiveTransaction count<674> time<0.744 091>

Power state<OFF> time<6.135 733>

## **Exporting endpoints data**

The data can be exported to a binary file (*raw data*) or to a text file (*text*) as shown in Figure 20.



Figure 20 - Export endpoints data

## **Exporting Setup requests**

Setup requests can be exported to a C header file according to their direction (Host-to-Device or Device-to-Host), type (Standard, Class or Vendor), recipient (Device, Interface, Endpoint or Others), bRequest, wValue, wIndex and wLength fields (see Figure 21).



Figure 21 - Export setup transfers

## Sample

The following sample shows the export of Setup requests in a C header file. The array s\_setupRequest contains the exported setup requests. The structure SETUP\_REQUEST contains the fields and the setup request data.

```
struct SETUP REQUEST
{
  unsigned char bmRequestType;
  unsigned char bRequest;
  unsigned short wValue;
  unsigned short wIndex;
  unsigned short wLength;
  unsigned short cbData;
  const unsigned char* pData;
};
const struct SETUP_REQUEST s_setupRequest[] =
{
    0x00, 0x05, 0x0001, 0x0000, 0x0000, 0x0000, "" },
   { 0x80, 0x06, 0x0200, 0x0000, 0x0000, 0x0000, "" },
   { 0x80, 0x06, 0x0300, 0x0000, 0x00FF, 0x0004, "\x04\x03\x09\x04" },
   { 0x80, 0x06, 0x0302, 0x0409, 0x00FF, 0x002C,
     "\x2C\x03\x55\x00\x53\x00\x42\x00\x2D\x00\x53\x00\x2F
      \x20\x00\x4D\x00\x2D\x00\x42\x00\x41\x00\x34\x00\x37\x00" },
    0x80, 0x06, 0x0300, 0x0000, 0x00FF, 0x0004, "\x04\x03\x09\x04" },
   { 0x80, 0x06, 0x0200, 0x0000, 0x0009, 0x0009,
          '\x09\x02\x22\x00\x01\x01\x00\xA0\x19" },
    0x00, 0x09, 0x0001, 0x0000, 0x0000, 0x0000, "" },
0x21, 0x0A, 0x0000, 0x0000, 0x0000, 0x0000, "" },
   { 0x00, 0x00, 0x0000, 0x0000, 0x0000, 0x0000, NULL }
};
```

## **Exporting descriptors**

## Description

The descriptors can be exported to a C header or to an assembly file with format 8051 (*8051 ASM*) as shown in Figure 22.



Figure 22 - Export descriptors

## Sample

The sample below shows USB mouse descriptors export to a C header file.

```
// Device 1 device descriptor
static const unsigned char descriptor1[] =
ł
  0x12, 0x01, 0x00, 0x01, 0x00, 0x00, 0x00, 0x08,
  0x6D, 0x04, 0x02, 0xC0, 0x20, 0x01, 0x01, 0x02,
  0x00, 0x01
};
// Device 1 configuration descriptor
static const unsigned char descriptor2[] =
  0x09, 0x02, 0x22, 0x00, 0x01, 0x01, 0x00, 0xA0,
  0x19, 0x09, 0x04, 0x00, 0x00, 0x01, 0x03, 0x01,
  0x02, 0x00, 0x09, 0x21, 0x00, 0x01, 0x00, 0x01,
  0x22, 0x48, 0x00, 0x07, 0x05, 0x81, 0x03, 0x08,
  0x00, 0x0A
};
// Device 1 string descriptor 0
static const unsigned char descriptor3[] =
ł
  0x04, 0x03, 0x09, 0x04
};
// Device 1 string descriptor 2
static const unsigned char descriptor4[] =
{
  0x2C, 0x03, 0x55, 0x00, 0x53, 0x00, 0x42, 0x00,
  0x2D, 0x00, 0x50, 0x00, 0x53, 0x00, 0x2F, 0x00,
  0x32, 0x00, 0x20, 0x00, 0x4D, 0x00, 0x6F, 0x00,
  0x75, 0x00, 0x73, 0x00, 0x65, 0x00, 0x20, 0x00,
  0x4D, 0x00, 0x2D, 0x00, 0x42, 0x00, 0x41, 0x00,
  0x34, 0x00, 0x37, 0x00
};
// Device 1 class descriptor
static const unsigned char descriptor5[] =
  0x05, 0x01, 0x09, 0x02, 0xA1, 0x01, 0x05, 0x09,
  0x19, 0x01, 0x29, 0x04, 0x15, 0x00, 0x25, 0x01,
  0x95, 0x04, 0x75, 0x01, 0x81, 0x02, 0x95, 0x01,
  0x75, 0x04, 0x81, 0x03, 0x05, 0x01, 0x09, 0x01,
  0xA1, 0x00, 0x09, 0x30, 0x09, 0x31, 0x15, 0x81,
  0x25, 0x7F, 0x75, 0x08, 0x95, 0x02, 0x81, 0x06,
  0xC0, 0x09, 0x38, 0x95, 0x01, 0x81, 0x06, 0x09,
  0x3C, 0x15, 0x00, 0x25, 0x01, 0x75, 0x01, 0x95,
  0x01, 0xB1, 0x22, 0x95, 0x07, 0xB1, 0x01, 0xC0
};
```

## **Exporting generator scripts**

## Description

The trace can be exported to an Ellisys USB Explorer 260 Generator script. This is useful to replay erroneous scenarios that were recorded with the analyzer and that are not easy to reproduce. By replaying the scenario with the generator, the developer has the opportunity to fix the issue and ensure it is correctly fixed with a stable setup.

All packets of the trace can be exported, or just the packets from the host, as shown in Figure 23.

Export	X
Export generator script	^
Traffic to export:	
Host and devices packets	
O Host packets only	
Exported items:	
O All	
Only filtered	
Back Export	
	~

Figure 23 - Export generator script

## Sample

The sample below shows host packets for a GetDescritor (Device) request.

```
void Main()
{
   SendPacket ( RawData => [ 0x2D, 0x00, 0x10 ], Interval => 0.0 );
   SendPacket ( RawData => [ 0xC3, 0x80, 0x06, 0x00, 0x01, 0x00, 0x00, 0x40,
   0x00, 0xDD, 0x94 ], Interval => 25.333us );
   WaitPacket( );

   SendPacket ( RawData => [ 0x69, 0x00, 0x10 ], Interval => 108.667us );
   WaitPacket( );

   SendPacket ( RawData => [ 0x69, 0x00, 0x10 ], Interval => 49.000us );
   WaitPacket( );

   SendPacket ( RawData => [ 0x69, 0x00, 0x10 ], Interval => 48.167us );
   WaitPacket( );

   SendPacket ( RawData => [ 0x69, 0x00, 0x10 ], Interval => 48.167us );
   WaitPacket( );

   SendPacket ( RawData => [ 0x2 ], Interval => 95.000us );

   SendPacket ( RawData => [ 0x69, 0x00, 0x10 ], Interval => 22.833us );
   WaitPacket( );
```

```
SendPacket ( RawData => [ 0x69, 0x00, 0x10 ], Interval => 48.167us );
WaitPacket( );
SendPacket ( RawData => [ 0x69, 0x00, 0x10 ], Interval => 48.333us );
WaitPacket( );
SendPacket ( RawData => [ 0xD2 ], Interval => 94.833us );
SendPacket ( RawData => [ 0x69, 0x00, 0x10 ], Interval => 22.167us );
WaitPacket( );
SendPacket ( RawData => [ 0x69, 0x00, 0x10 ], Interval => 48.500us );
WaitPacket( );
SendPacket ( RawData => [ 0xD2 ], Interval => 63.500us );
SendPacket ( RawData => [ 0xE1, 0x00, 0x10 ], Interval => 24.167us );
SendPacket ( RawData => [ 0x4B, 0x00, 0x00 ], Interval => 25.333us );
WaitPacket( );
```

}

## Chapter 7: Software and drivers installation

## Install or update the analysis software

The analysis software's latest version is available at:

http://www.ellisys.com/support/download.php

After you have downloaded the software installation file, run it and follow the step-by-step installation instructions.

Please note that the installation requires Microsoft Windows Installer 2.0 or later. If the installation does not run smoothly, or if the system indicates that there is a version error, update your Windows Installer.

## Automatically install or update the device driver

If the latest version of the analysis software is properly installed, the driver should install automatically when the analyzer is connected. If not, please refer to the following paragraph.

## Manually install or update the device driver

- 1. Install the latest version of the USB analysis software as explained in the previous section.
- 2. Connect the USB analyzer.
- 3. If you are connecting the analyzer for the first time, please wait until Windows displays a message that says it discovered a new device and then jump to step 5.

 If you want to update an already-installed device driver, go to the Device Manager (Start | Control Panel | System icon | Hardware tab | Device Manager button). Open "Ellisys protocol analyzers" and right click on your analyzer. Select "Update Driver..." (see Figure 24).



Figure 24 - Updating driver from the Device Manager

5. Select "Install from a list or specific location (Advanced)" and click next (see Figure 25).



Figure 25 - Install from a list or specific location

6. Select only "Include this location in the search" and type the folder path in which you have installed the software (see Figure 26)

Found New Hardware Wizard	
Please choose your search and installation options.	
Search for the best driver in these locations.	
Use the check boxes below to limit or expand the default search, which includes local paths and removable media. The best driver found will be installed.	
Search removable <u>m</u> edia (floppy, CD-ROM)	
Include this location in the search:	
C:\Program Files\Ellisys\Visual USB\Drivers Visual Browse	
Don't search. I will choose the driver to install.	
Choose this option to select the device driver from a list. Windows does not guarantee that the driver you choose will be the best match for your hardware.	
< <u>B</u> ack <u>N</u> ext > Cancel	

Figure 26 - Search for the best driver

7. Windows will find the driver and install it (see Figure 27).

Found New Hardware Wizard		
Please wait while the wizard insta	Ils the software	
USB Explorer 200		
ellex200.sys To C:\WINDOWS\Sy	ustem32\Drivers	
	< <u>B</u> ack <u>N</u> ext > Cancel	

Figure 27 - Driver installation

8. After the new files have been copied, the installation is completed (see Figure 28).



Figure 28 – Driver installation completed

You can then launch the USB analysis software and use it.

## Install an optional functionality

Follow the steps below to install an optional functionality on your analyzer:

 Obtain the unlocking software from your local distributor or directly from Ellisys if there is no distributor in your country (see Figure 29). Please contact us for more information on <u>http://www.ellisys.com/sales/</u>



Figure 29 - I con of the unlocking software

2. If the software is embedded in a compressed file, first decompress it. Current operating systems should be able to decompress it without additional software (see Figure 30). If not, you will easily find some free software on the Web to decompress it.

	<b>Open</b> Search Explore
	Extract All
	Send To 🔸
	Cut
	Сору
	Create Shortcut
	Delete
	Rename
	Properties
Ĩ	
10-1 ZIF	2345.

Figure 30 - Extracting compressed software

TR

3. Launch the software. It will display your analyzer serial number on which it is intended to run (see Figure 31).

👫 Opti	ions Configurator	<
	nis software to enable options of the racker 110.	
⇒	Please connect the USB Tracker 110 with serial number TR110-12345 to enable the following options:	
	Export functionality	
	Set Options	
	Exit	

Figure 31 - Software waiting for an analyzer connection

4. As soon as the analyzer is connected, the software will activate the button "Set Options" (see Figure 32). Only the options displayed in the white rectangle will be activated. If this does not correspond to the options you bought, please contact your local distributor.

🖏 Opt	ions Configurator	
	nis software to enable options of the racker 110.	ellisys
1	Please connect the USB Tracker 110 TR110-12345 to enable the following	
	Export functionality	
	Set Options	
		E <u>x</u> it

Figure 32 - Software ready to activate optional functionality

If you connect another USB analyzer, an error message will be displayed (see Figure 33). If you connected your analyzer and this message appeared, please contact your local distributor.

👫 Opti	ions Configurator 🛛 🗙
Use this software to enable options of the USB Tracker 110.	
×	Please connect the USB Tracker 110 with serial number TR110-12345 to enable the following options:
	Export functionality
	USB Tracker 110 with wrong serial number detected!
	Set Options
	Exit

Figure 33 - Wrong serial number detected

6. Press "Set Options" to activate the optional functionality. The software will then display a success message (see Figure 34).

👫 Opti	ons Configurator 🔀
	is software to enable options of the ellisys
1	Please connect the USB Tracker 110 with serial number TR110-12345 to enable the following options:
	Export functionality
5	Set Options have been set.
<b>–</b>	Thank you for using Ellisys products.
	Exit

Figure 34 - Optional functionality properly installed

The optional functionality is now installed.

## Chapter 8: Frequently Asked Questions

# The basic component of USB is the packet, but I can only see transactions and transfers. Is it possible to view packets?

Yes, packets for the selected transaction are displayed in the right hand window. We have chosen this method to avoid overloading the transaction tree and because useful information is actually only transmitted when a transaction has been completed.

#### Which standard requests is the software able to decode?

The software decodes standard requests and descriptors, but also all class requests and descriptors, primarily for Audio, Mass Storage, Communication, Human Interface Device, Still Image, Printer, Bluetooth HCI, Video, Wire Adapter, Device Firmware Upgrade and Hub. The software also has an extension enabling the decoding of Vendor Specific protocols. The USB classes decoding is an option sold separately.

#### I would like to filter out certain undesirable information. What do I need to do?

A text zone for each column is provided for this. Simply enter '2,3' in the device column to display peripherals 2 and 3 only. In the same way, entering '!0,1' in the endpoint column hides all the endpoints except 0 and 1.

# The detailed view enables me to see high-level textual information. What do I need to do to see numeric low-level information?

Low-level information (decimal, hexadecimal and binary data) is available by clicking on the button  $\bigotimes$  (Show low-level) located at top of the frame for the packet (right view).

# The software shows a descriptor in the detailed view but it seems that some fields are missing. Where are they?

The software hides fields that are irrelevant. You can display them by clicking on the  $\aleph$  (Show full-details) button. Additional fields will appear in a slightly darker color.

#### What do consecutive SOF and NAK mean?

Consecutive SOF and NAK gather similar transactions to avoid overloading the screen. So, recording a bus with no activity for a few seconds will gather thousands of SOFs on just one line.

#### Will an invalid SOF packet be lost in a consecutive SOF?

No, an invalid SOF will separate the group to which it belongs into two valid groups and will be placed between these two groups. It will also be identified by a logo that indicates it is invalid.

#### The software tells me a device is wrongly connected. What does it mean?

It is possible that the analyzer is connected on the same bus controller as a device being tested. This configuration is not supported. Please refer to the Getting Started Manual for more information on the appropriate connections.

#### I bought a software option but it is not activated. What can I do?

Please ensure you executed the activation software sent by email through your local distributor (read *Chapter 7: Install an optional functionality* for more information on how to install an optional functionality).

Please check afterwards that your software version is the latest available on the download section of our website:

http://www.ellisys.com/support/download.php

Then restart the software with the analyzer connected.

#### How do I print from the software?

Printing is done using the data export feature. Examples of exported data are available in the installation folder of the USB analysis software (by default *C:\Program Files\Ellisys\Visual USB*). Look for files into the Samples folder.

Here is a description of some of the files available in this folder:

#### Mouse transactions.txt

Transactions of a USB mouse exported in text format, easily printable for later analysis.

#### • Mouse transactions.xml

Transactions of a USB mouse exported in XML format, analyzable with a custommade software.

#### Mouse descriptors.h

Descriptors of a USB mouse exported in a C header file.

#### • Mouse setup requests.h

All Setup requests of a USB mouse exported in a C header file.

#### Mouse.ufo

File used as a base for all exportation samples.

Be aware that some exported files may become large and it is often necessary to filter transactions before exportation. Instant Filter can easily suppress any redundant information (See <u>http://www.ellisys.com/products/usbex200/feature02.php</u> for more information).

## **Chapter 9: Troubleshooting**

### I am done with a recording and I notice several CRC errors. What's going on?

Check to make sure that the total length of the cables connected to the analyzer DUT does not exceed 3 meters. Then try an analysis of a sales device before reconnecting the device that you wish to test.

# The analysis software indicates that the devices are not correctly connected to the analyzer. What should I do?

The devices connected to the test computer must be below the analyzer for a correct analysis to be carried out. Refer to the Getting Started Manual for more information about correct connection of your analyzer.

# The analysis software indicates that the analyzer is connected to the same controller as the device under test. What should I do?

If you would like to use the analyzer and the device that you want to analyze on the same computer, you must have at least two powerful host controllers. Refer to the Getting Started Manual for more information about correct connection of your analyzer.

# When traffic is recorded in real time, the software indicates that an *overflow* error has appeared. What does this mean?

This error may appear when you try to analyze a device that uses a substantial amount of bandwidth, such as a USB hard drive, with a computer that is not very powerful or does not have a USB 2.0 connection. The configuration that uses two computers is most suitably adapted for an analysis of this type of device. Make sure that your analysis computer is powerful enough and has a USB 2.0 connection.