

CATC SAS001MA
SAS / SATA Analyzer



In Stock

Used and in Excellent Condition

Open Web Page

<https://www.artisanng.com/79361-5>

All trademarks, brandnames, and brands appearing herein are the property of their respective owners.



Your **definitive** source
for quality pre-owned
equipment.

Artisan Technology Group

(217) 352-9330 | sales@artisanng.com | artisanng.com

- Critical and expedited services
- In stock / Ready-to-ship

- We buy your excess, underutilized, and idle equipment
- Full-service, independent repair center

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

SASTracer / Trainer **SAS - SATA Protocol Analyzer & Exerciser**



September 1, 2005

Michael Micheletti
Sr. Product Marketing Manager

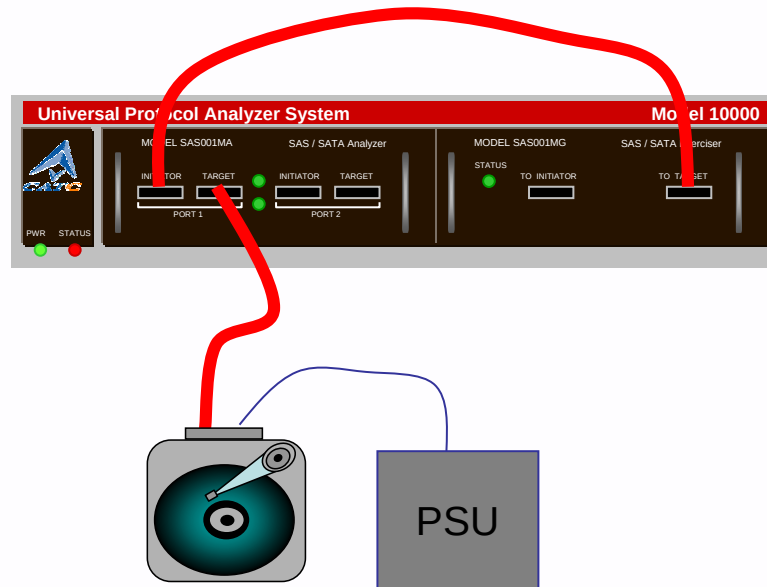
LJDN-ST-CA-0241-0001

SASTracer / Trainer Overview

- Integrated Analyzer plus Exerciser
 - Transmit and Record in one Integrated System
 - 2 port Record / 1 port Generation
 - 1.5/3Gbps SAS/SATA compliant
 - Host/Initiator or Target emulation
- Error Injection:
 - CRC Error / Scrambling error / ALIGN error
 - Code Violation / RD error / Invalid state transition
- Protocol Compliance:
 - Verify compliance for SAS and SATA
 - SAS Compliance Test Suite™
 - Over 100 test scripts for validating link/transport layer conformance
 - Available at no additional cost with Exerciser



Test Setup: Initiator Emulation



3

Traffic Generation File Format

Two sections in file

Declarations Block:

- Include Files
- Constants
- Templates
- Patterns

Generation Block:

- Generate { ... }

```
%include "settings.inc"
%include "PrimitivesDecl.inc"

set
generationmode=gen_mode_sata_host
set speed=link_speed_3g
set autoalignsata=true

Generation {
    CONNECT
    IDLE (100)
    SATA_R_RDY    ( 4 )
    SATA_CONT
    IDLE (10)
}
```

4

Include files: Settings.inc

Two Approaches:

1) Auto-Mode

- Automatically uses Default settings from Specifications

OOB, Speed Neg, ALIGNs

2) Manual mode

- COMINIT/COMWAKE OOB Signal
- SATA Link Initialization
- SAS/SATA Speed Negotiation
- Autowait
- Wait Command Timeout
- Scrambling Mode

Burst/Idle Intervals, etc..

*SNLT, SNTT timing, etc..
WAIT After, WAIT Before*

Plus....

- Emulation mode
- Set Speed

*HOST or DEVICE
1.5G or 3G*

5

Configuration

Settings.inc - Notepad

```
##### AutoMode Settings #####
Set AutoOOBMode = TRUE # go
Set AutoHoldMode = FALSE # res
Set AutoDMAT = FALSE # res
Set AutoHandshake = FALSE # res
Set AutoSpeedNeg = TRUE # aut
Set AutoAlignSAS = ON # aut
Set AutoAlignSATA = OFF # aut

##### COMINIT OOB Signal Settings #####
Set COMINIT_NumBursts = 6 # The
Set COMINIT_BurstLen = 160 # Bur
Set COMINIT_IdleLen = 480 # Idl
Set COMINIT_NegLen = 800 # Neg

##### COMWAKE OOB Signal Settings #####
Set COMWAKE_NumBursts = 6 # The
Set COMWAKE_BurstLen = 160 # Bur
Set COMWAKE_IdleLen = 160 # Idl
Set COMWAKE_NegLen = 280 # Neg
```

Table 36— OOB signal transmitter requirements

Signal	Burst time	Idle time	Negation time
COMWAKE	160 OOBt	160 OOBt	280 OOBt
COMINIT/RESET	160 OOBt	480 OOBt	800 OOBt
COMSAS	160 OOBt	1 440 OOBt	2 400 OOBt

6

Precedence

LeCroy SASTracer Bus and Protocol Analyzer - [C:\User\Desktop\SATA.ssg]

GEN MODE - SATA Host, SSC is ON

LINK SPEED - 3Gbps

RateMatching	MultiSpeed
OFF	OFF

CONFIGURATION

AutoOOB	AutoHOLD	AutoDMAT	Aut
ON	OFF	OFF	

OOB Settings - COMINIT

Num Bursts	Burst Length	Idle Length	Neg
99	160	480	

OOB Settings - COMWAKE

Num Bursts	Burst Length	Idle Length	Neg
6	160	160	

OOB Settings - COMSAS

Num Bursts	Burst Length	Idle Length	Neg

Generation Script Editor

```

1  %include "generation\include\Settings.inc"
2
3  Set COMINIT_NumBursts = 99
4
5

```

SATA.ssg

SASTracer SN:214

Ready

Ln 3, Col 2 Search

7

Precedence

LeCroy SASTracer Bus and Protocol Analyzer - [C:\User\Desktop\SATA.ssg]

GEN MODE - SATA Host, SSC is ON

LINK SPEED - 3Gbps

RateMatching	MultiSpeed
OFF	OFF

CONFIGURATION

AutoOOB	AutoHOLD	AutoDMAT	Aut
ON	OFF	OFF	

OOB Settings - COMINIT

Num Bursts	Burst Length	Idle Length	Neg
6	160	480	

OOB Settings - COMWAKE

Num Bursts	Burst Length	Idle Length	Neg
6	160	160	

OOB Settings - COMSAS

Num Bursts	Burst Length	Idle Length	Neg

Generation Script Editor

```

1
2  Set COMINIT_NumBursts = 99
3  |
4  %include "generation\include\Settings.inc"
5

```

SATA.ssg

SASTracer SN:214

Ready

Ln 3, Col 1 Search

8

Configuration Settings

- Most global setting can be in the Global settings and then changed in the generation block
- Exception: 3 commands only have effect when set in global settings:
 - set Link Speed =
 - set GenerationMode =
 - set SSC =

9

AutoAlign settings

- SAS - Sends 2 ALIGN Primitives every 2048 dwords
- SATA - Sends 2 ALIGN Primitives every 256 dwords

The Global Setting "AutoAlign"

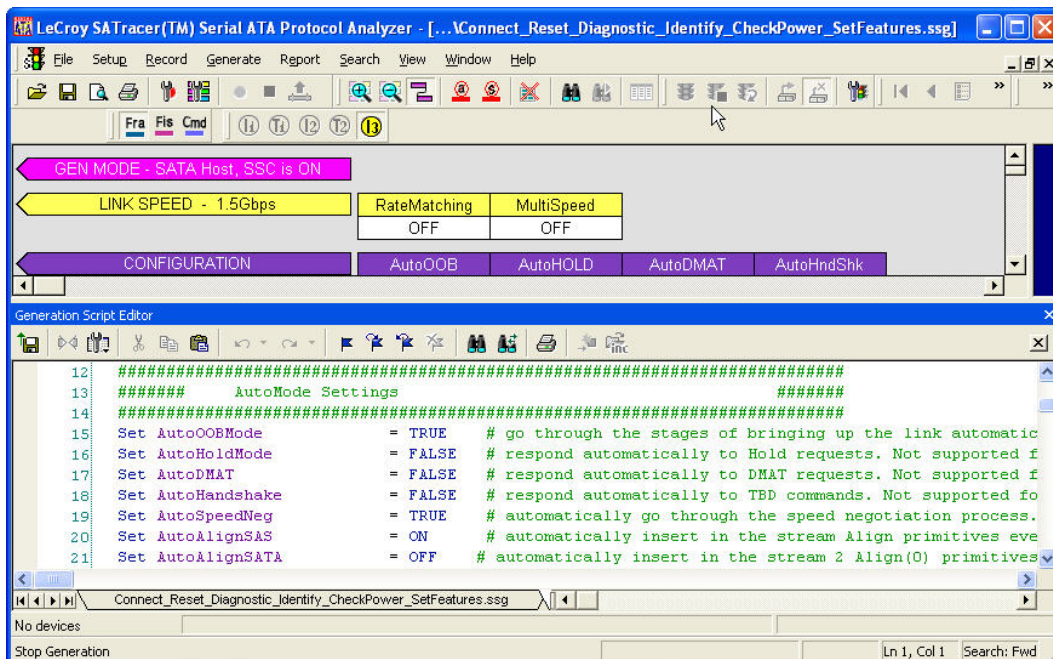
set AutoAlignSAS = ON / OFF

set AutoAlignSATA=ON / OFF

- Defaults:
 - For SATA host/device emulation, then
 - AutoAlignSATA is assumed to be ON
 - AutoAlignSAS is assumed to be OFF.
 - For SAS emulation, then the assumptions are opposite:
 - AutoAlignSATA is assumed to be OFF
 - AutoAlignSAS is assumed to be ON
 - For STP,
 - AutoAlignSAS and AutoAlignSATA are assumed to be ON

SATA Exercise 1

Change Settings

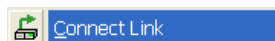


11

Bringing the link up

Three Ways to Transmit OOB

1. Icon/Menu macros



2. Automated in a Script

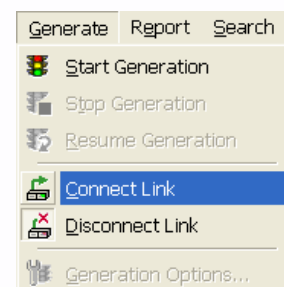
- CONNECT keyword

3. Manually in a Script

- Set AutoOOBMode = FALSE
- Set AutoSpeedNeg = FALSE
- Set AutoAlignSAS = OFF
- Set AutoAlignSATA = OFF
- COMINIT
- IDLE(800)
- COMWAKE
- IDLE(80)



Using Commands to
transmit OOB sequence



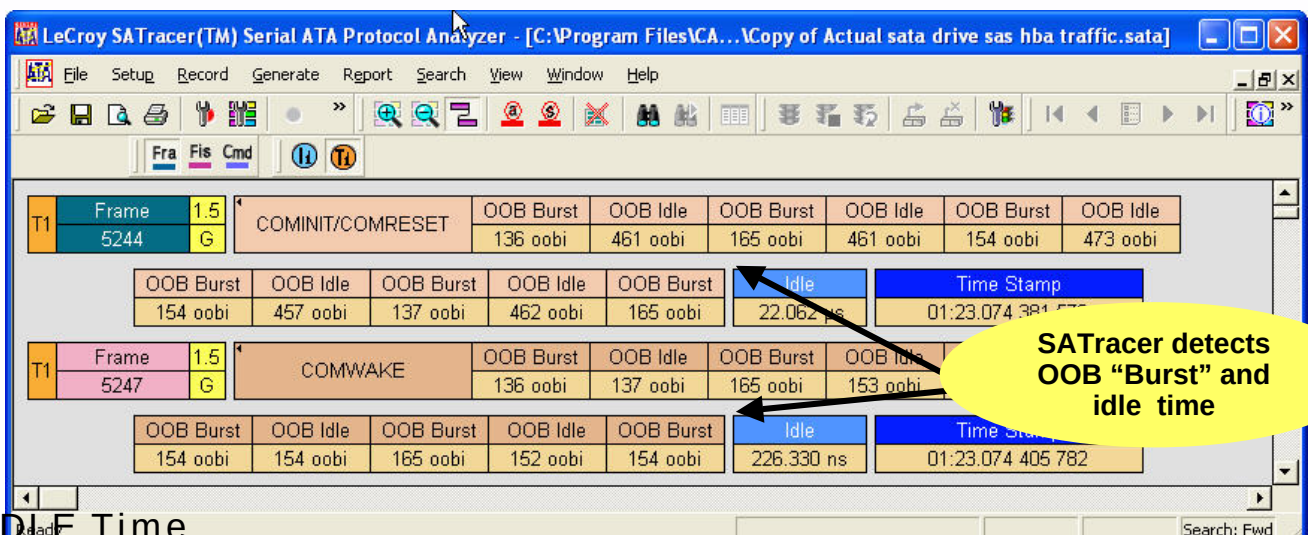
12

Connect Command

- Basic CONNECT command or icon will send:
 - COMINIT/COMRESET
 - COMSAS
 - COMWAKE
- To not send COMSAS - use manual Control:
 - COMINIT
 - IDLE(800)
 - COMWAKE
 - IDLE(80)

13

OOB Detection



Idle Time

OOB Signal	May detect	Shall detect	Shall not detect
COMWAKE	55 to 175 ns	101.3 to 112 ns	< 55 or > 175 ns
COMINIT/COMRESET	175 to 525 ns	304 to 336 ns	< 175 or > 525 ns
COMSAS	525 to 1575 ns	911.7 to 1008 ns	< 525 or > 1575 ns

14

Primitives

DWORD values Predefined in *PrimitivesDecl.inc* file

```

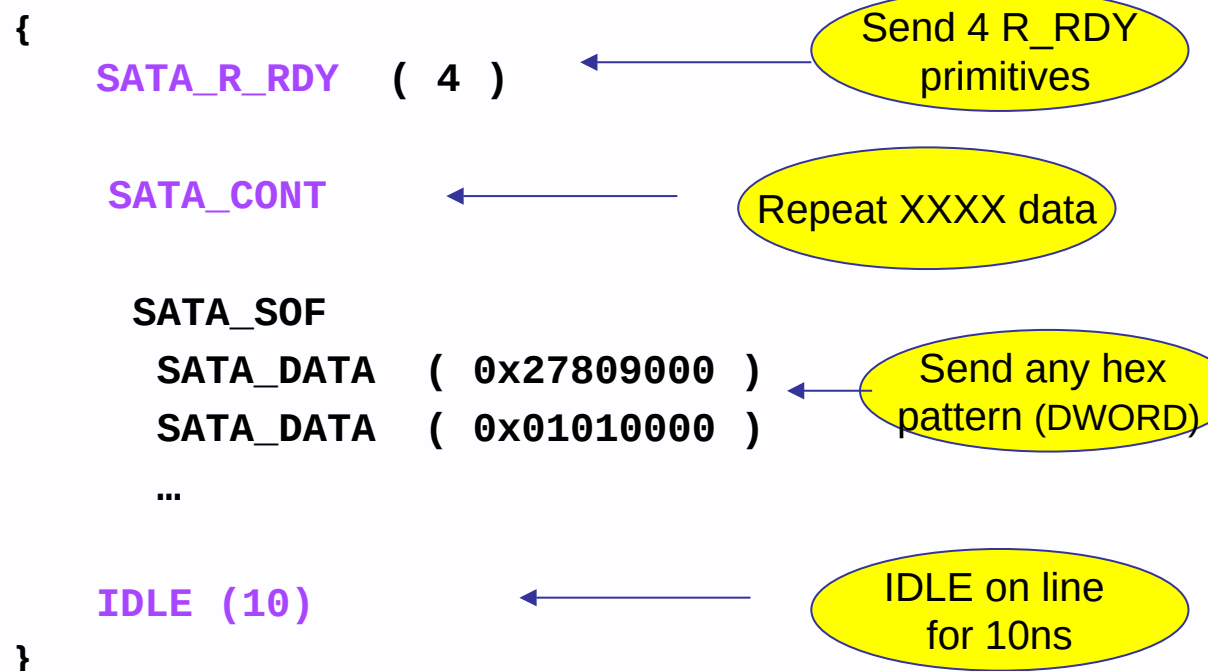
39 Primitive SATA_SOF           = K28.3 D21.5 D23.1 D23.1
40 Primitive SATA_EOF           = K28.3 D21.5 D21.6 D21.6
41 Primitive SATA_CONT          = K28.3 D10.5 D25.4 D25.4
42 Primitive SATA_DMAT          = K28.3 D21.5 D22.1 D22.1
43 Primitive SATA_HOLD          = K28.3 D10.5 D21.6 D21.6
44 Primitive SATA_HOLDA         = K28.3 D10.5 D21.4 D21.4
45 Primitive SATA_PMACK         = K28.3 D21.4 D21.4 D21.4
46 Primitive SATA_PMNAK         = K28.3 D21.4 D21.7 D21.7
47 Primitive SATA_PMREQ_P       = K28.3 D21.5 D23.0 D23.0
48 Primitive SATA_PMREQ_S       = K28.3 D21.4 D21.3 D21.3
49 Primitive SATA_R_ERR         = K28.3 D21.5 D22.2 D22.2
50 Primitive SATA_R_IP          = K28.3 D21.5 D21.2 D21.2
51 Primitive SATA_R_OK          = K28.3 D21.5 D21.1 D21.1
52 Primitive SATA_R_RDY         = K28.3 D21.4 D10.2 D10.2
  
```

15

Generation Block Syntax

1 of 2

Generation



16

Const SMP_FRAME_TYPE_RESPONSE = 0x41 ← "Const" = Constant

```
FRAME SendSMPRptMfgInfoResponse : SendSMPFrame
{
  Prolog = SOF
  FrameType = SMP_FRAME_TYPE_RESPONSE
    Function = SMP_REPORT_MANUFACTURER_INFO
      Ignored_1A : 32
      Reserved_2A : 8
      Reserved_2B : 8
      VendorID : 64
      VendorSpecific : 160
  CRC : 32
  Epilog = EOF
}
```

← "Frame" keyword

WAIT_FOR {WF_SATA_R_OK}

← "WAIT" Command

17

Wait Conditions

- **WAIT_FOR (<event>)**
 - Exerciser will enter wait state until <event> is received
- **Events:**
 - Primitives
 - le: WAIT_FOR { WF_SATA_X_RDY }
 - Frames
 - le: WAIT_FOR { WF_OPEN_FRAME }
 - Credit
 - le : WAIT_FOR { WF_CREDIT_AVAIL }
 - OOB events

18

“SATA_Wait” Global Settings

- Wait before transmitting:
 - PMACK - WAIT_FOR_PMREQ
 - PMNAK - WAIT_FOR_PMREQ
 - R_ERR - WAIT_FOR_WTRM
 - R_IP - WAIT_FOR_SOF
 - R_OK - WAIT_FOR_WTRM
 - R_RDY - WAIT_FOR_X_RDY
 - SYNC - WAIT_FOR_SYNC
- Wait after transmitting:
 - X_RDY - WAIT_FOR_R_RDY
 - WTRM - WAIT_FOR_RCV_STATUS (= R_ERR or R_OK)
 - PMREQ_P - WAIT_FOR_PM_STATUS
 - PMREQ_S - WAIT_FOR_PM_STATUS
 - SYNC - WAIT_FOR_SYNC

19

SATA Exercise 2

Export Trace to create Exerciser Scripts



- Save time with “record - playback” capability
- Re-create problems reported in the Field
- Easily customize traffic with variations in:
 - Align bursts
 - Addressing
 - Status bits

```

149|
150|
151| SendOpenAddressFrameSSP
152| {
153|     SourceAddress = ( 500062B0 000002F5 )
154|     DestinationAddress = ( 50000000 00000001 )
155|     ArbitrationWaitTime = 0x184
156|     CompatibleFeatures = 0x0
157|     ConnectionRate = 0x9
158|     Features = 0x0
159|     InitiatorConnectionTag = 0x2
160|     InitiatorPort = 0x1
161|     MoreCompatibleFeatures = 0x0
162|     PathwayBlockedCount = 0x0
163|     # CRC = 0x3E84AF7 # good crc
164| }
165|
166| "RRDY (NORMAL)"
167| IDLE ( 1 )
168|
169| "RRDY (NORMAL)"
170|
171| SendSSPFrameCommand
172| {
173|     Retransmit = 0
174|     FillBytes = 0
175|     Tag = 0x98
176|     TargetPortTransferTag = 0xFFFF
177|     DataOffset = 0x0
178|     HashedDestinationAddress = 0x7B2777
179|     HashedSourceAddress = 0xF3463E
180|     Data = ( 00000000 00000000 00000000 A0000000 )
181|     # CRC = 0x61F1066C # good crc

```

20

SATA Exercise 3

Using *WAIT_FOR* Commands to Emulate HOST Traffic

```
1. {  
2.   Wait_For (wf_sata_x_rdy)  
3.   SATA_R_RDY  ( 4 )  
4.   SATA_CONT  
5.   Wait_For {wf_sata_SOF}  
6.   SATA_R_IP  (2)  
7.   SATA_CONT  
8.   Wait_For {wf_sata_WTRM}  
9.   SATA_R_OK  (6)  
10.  SATA_CONT  
11.}
```

21

SATA Exercise 4

- Error Injection
 - CRC Error
 - Running Disparity Error
 - Invalid Frame Type



LeCroy

Protocol Solutions Group

SAS / SATA InFusion



LeCroy

Infusion Overview

- Modifies “live” bus traffic at protocol level
 - SAS and SATA 1.5/3Gbps compatible
 - InFusion sits in-line between two PHYs
 - Drops packets, insert errors, Break link, etc..



Key Benefits

- **Real world testing**
 - Uncover fault-handling problems that may only appear under “loaded” conditions
- **Easy Wizard Interface**
 - No programming or scripting needed
- **Corner case testing**
 - Change field values within a frame to create invalid or marginal traffic
- **Hardware independent**
 - Consistent and repeatable test cases are independent of system or configuration
- **Cost-effective solution**
 - Attractive price-point ; 1.5/3Gbps SAS & SATA compatible
- **Standalone instrument**
 - Control InFusion via Built-in LCD display

25

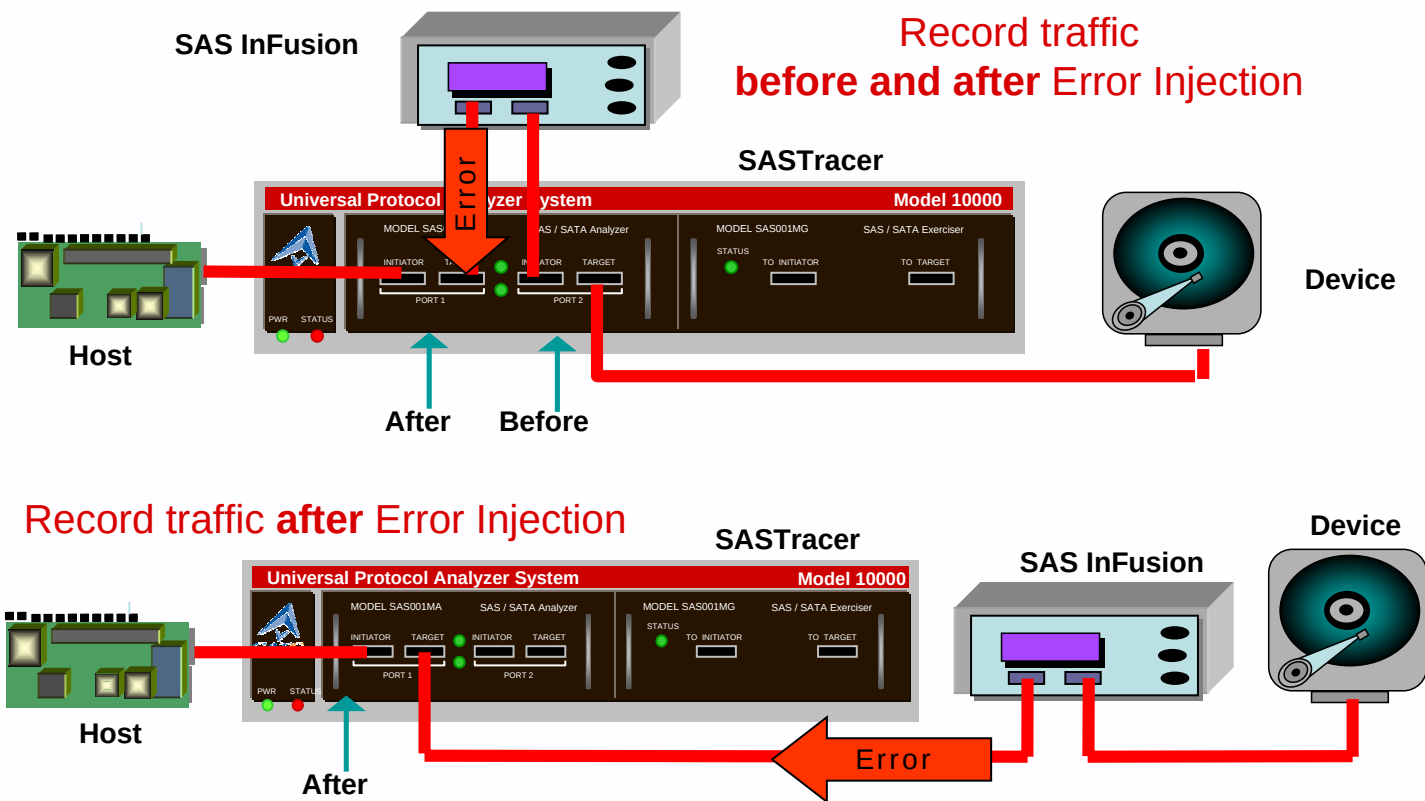
InFusion Hardware



- Dimensions: 5" x 7" x 2 ½"
- SATA single-lane connector
- Front panel controls
 - LCD (Monochrome)
 - Selection buttons
- Host-Attached or standalone operation
 - 10 test scenarios memory resident
- 10/100 BaseT interface to host PC
(USB port disabled until future release)
- BNC connectors for Trigger in / out (V1.1)
- AC power 150W

26

Using InFusion with Analyzer

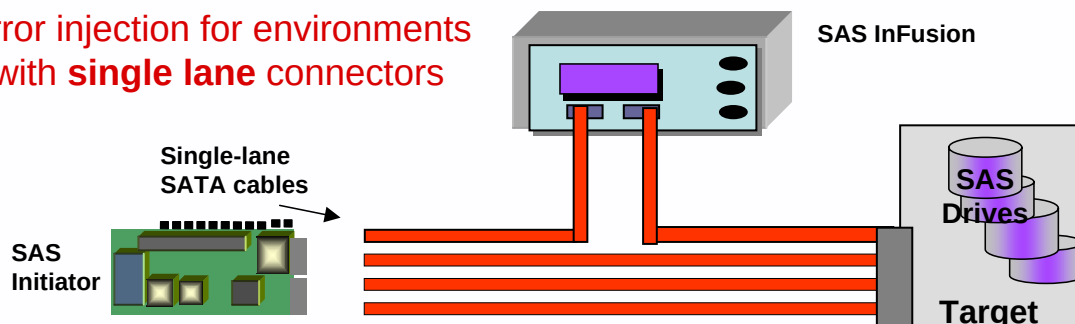


27

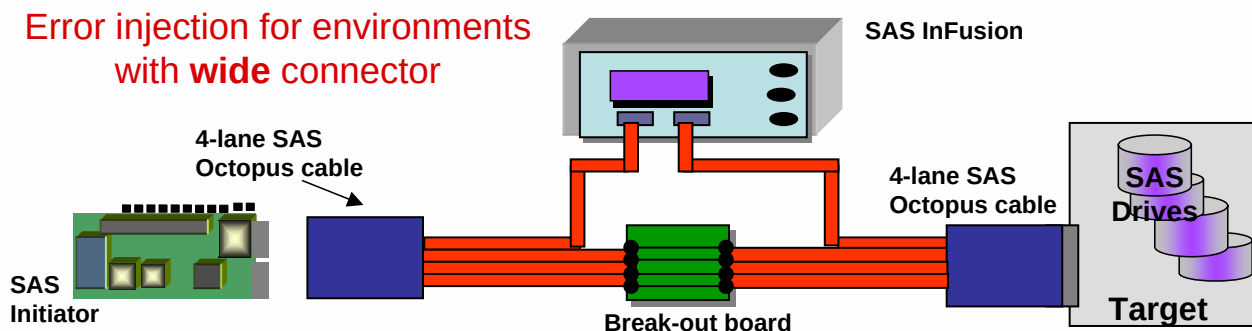
Example Test Setups

1 of 3

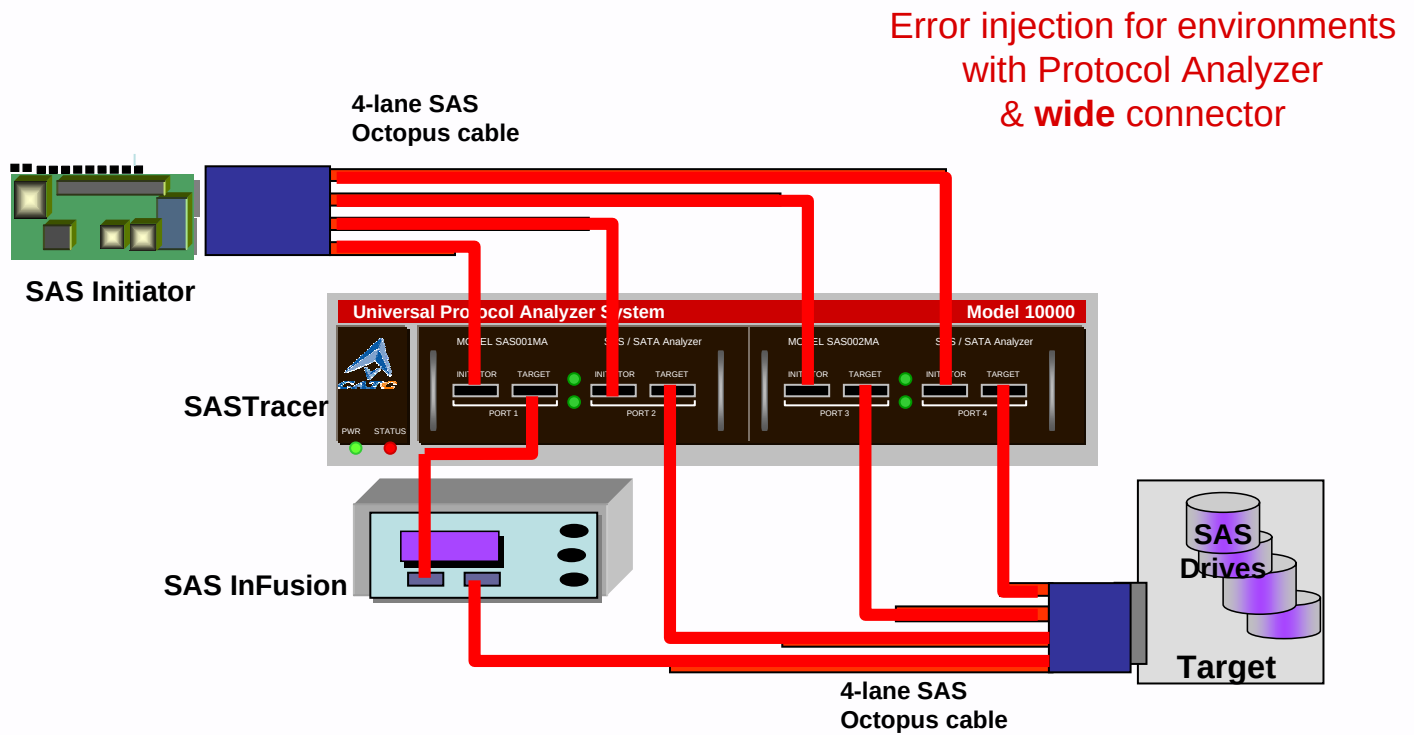
Error injection for environments with **single lane** connectors



Error injection for environments with **wide** connector

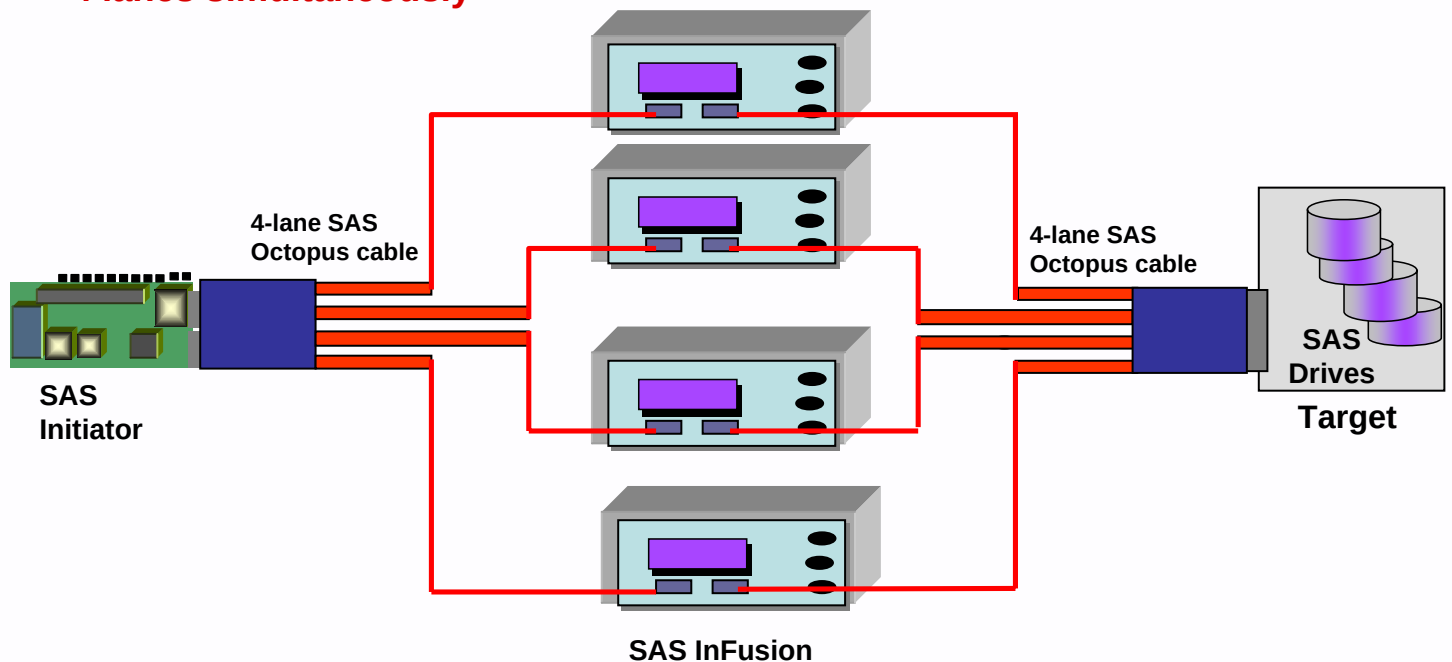


28



29

Real-time error injection on 4 lanes simultaneously



30

InFusion Capabilities

- Inject Error:
 - Insert CRC, RD or bad code
- Insert Bus Event
 - Hard Reset primitive / Disconnect link (and Reconnect)
- Remove
 - Primitive – ie: Drop “SOF” to test broken frame detection
 - Frame, FIS – ie: Drop “Status Frame” to force Task Management Query
- Substitute:
 - Primitive – ie: change “ACK” to “NAK”
 - Data pattern – ie: Change field values within a packet to create invalid protocol conditions

31

Custom Test “Scenarios”

- Test Scenarios controlled by user-defined criteria:
 - Single protocol event
 - Time-based event
 - Counter-based event
 - Combination (time, count and protocol event)
- Up to 10 test Scenarios can be resident
 - Test Scenarios can be:
 - run once
 - Loop, run sequentially (using C+ API)

32

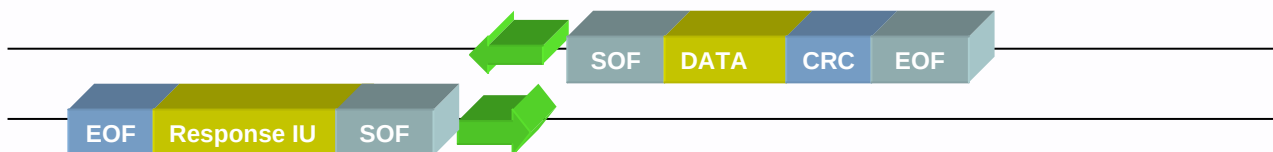
Sample Usage Cases

Every 500 <i>Queue Tag</i> = "0x8"	replace	With <i>Queue Tag</i> = "0x24"
To verify Incomplete Command recovery (Invalid Tag)		
Every 1.00 sec wait for R-RDY then	replace	With IDLE
To verify credit blocked condition		
Every 5000 XFR_RDY IU	insert	BREAK LINK
To verify Link Recovery (with pending los)		
Every 100 SOF count 7 DWORDS then	insert	EOF
To verify broken framedetection		
Every 100 Data frames - wait for ALIGN then	replace	With IDLE
To verify SYNC_ACQUIRE recovery		
Every 10.00 Sec, wait for <i>Hashed Address</i> = "xxxxx" then	replace	With <i>Hashed Address</i> = "yyyyy"
To verify OPEN_REJECT (NO DESTINATION)		
Every 500 SATA reg_host-to-dev FIS wait for R_OK then	replace	with R_ERR
to verify SATA Incomplete transaction		

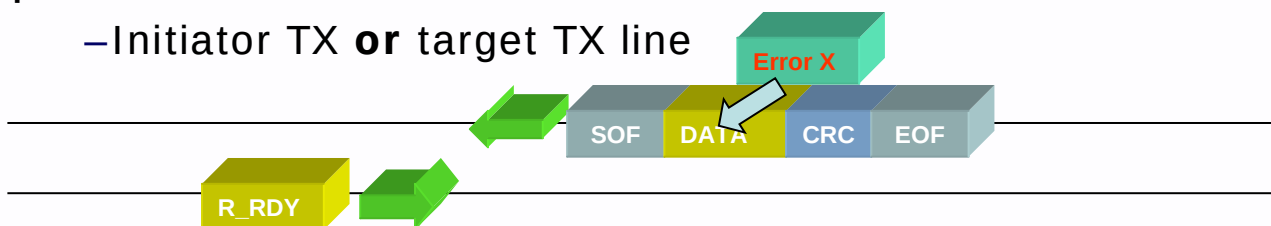
33

Operational Characteristics 1 of 2

- Traffic can be monitored from both directions
 - Initiator TX **or** target TX line



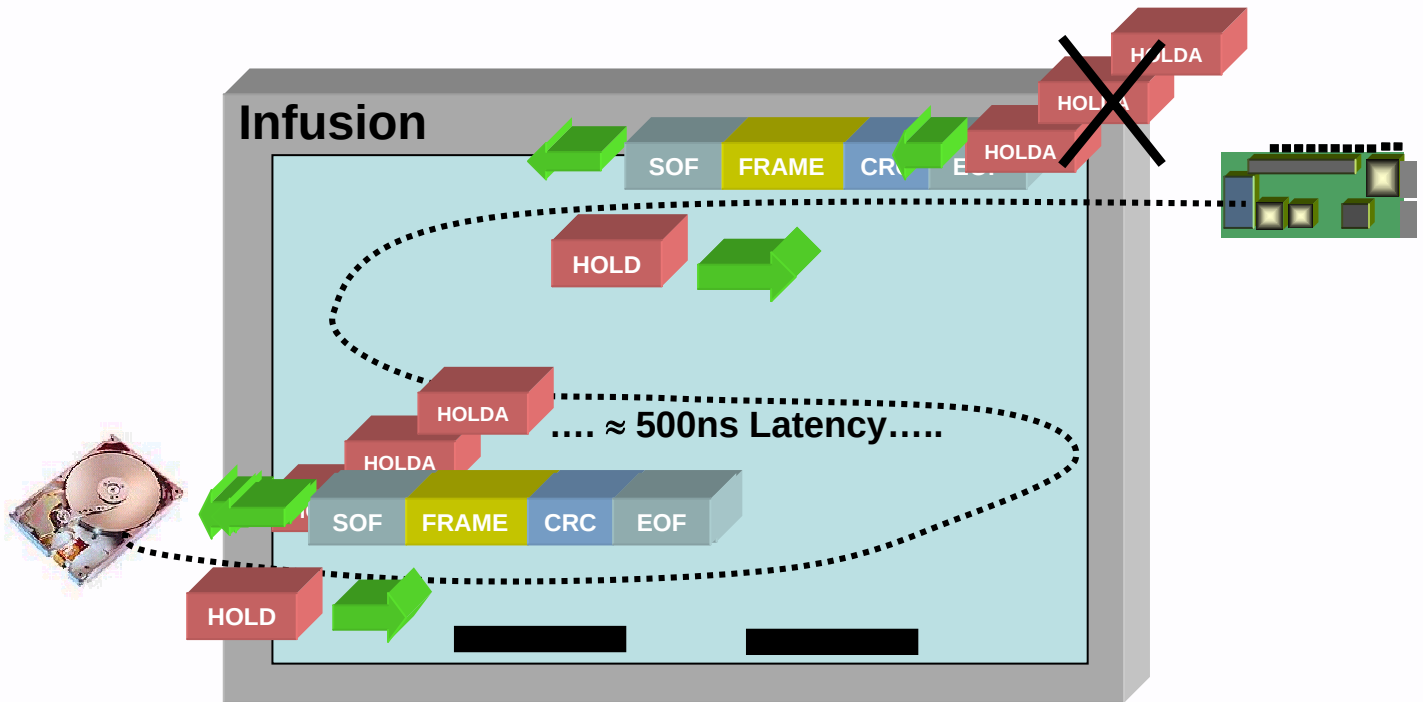
- Traffic modification can occur in only one direction per scenario
 - Initiator TX **or** target TX line



- Modification of frame contents will automatically recalculate CRC

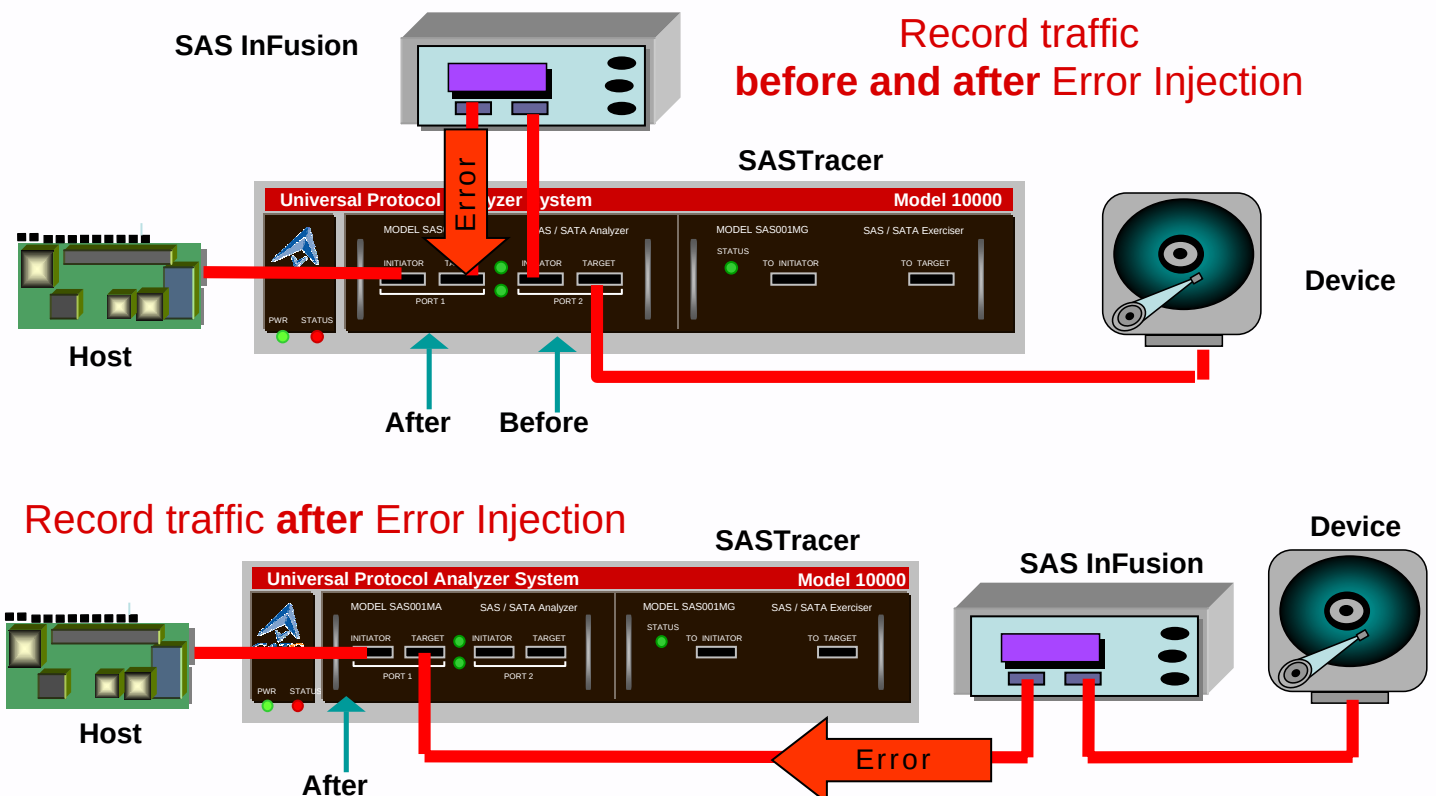
34

SATA HOLD/HOLDA



35

Using InFusion with Analyzer



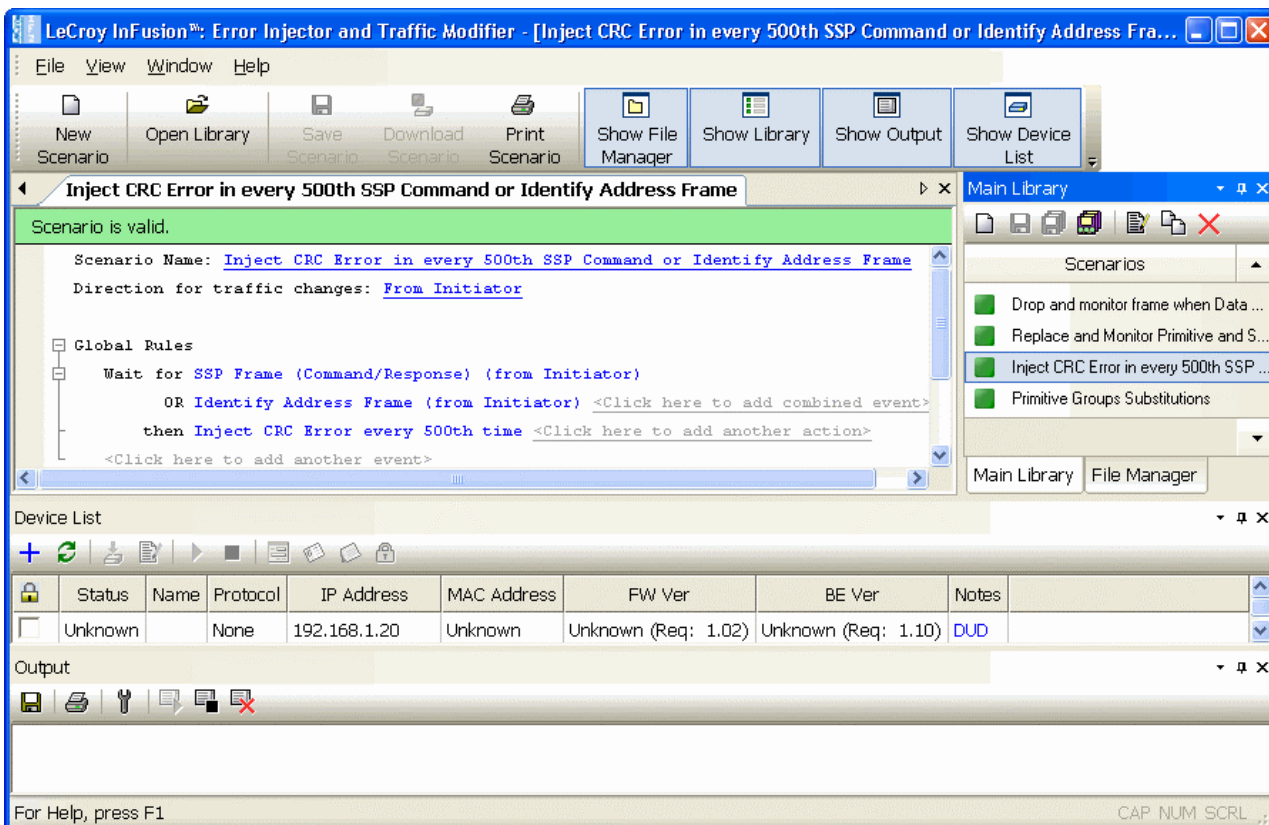
36

Controlling InFusion

- 10/100 BASE-T Ethernet port
 - Remote access over LAN or WAN
 - Use X-over cable or Gbe Router or private LAN
 - USB Interface included - *not enabled this release*
- Win2K/XP Software Client
 - Discover & control all IF boxes on LAN
 - Winsock interface
 - Create test scenarios
 - Wizard interface
 - C+ API

37

InFusion GUI Interface



38

Infusion Test Scenarios

▪ Initiator Link Layer

- Wait for OPEN ACCEPT & Remove
- Wait for OPEN ACCEPT & change to OPEN REJECT (No Destination)
- Wait for OPEN ACCEPT & change to OPEN REJECT (RETRY)
- Wait for OPEN ACCEPT & change to OPEN REJECT (STP RESOURCES BUSY)
- Wait for ACK & change to NAK
- Wait for ACK & Remove

▪ Initiator Transport Layer

- Wait for SSP Data Frame & change FRAME TYPE "01" to invalid FRAME TYPE "06"
- Wait for SSP Data Frame & change Hashed SAS Destination Address to invalid Hashed Destination Address
- Wait for SSP Frame & change Hashed SAS Hashed Source Address to invalid SAS Hashed Source Address
- Wait for SSP Data Frame & change Frame Length to incorrect length
- Wait for XFER_RDY Frame (05) & change WRITE DATA LENGTH to "0000"
- Wait for SSP Data Frame & change Frame Type to SATA DATA FIS (46)
- Wait for SSP RESPONSE IU & change all reserved bits from 0 to 1 for DWORD 6
- Wait for SSP XFER RDY IU with data offset = "0000" & Change data offset to "1010"
- Wait for SSP DATA IU with TAG_VALUE "B4" & Change frame(s) TAG_VALUE to "64"
- WAIT for SSP RESPONSE IU with STATUS = Good (00) & change to STATUS = ABORT (02)

▪ Target Link Layer

- Wait For SOAF & insert Extra SOAF
- Wait For OPEN ACCEPT & Change to OPEN REJECT (No Destination)
- Wait For OPEN ACCEPT & Remove
- Wait For OPEN ACCEPT & change to OPEN REJECT (RETRY)
- Wait For RESPONSE IU & INJECT CRC error
- Wait for RRDY & Remove

▪ Target Transport Layer

- Wait for SSP Command Frame & Change NO OF FILL BYTES
- Wait for XFER RDY & Change TPT_TAG = "B4B4"
- Wait for TASK MANAGEMENT command & Change LUN field "0000" to Invalid LUN "9999"
- Wait for ACK & Remove

39

InFusion Capabilities

- Inject Error:
 - Insert CRC, RD or invalid primitives
- Insert Bus Event
 - Hard Reset primitive / Disconnect link (and Reconnect)
- Remove
 - Primitive – ie: Drop "SOF" to test broken frame detection
 - Frame, FIS – ie: Drop "Status Frame" to force Task Management Query
- Substitute:
 - Primitive – ie: change "ACK" to "NAK"
 - Data pattern – ie: alters field values within a packet to create invalid protocol conditions

40

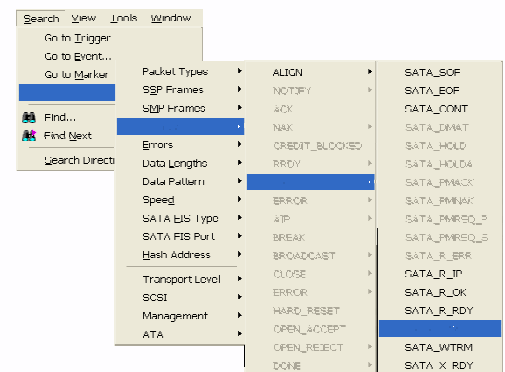
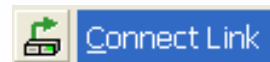
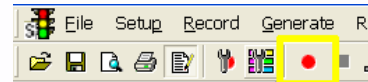
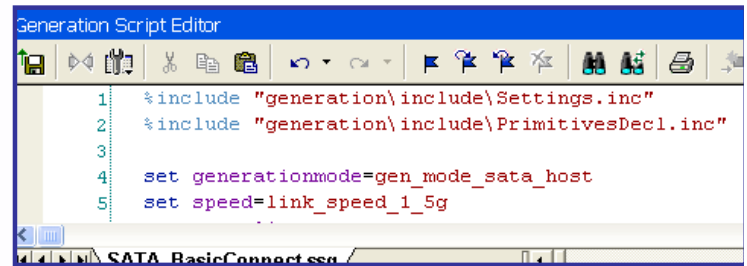
- END

41

- BACKUP

Exercise1: Basic Connecting

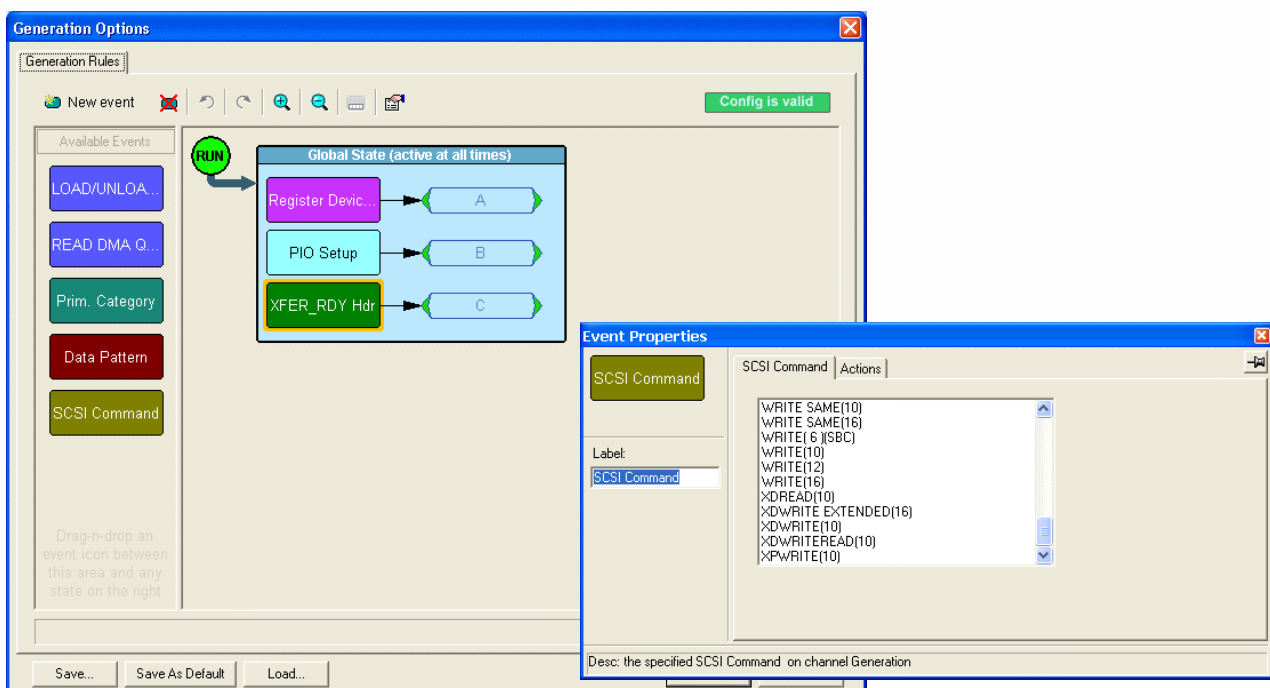
1. Create a new Traffic Gen Script (file/New), add text, save as SATA_Basic_Connect.ssg
2. Press record
3. Press Connect Icon
4. Search trace for SATA SYNC
5. Switch back to script window and press disconnect.



43

Advanced Wait Conditions

- Wait_For {WF_REC_RESOURCES_OUTPUT_A}
- Wait_For {WF_REC_RESOURCES_OUTPUT_[A-F]}



44

Global Settings Block

Tell Exerciser to automatically respond to:

– Power on sequencing:

- Set **SAS_Auto_OOB** = On
 - Responds to SAS OOB sequence automatically
- Set **SAS_Auto_SpeedNeg** = On
 - Responds to speed negotiation sequence automatically

– ALIGN sequencing

- Set **Auto_Align_SAS** = On
 - Sends Aligns every 2048 dwords
- Set **Auto_Align_SATA** = On
 - Sends Aligns every 256 dwords

45

SAS Trainer

Serial Attached SCSI Exerciser API

```

47
48  Generation
49  {
50      CONNECT
51
52      SendIdentifyAddressFrame
53      {
54          DeviceType = 0x1
55          PhyIdentifier = 0x1
56          SSPInitiatorPort = 1
57          STPInitiatorPort = 1
58          SMPInitiatorPort = 1
59          SSPTargetPort = 0
60          STPTargetPort = 0
61          SMPTargetPort = 0
62          SASAddress = { SAS_ADDRESS_INITIATOR }
63      }
64
65      wait_for { wf_identify_frame }
66

```

46

SAS speed negotiation sequence 2

•Slow-to-fast

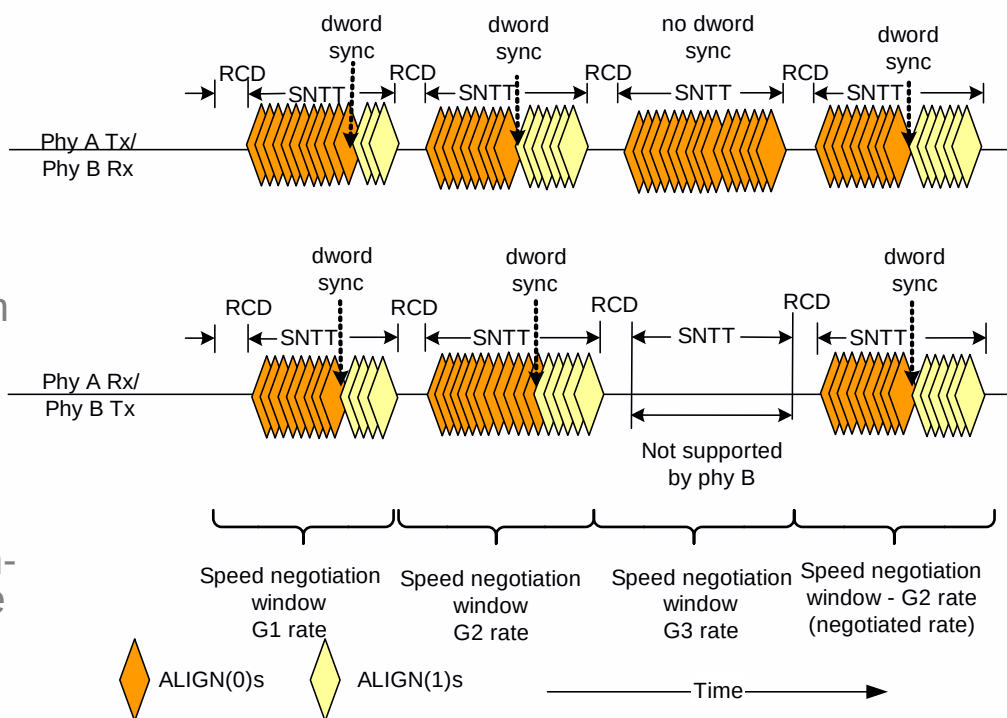
•Both phys run same set of speed negotiation windows

•1.5, then 3.0, then 6.0 (if needed), etc. until they find:

–a supported rate; then

–a (faster) non-supported rate

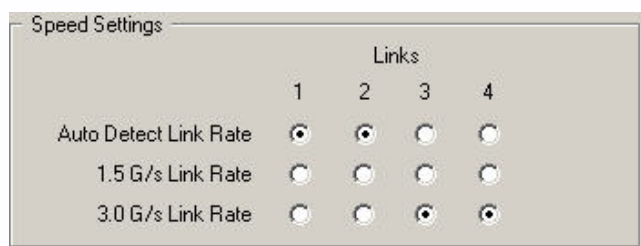
•Last window returns to the highest supported rate detected



47

SASTracer & Speed Negotiation

Choose Autodetect in Recording options – SASTracer allows either 1.5Gbps and 3Gbps negotiation



11	Frame 115	1.5 G	Unknown	Idle	Time Stamp
				0.000 ns	47.709 921 862
11	Frame 116	1.5 G	ALIGN (1) K28.5 D07.0 D07.0 D07.0	Sequence 3	Idle Time Stamp
				0.000 ns	47.709 921 887
11	Frame 117	1.5 G	Unknown	Idle	Time Stamp
				0.830 ns	47.709 921 967
11	Frame 118	1.5 G	ALIGN (1) K28.5 D07.0 D07.0 D07.0	Sequence 3	Idle Time Stamp
				40.000 ns	47.709 921 995
11	Frame 119	3 G	Unknown	Idle	Time Stamp
				0.000 ns	47.709 922 115
11	Frame 120	3 G	ALIGN (1) K28.5 D07.0 D07.0 D07.0	Idle	Time Stamp
				0.000 ns	47.709 922 127
11	Frame 121	3 G	ALIGN (1) K28.5 D07.0 D07.0 D07.0	Sequence 2	Idle Time Stamp
				0.830 ns	47.709 922 140
11	Frame 122	3 G	Unknown	Idle	Time Stamp
				0.000 ns	47.709 922 167
11	Frame 123	3 G	Unknown	Idle	Time Stamp
				1.660 ns	47.709 922 180

Speed Negotiates to 3Gbp

48

IDENTIFY Sequence

- After speed Negotiation - each link exchanges IDENTIFY frames
- Informs Initiator of all attached SAS addresses

↓ IDENTIFY from Initiator

SOAF			
Identify Type: 10	Restricted: 00	Initiator Ports: 0E	TargetPorts: 00
Restricted: 00000000			
Restricted: 00000000			
SAS Address: 500062B0			
SAS Address: 000002F5			
Phy ID: 01	Reserved: 000000		
Reserved: 00000000			
CRC: 9A4B1FE7			
EOAF			

Targets don't do much with IDENTIFY data from Initiator

Initiator uses IDENTIFY Data from target to perform DISCOVER

↑ IDENTIFY from Target

SOAF			
Identify Type: 10	Restricted: 00	Initiator Ports: 00	TargetPorts: 08
Restricted: 00000000			
Restricted: 00000000			
SAS Address: 50000000			
SAS Address: 00000001			
Phy ID: 00	Reserved: 000000		
Reserved: 00000000			
CRC: A9E90668			
EOAF			

49

"SAS_Wait" Global Settings

- Wait after transmitting:
 - Close – Wait For Close
 - EOF – Wait For ACK/NAK
 - Open Frame - Wait for Open_Accept
 - Open Frame - Wait for Open_Reject(X)
 - Identify Frame – Wait for Identify Frame
 - SMP_Request – Wait for SMP Response
- Wait before transmitting:
 - Identify Frame – Wait for Identify Frame
 - Open_Accept - Wait for Open Frame
 - Open_Reject(x) - Wait for Open Frame
 - AIP - Wait for Open Frame
 - SMP_Response Wait for SMP Request

50

Packets and Frame

```

1. Packet SendSSPFrame
2. {
3.   Prolog = SOF
4.   #####
5.   FrameType           : 8
6.   HashedDestinationAddress : 24
7.   Reserved_A          : 8
8.   HashedSourceAddress   : 24
9.   Reserved_B          : 8
10.  Reserved_C          : 8
11.  Reserved_D          : 6
12.  Retransmit          : 1
13.  Reserved_E          : 1
14.  Reserved_F          : 6
15.  FillBytes           : 2
16.  Reserved_G          : 32
17.  Tag                 : 16
18.  TargetPortTransferTag : 16
19.  DataOffset          : 32
20.  #####
21.  Data                 : *
22.  #####
23.  CRC                  : 32
24.  #####
25.  Epilog = EOF
26. };

```

Keyword
Table Field

Table 93 — SSP frame format

Byte/Bit	7	6	5	4	3	2	1	0
0	FRAME TYPE							
1	(MSB)	HASHED DESTINATION SAS ADDRESS						(LSB)
3								
4	Reserved							
5	(MSB)	HASHED SOURCE SAS ADDRESS						(LSB)
7								
8	Reserved							
9	Reserved							
10	Reserved					RETRANSMIT		Reserved
11	Reserved					NUMBER OF FILL BYTES		
12	Reserved							
13		Reserved						
15								
16	(MSB)	TAG						(LSB)
17								
18	(MSB)	TARGET PORT TRANSFER TAG						(LSB)
19								
20	(MSB)	DATA OFFSET						(LSB)
23								
24								
m								
	Fill bytes, if needed							
n - 3	(MSB)	CRC						(LSB)
n								

51

SASTrainer Conditional Branching:

Branching logic is critical for Drive Emulation

—Allows intelligent responses to commands from Initiator ie:

IF WRITE 10

Then Send RRDY

ELSE

IF READ 10

Then Send DATA...

WAIT (5000) {

```

when { WF_ACK WF_SATA_SYNC } do
{
  ACK (16)
  IDLE ( 800 )
  "AIP (NORMAL)"
}
elsewhen { WF_EOAF } do
{
  SendSMPPrptMfgInfoRequest
  "RRDY (NORMAL)"
  SendSMPPrptMfgInfoResponse
}
elsewhen { WF_EOAF WF_SATA_X_RDY } do
{
  "RRDY (NORMAL)"
  PAUSE
}
on_timeout
{
  SATA_HOLD (16)
  BREAK
}
}

```

52

SubClass

```

1  %include "Generation\Include\Settings.inc"
2  %include "Generation\Include\PrimitivesDecl.inc"
3  %include "Generation\Include\WaitCommands.inc"
4
5  Packet MyBaseFrame
6  {
7      Prolog = SOF
8      FrameType      : 8 = 1
9      HashedAddress : 24
10     Data           : *
11     CRC            : 32
12     Epilog = EOF
13 }
14
15 Packet MyFrameType2:MyBaseFrame
16 {
17     FrameType = 7
18 }
19
20 Generation
21 {
22     MyBaseFrame { DATA = {01020304 }}
23     MyFrameType2 { DATA = {05060708 091011}}
24     MyBaseFrame { FrameType=0x11 DATA = {05060708 091011}}
25 }

```

BEGIN									
13	Frame	1.5	Incomplete	SOF	Data	CRC	EOF	Idle	Time Stamp
16	16	G	SSP FRAME		01 00 00 00 01 02 03 04	0x26FBF03E		0.000 ns	00.000 000 427
13	Frame	1.5	Incomplete	SOF	Data	CRC	EOF	Idle	Time Stamp
17	17	G	SSP FRAME		07 00 00 00 05 06 07 08 00 09 10 11	0xEB66AB26		0.000 ns	00.000 000
13	Frame	1.5	Incomplete	SOF	Data	CRC	EOF	Time Stamp	
18	18	G	SSP FRAME		11 00 00 00 05 06 07 08 00 09 10 11	0xF72E8EC5		00.000 000 720	

53

SASTrainer: Key Points

- **Integrated Analyzer / Exerciser platform**
 - Small, convenient system can Transmit/Analyze SAS & SATA
- **Export Trace to create Exerciser Scripts**
 - Save time writing scripts with “record - playback” capability
- **Automatic Protocol Handshaking**
 - Automatic ALIGN and OOB handshaking; plus Wait Events make sophisticated emulation scripts possible
- **Unlimited Branching**
 - Allows intelligent responses to commands from Initiator;
- **LeCroy platform Upgradeable to Support 4 wide Analysis**
 - Modular system is cost effective and flexible

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at [artisanTG.com](https://www.artisanTG.com) for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisanTG.com | [artisanTG.com](https://www.artisanTG.com)

