CATC SAS001MA

SAS / SATA Analyzer

In Stock

Used and in Excellent Condition

Open Web Page

https://www.artisantg.com/79361-5

LeCroy

Protocol Solutions Group

SASTracer / Trainer

SAS - SATA Protocol Analyzer & Exerciser

September 1, 2005

Michael Micheletti
Sr. Product Marketing Manager

LJDN-ST-CA-0241-0001

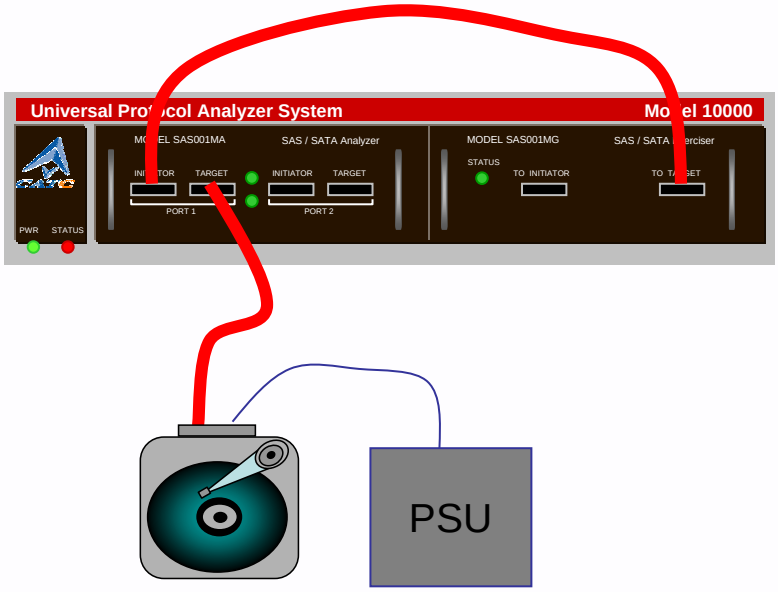
LeCroy

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



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)
}
```

Include files: Settings.inc

Two Approaches:

1) Auto-Mode

- Automatically uses Default settings from Specifications — *OOB, Speed Neg, ALIGNs*

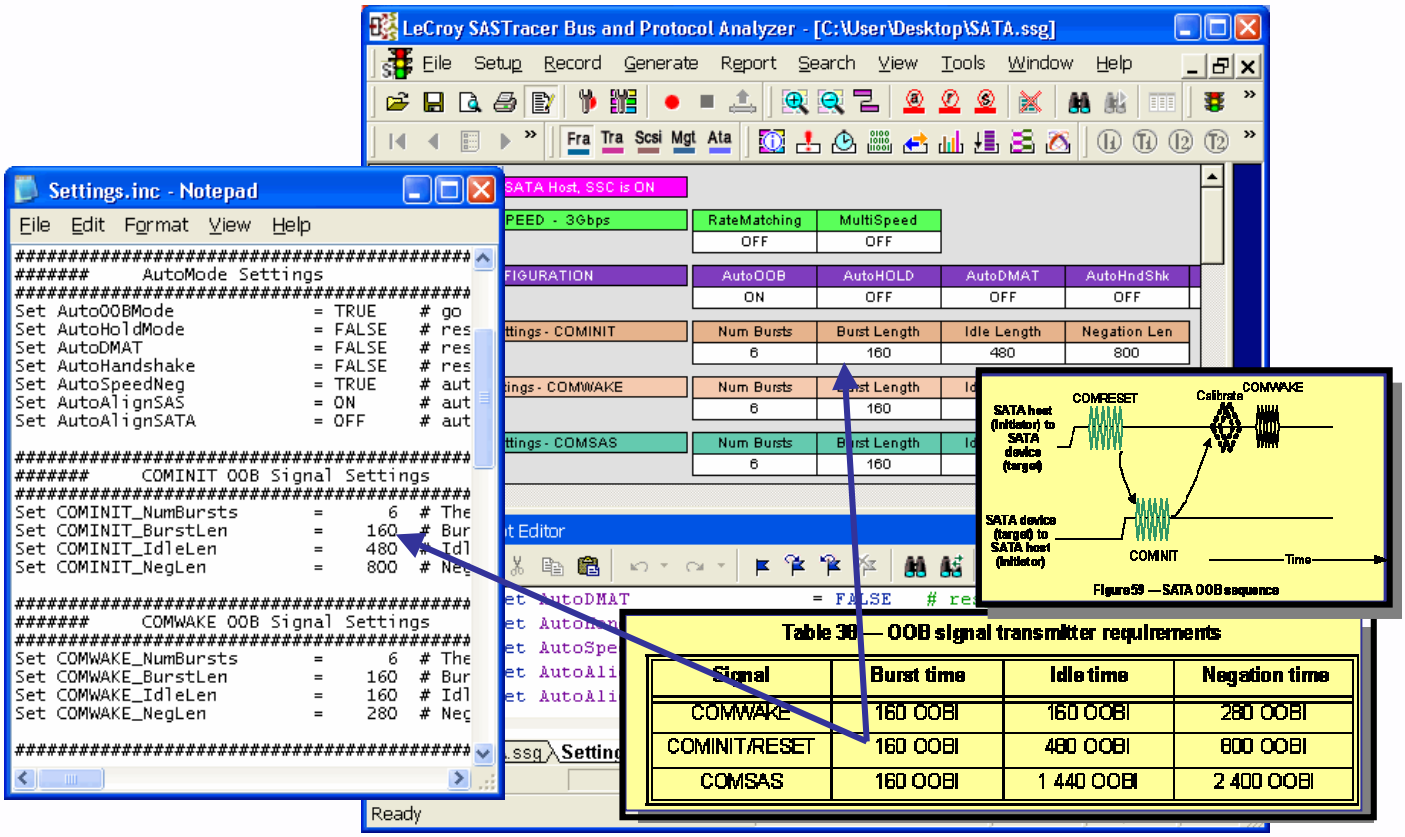
2) Manual mode

- COMINIT/COMWAKE OOB Signal — *Burst/Idle Intervals, etc..*
- SATA Link Initialization
- SAS/SATA Speed Negotiation — *SNLT, SNTT timing, etc..*
- Autowait — *WAIT After, WAIT Before*
- Wait Command Timeout
- Scrambling Mode

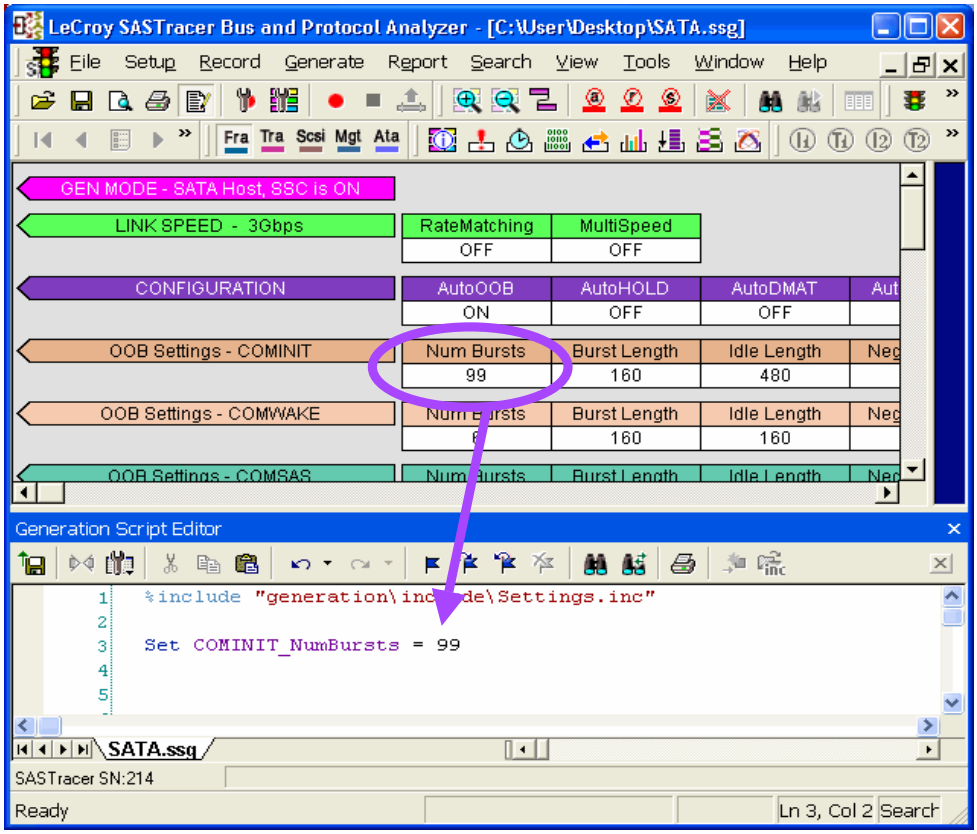
Plus....

- Emulation mode — *HOST or DEVICE*
- Set Speed — *1.5G or 3G*

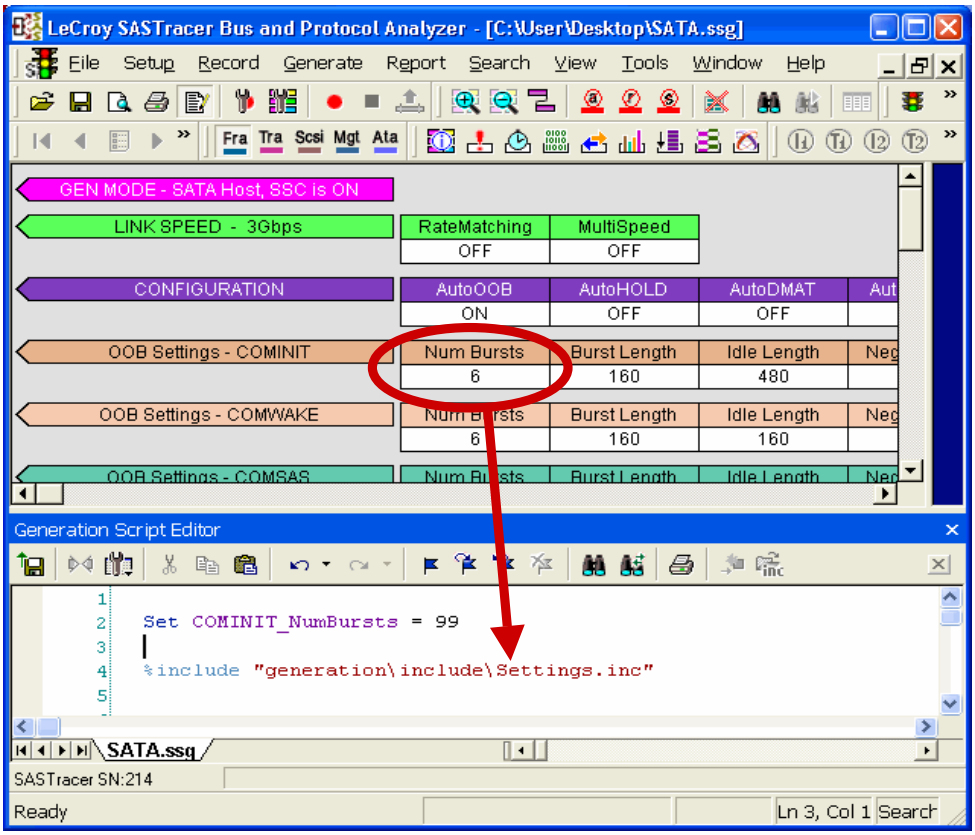
Configuration



Precedence



Precedence



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 =

AutoAlign settings

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

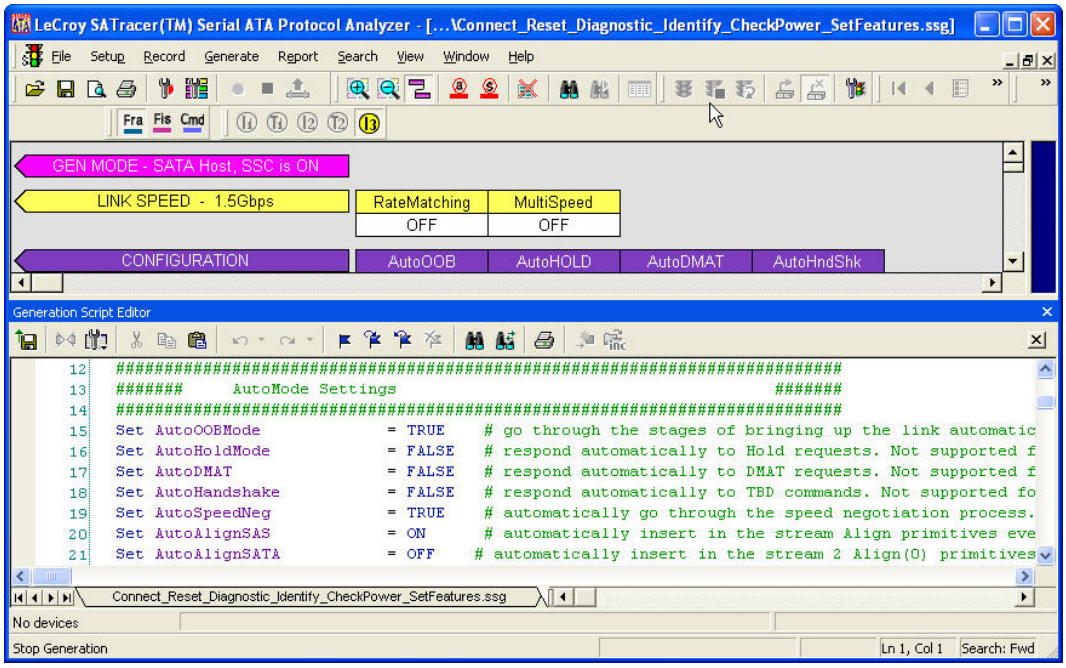
The Global Setting "AutoAlign"

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

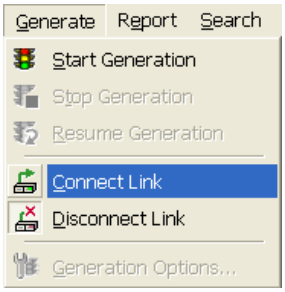
SATA Exercise 1

Change Settings



Bringing the link up

Three Ways to Transmit OOB



1. Icon/Menu macros

2. Automated in a Script
   - `CONNECT` keyword

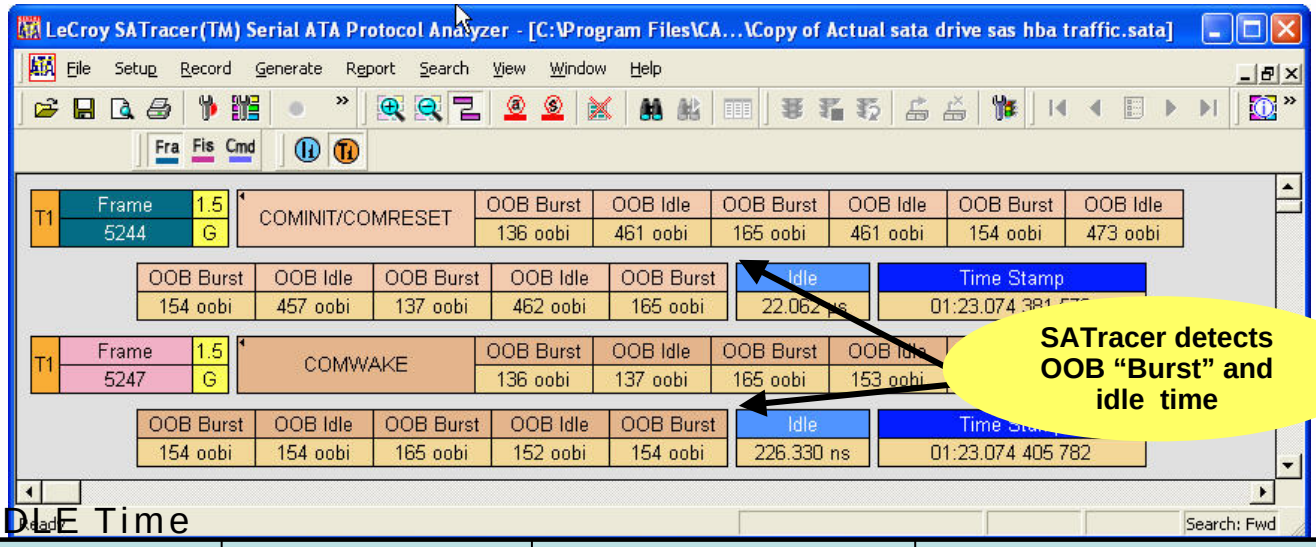
3. Manually in a Script
   - `Set AutoOOBMode = FALSE`
   - `Set AutoSpeedNeg = FALSE`
   - `Set AutoAlignSAS = OFF`
   - `Set AutoAlignSATA = OFF`
   - *`COMINIT`*
   - *`IDLE(800)`*
   - *`COMWAKE`*
   - *`IDLE(80)`*

Using Commands to transmit OOB sequence

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)

OOB Detection

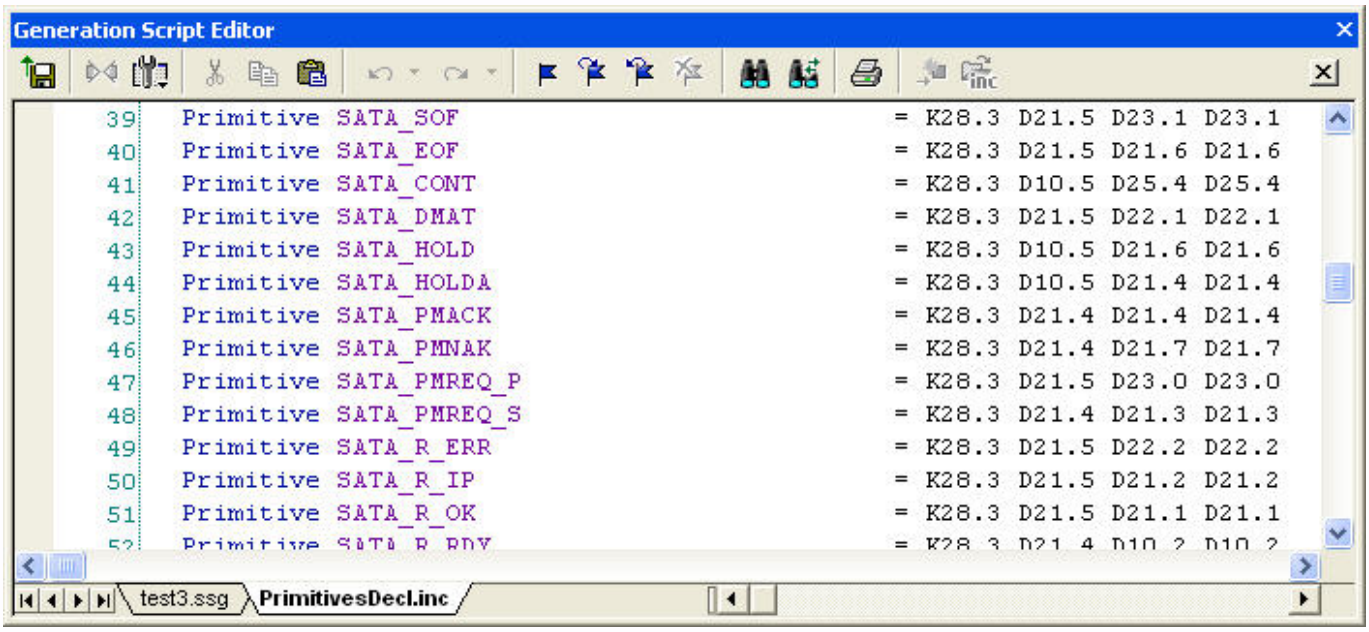


IDLE Time

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

Primitives

DWORD values Predefined in *PrimitivesDecl.inc* file



Generation Block Syntax 1 of 2

```
Generation
{
    SATA_R_RDY   ( 4 )

    SATA_CONT

      SATA_SOF
        SATA_DATA   ( 0x27809000 )
        SATA_DATA   ( 0x01010000 )
        …

    IDLE (10)
}
```

Send 4 R_RDY primitives

Repeat XXXX data

Send any hex pattern (DWORD)

IDLE on line for 10ns

Generation Block Syntax 2 of 2

```
Const SMP_FRAME_TYPE_RESPONSE        = 0x41

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
}

 WAIT_FOR {WF_SATA_R_OK}
```

"Const" = Constant

"Frame" keyword

"WAIT" Command

Wait Conditions

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

"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

SATA Exercise 2

Export Trace to create Exerciser Scripts

=

```
149
150     SendOpenAddressFrameSSP
151     (
152         SourceAddress       = ( 500062B0 000002F5 )
153         DestinationAddress = ( 50000000 00000001 )
154         ArbitrationWaitTime = 0x184
155         CompatibleFeatures = 0x0
156         ConnectionRate = 0x9
157         Features = 0x0
158         InitiatorConnectionTag = 0x2
159         InitiatorPort = 0x1
160         MoreCompatibleFeatures = 0x0
161         PathwayBlockedCount = 0x0
162       # CRC = 0x3EA84AF7  # good crc
163     )
164     IDLE ( 24 )
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
```

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

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.}
```


SATA Exercise 4

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

LeCroy

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..

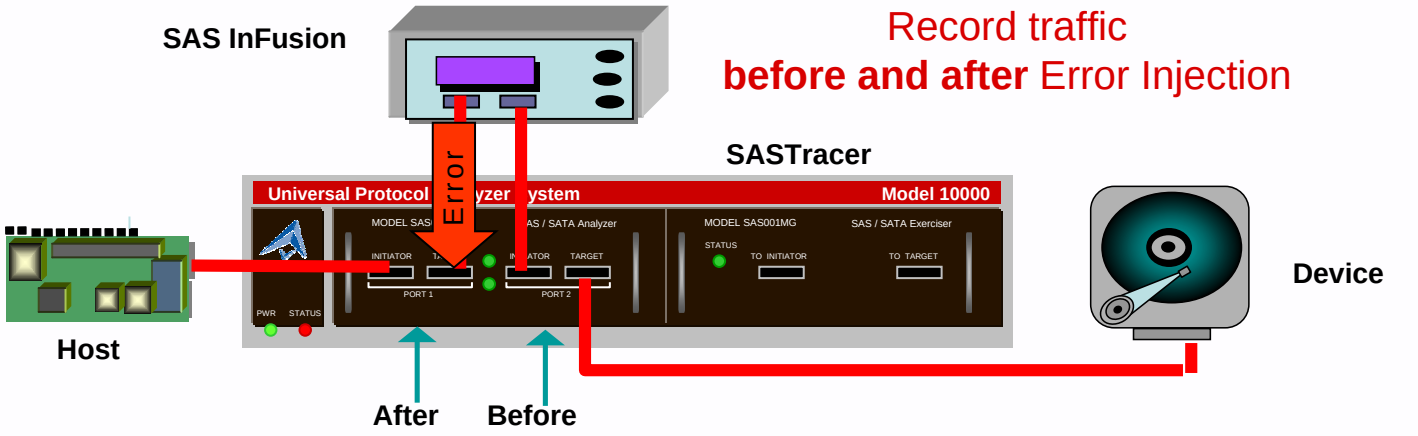
Key Benefits

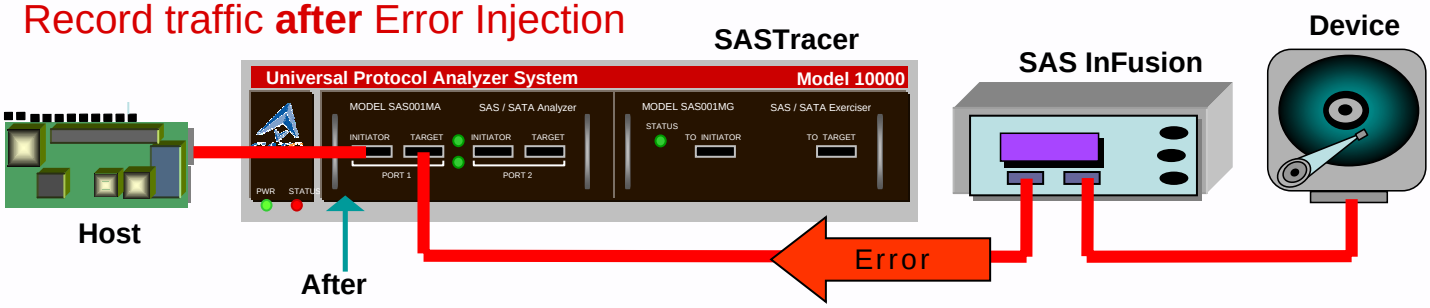
- **Real world testing**
  - Uncover fault-handling problems that may only appear under “loaded” conditions
- **Easy Wizard Interface**
  - No programming or scripting needed
- **Corner case testing**
  - Change field values within a frame to create invalid or marginal traffic
- **Hardware independent**
  - 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

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

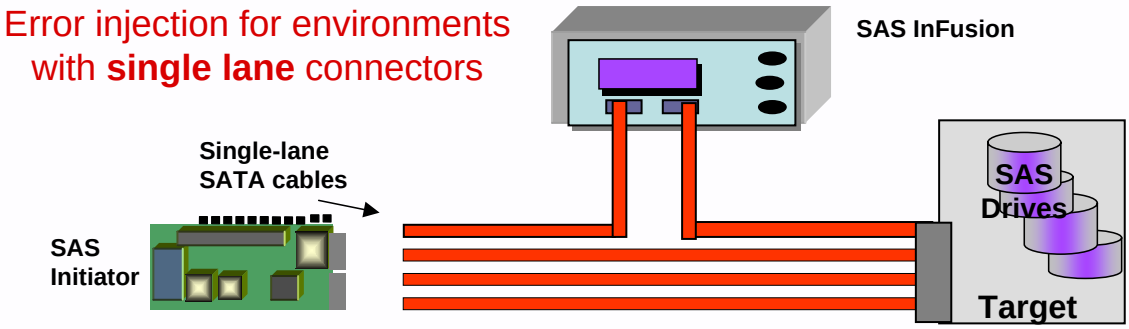
Using InFusion with Analyzer

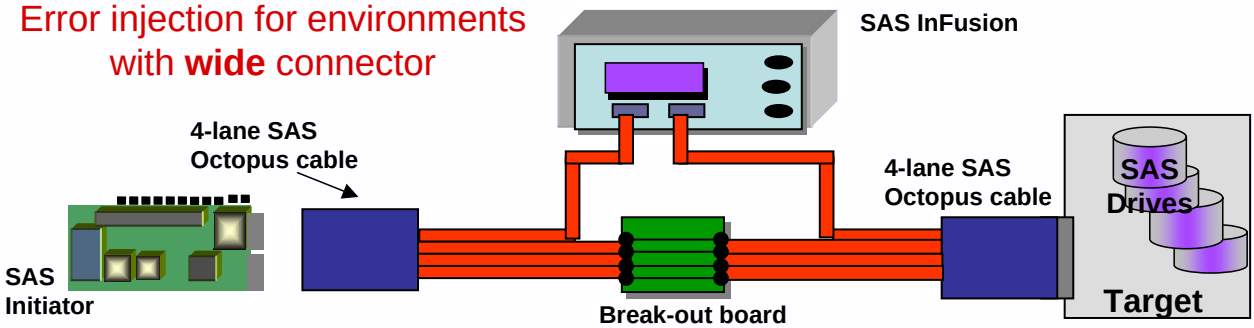




Example Test Setups

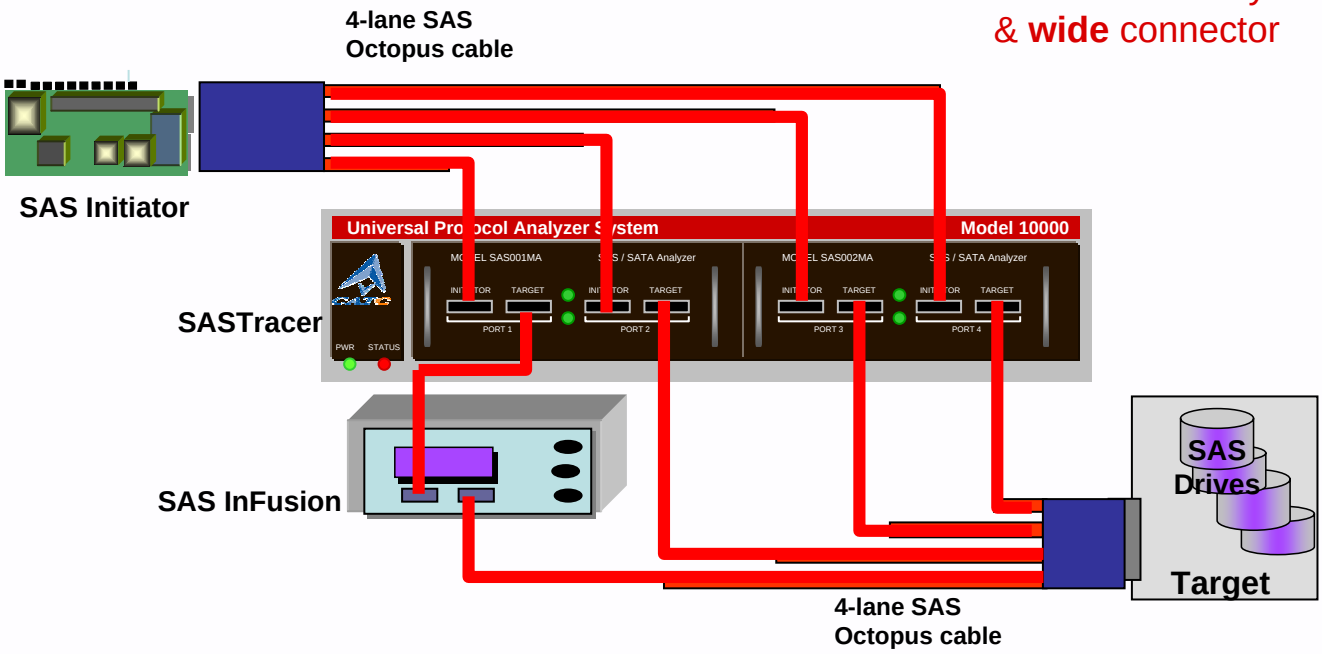
1 of 3





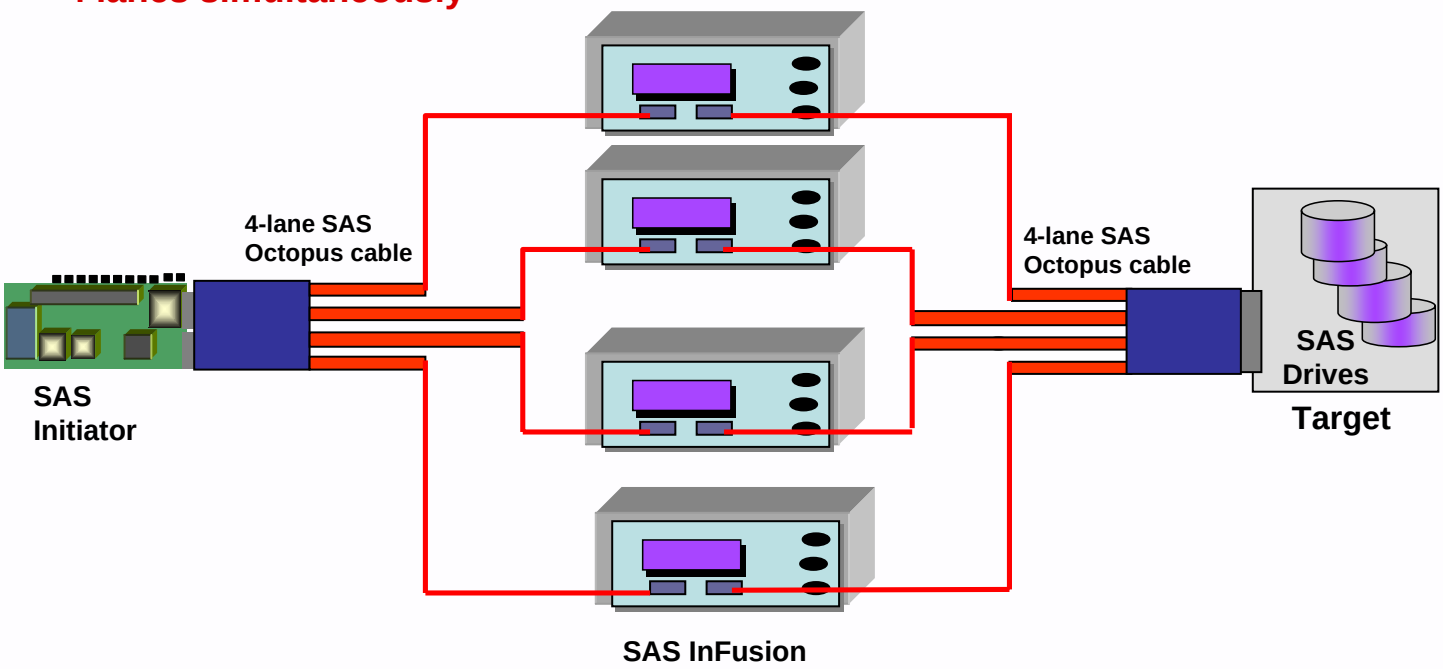
Example Test Setups

Error injection for environments
with Protocol Analyzer
& **wide** connector



Example Test Setups

**Real-time error injection on
4 lanes simultaneously**



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

Custom Test “Scenarios”

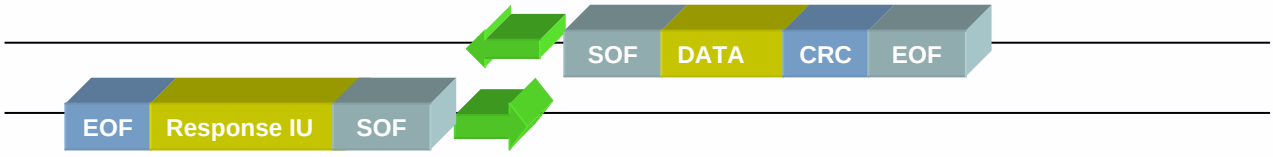
- Test Scenarios controlled by user-defined criteria:
  - Single protocol event
  - Time-based event
  - Counter-based event
  - Combination (time, count and protocol event)
- Up to 10 test Scenarios can be resident
  - Test Scenarios can be:
    - run once
    - Loop, run sequentially (using C+ API)

Sample Usage Cases

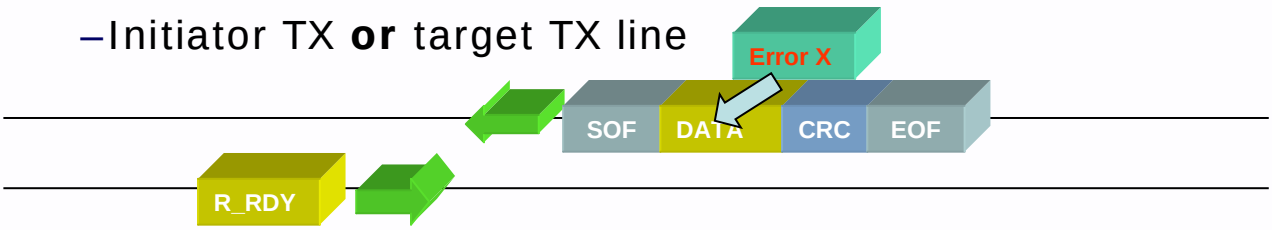
Every 500 *Queue Tag* = "*0x8*"	replace	*With Queue Tag* = "*0x24*"
To verify Incomplete Command recovery (Invalid Tag)		
Every 1.00 sec wait for R-RDY then	replace	With IDLE
To verify credit blocked condition		
Every 5000 XFR_RDY IU	insert	BREAK LINK
To verify Link Recovery (with pending los)		
Every 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 *Hashed Address* = "xxxxx" then	replace	With *Hashed Address* = "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		

Operational Characteristics 1 of 2

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



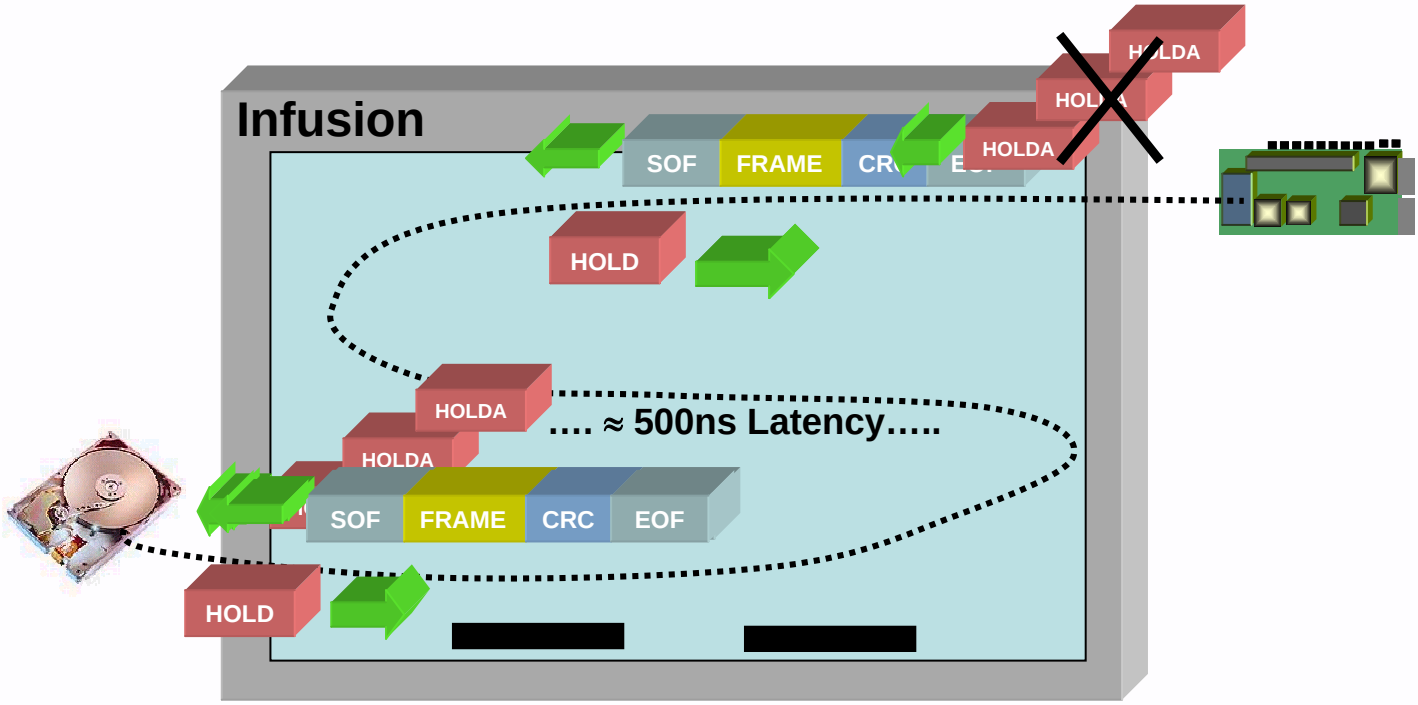
- Traffic modification can occur in only <u>one</u> direction per scenario
  - Initiator TX **or** target TX line



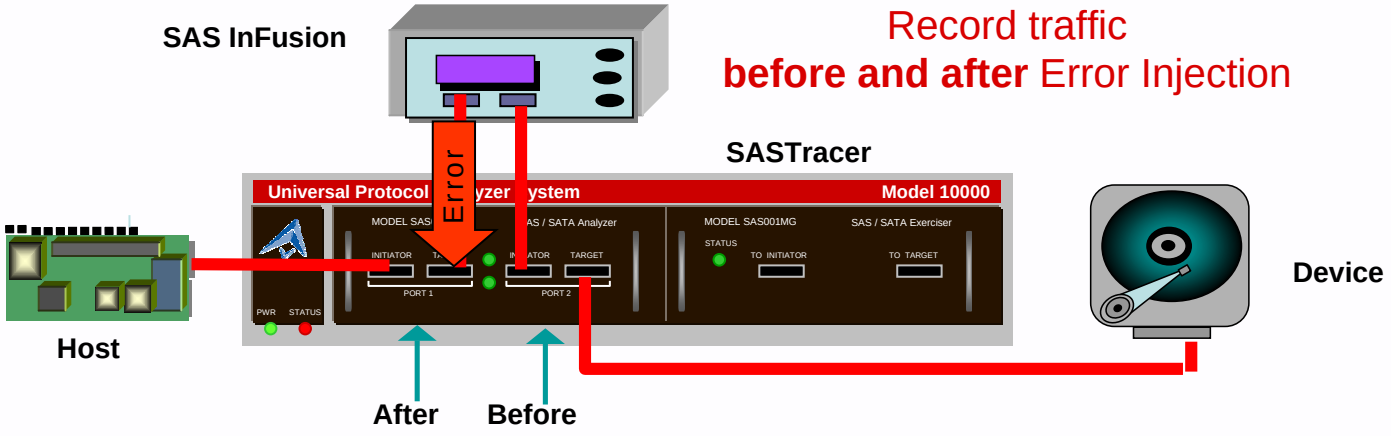
- Modification of frame contents will automatically recalculate CRC

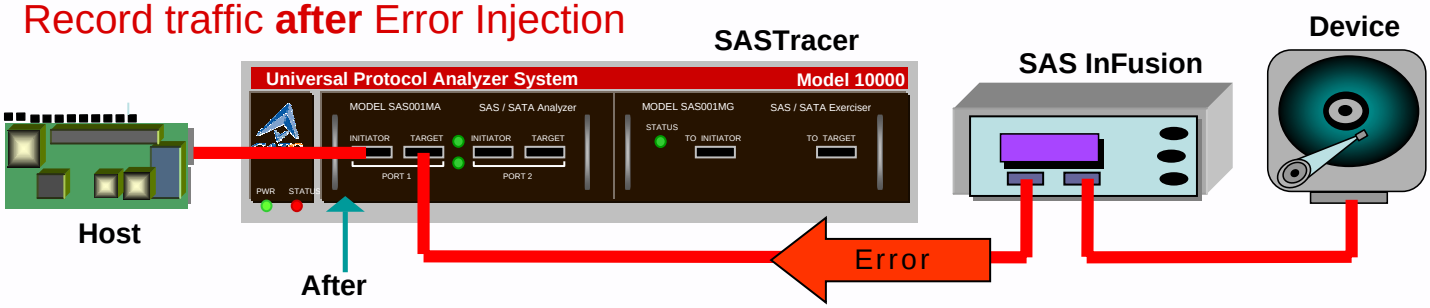
Operational Characteristics 2 of 2

SATA HOLD/HOLDA



Using InFusion with Analyzer

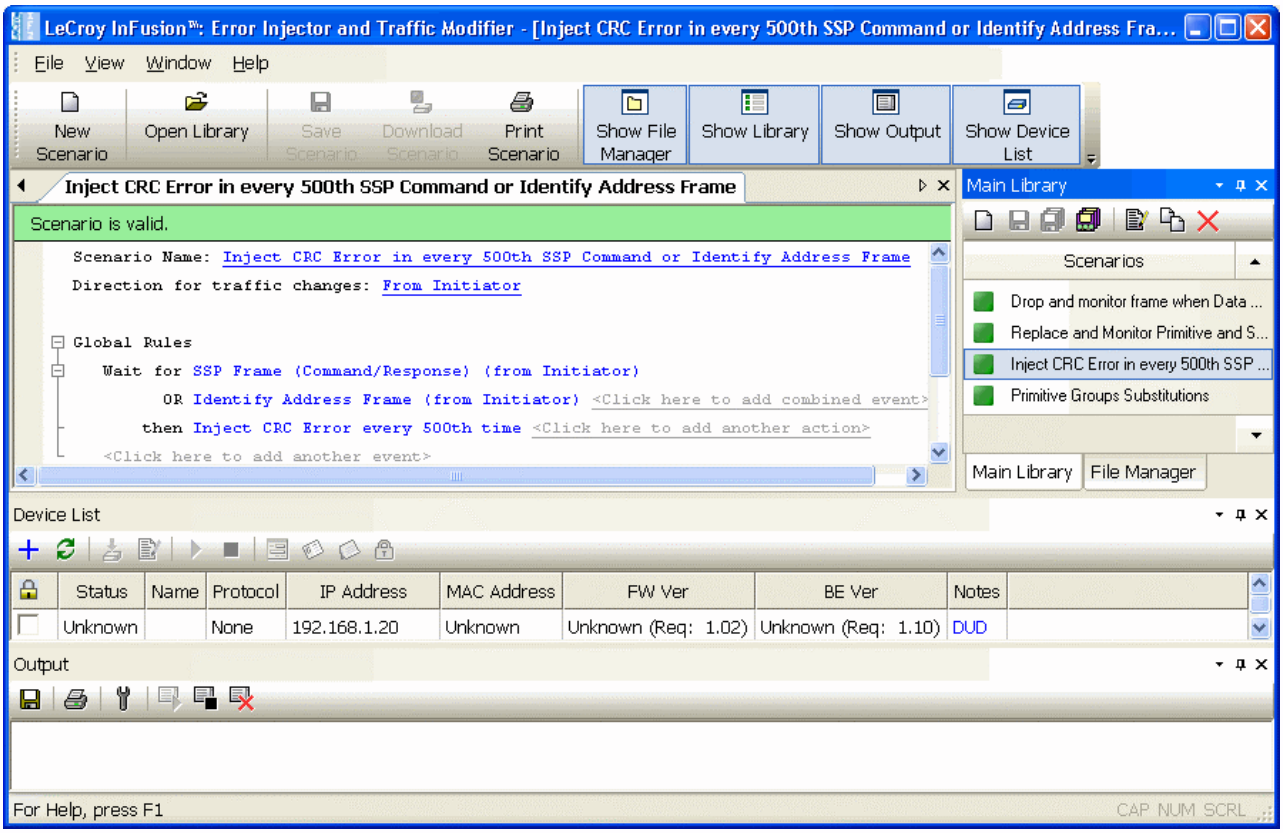




Controlling InFusion

- 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

InFusion GUI Interface



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

InFusion Capabilities

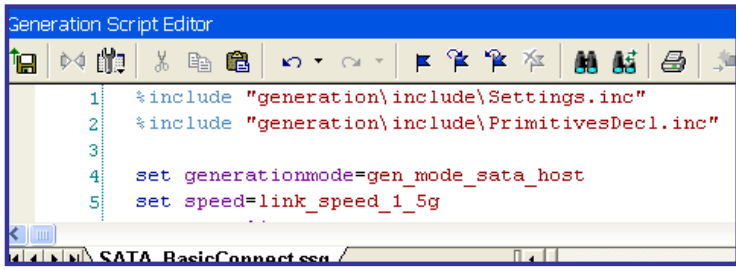
- Inject Error:
  - Insert CRC, RD or invalid primitives
- Insert Bus Event
  - Hard Reset primitive / Disconnect link (and Reconnect)
- Remove
  - Primitive – ie: Drop “SOF” to test broken frame detection
  - Frame, FIS – ie: Drop “Status Frame” to force Task Management Query
- Substitute:
  - Primitive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

- END

- BACKUP

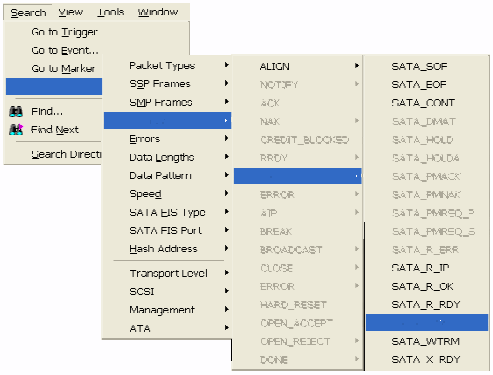
Exercise1: Basic Connecting

1. Create a new Traffic Gen Script (file/New), add text, save as SATA_Basic_Connect.ssg



2. Press record
3. Press Connect Icon

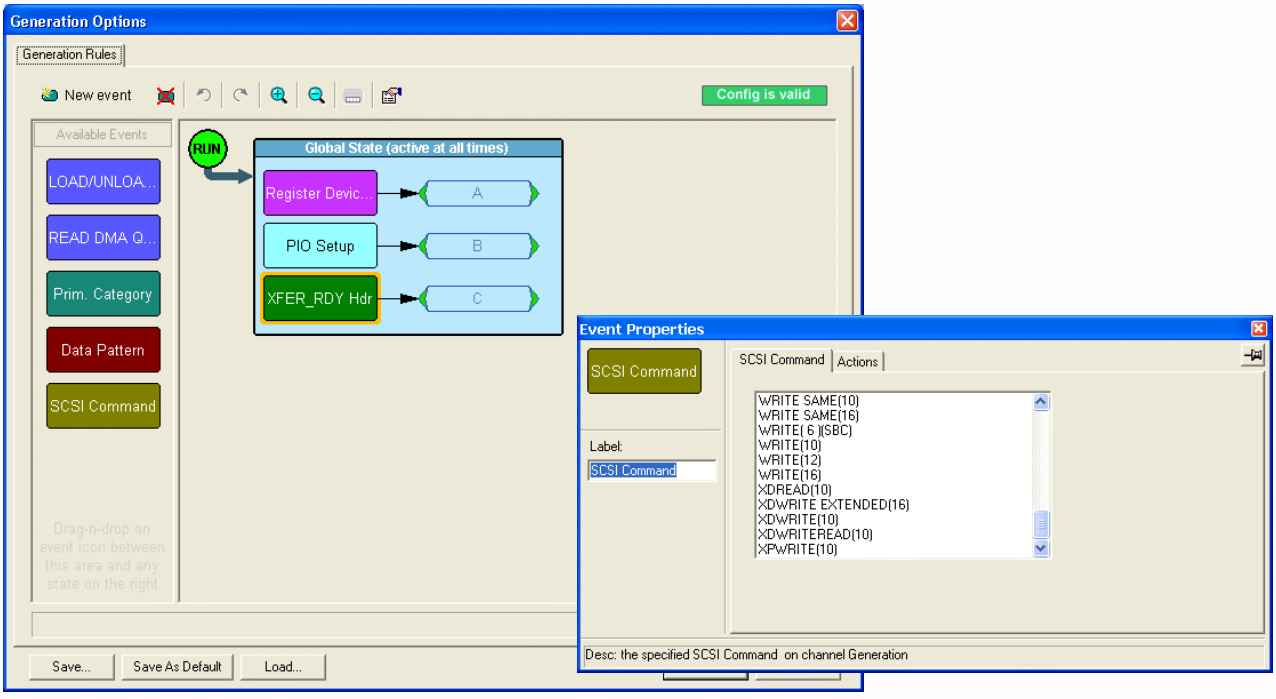
4. Search trace for SATA SYNC



5. Switch back to script window and press disconnect.

Advanced Wait Conditions

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



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

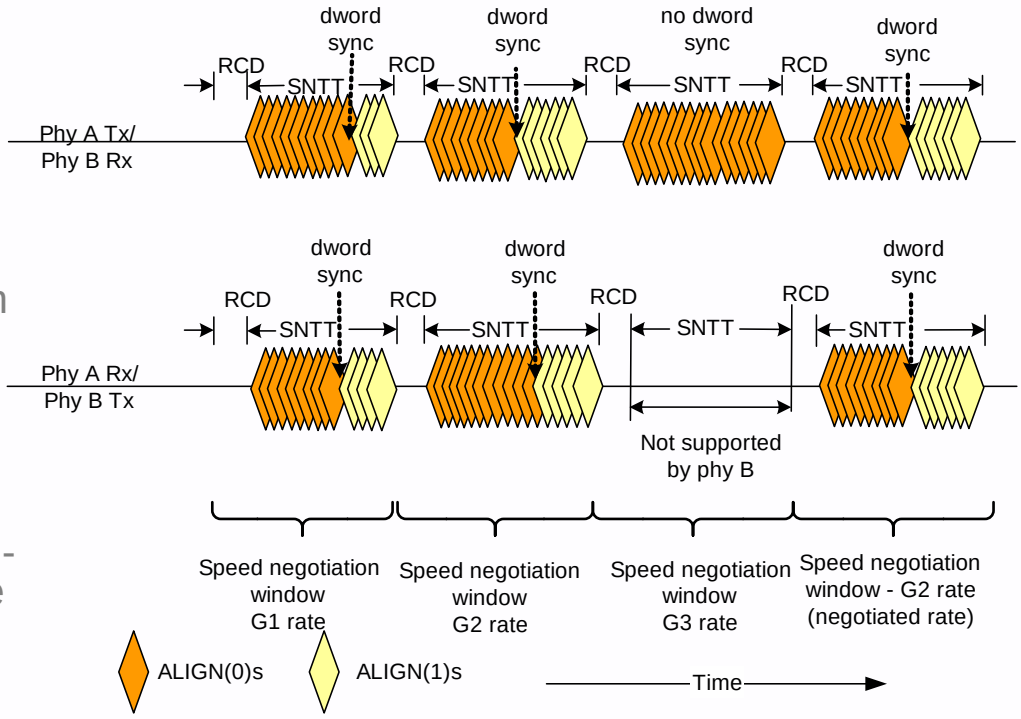
SAS Trainer

- 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
```

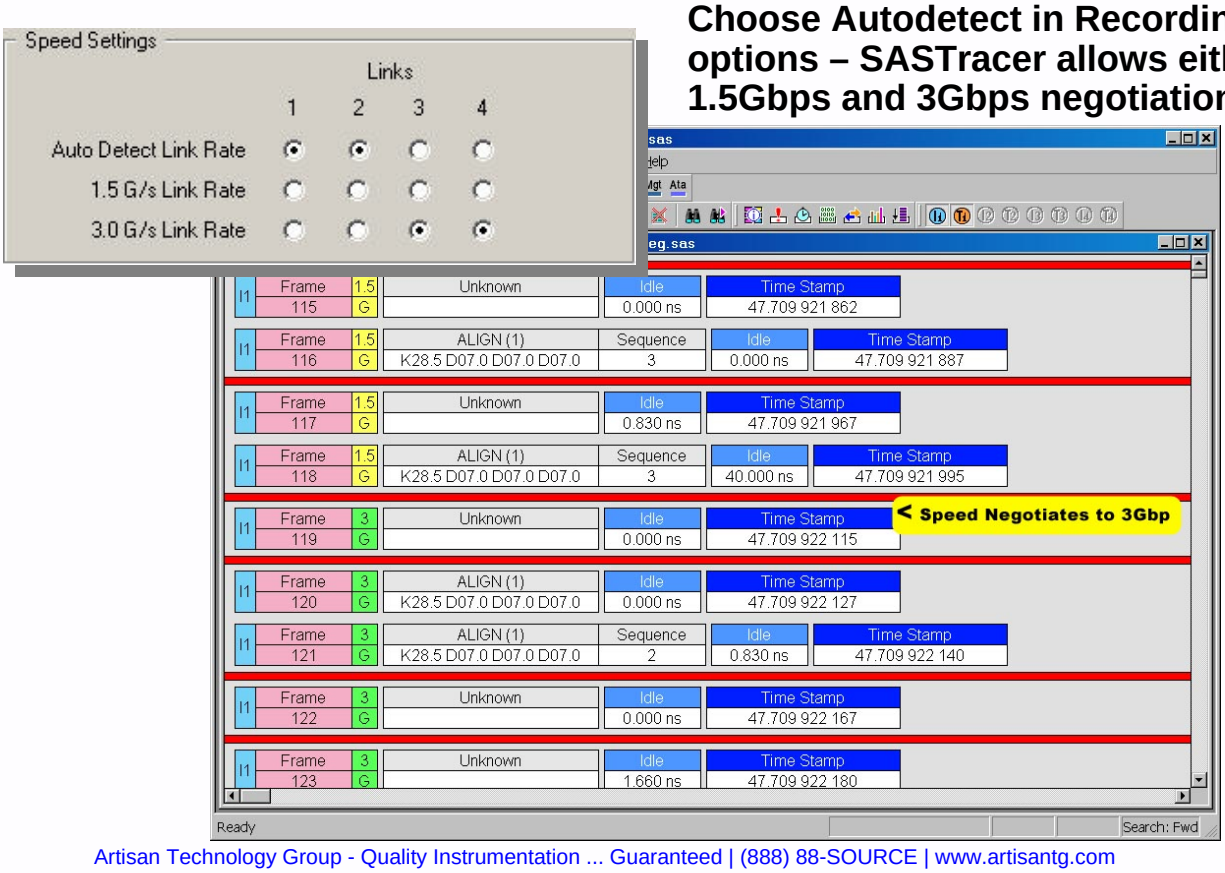
SAS speed negotiation sequence 2

- 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



SASTracer & Speed Negotiation

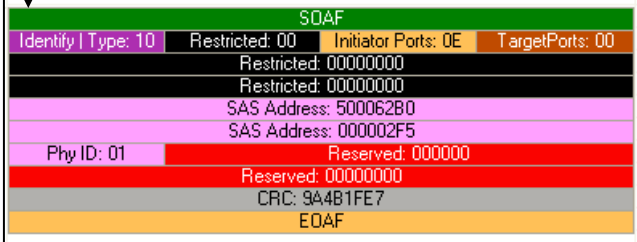
Choose Autodetect in Recording options – SASTracer allows either 1.5Gbps and 3Gbps negotiation



IDENTIFY Sequence

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

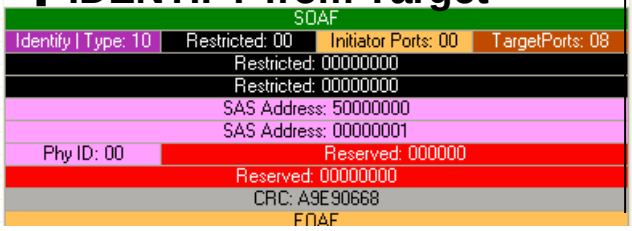
IDENTIFY from Initiator



Targets don't do much with IDENTIFY data from Initiator

Initiator uses IDENTIFY Data from target to perform DISCOVER

IDENTIFY from Target



"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

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						
3								(LSB)
4	Reserved							
5	(MSB)	HASHED SOURCE SAS ADDRESS						
7								(LSB)
8	Reserved							
9	Reserved							
10	Reserved						RETRANSMIT	Reserved
11	Reserved						NUMBER OF FILL BYTES	
12	Reserved							
13		Reserved						
15								
16	(MSB)	TAG						
17								(LSB)
18	(MSB)	TARGET PORT TRANSFER TAG						
19								(LSB)
20	(MSB)	DATA OFFSET						
23								(LSB)
24		INFORMATION UNIT						
m								
	Fill bytes, if needed							
n - 3	(MSB)	CRC						
n								(LSB)

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**

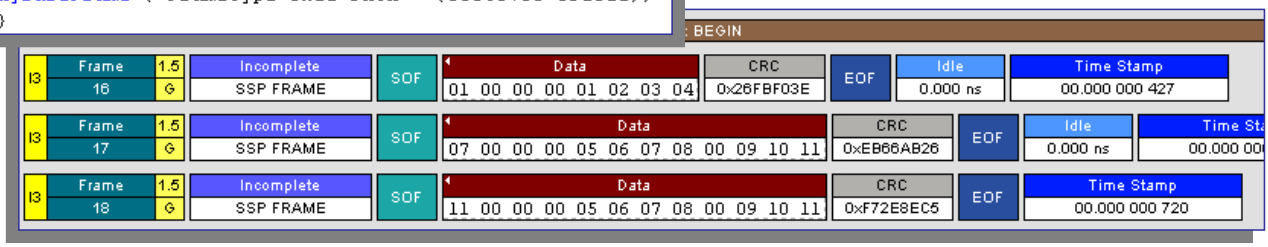
Then Send **DATA**…

WAIT (5000) {

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

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  }
```



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