Agenda

– **Target Applications**
– **M9195B Overview and HW Features**
– **Product Overview and Enhancements**
– **Vector Definition and Timing Model**
– **Programming the M9195B Using STIL**
– **DSR Pattern Editor and Data Converters**
– **Product Structure and Pricing**
– **Advanced Topic: Programming Techniques for RFFE**
– **Advanced Topic: Margin testing for Design Validation Test**
What is a Digital Stimulus/Response Module?

**Customer Problem:** The customer needs a method to digitally control the DUT during functional/parametric testing

**Solution: M9195B DSR**
- Emulate the DUT control interface (e.g. RFFE or SPI)
- Write data values to the DUT
- May need to interact with RF (Triggers)
- Special features for DVT:
  - Parametric testing
  - Shmoo (timing margin)
Customer Problem: The DVT customer needs to be able to vary the edge-placement of input signals to the DUT to ensure it works over the designed-range.

Solution: Easy edge-placement with the M9195B.
Customer Problem: The customer testing low-power devices need flexible digital pattern generation with ability to verify the IC is functional properly at low voltages < 150mV

Solution: M9195B multi-module DSR
• Dynamic digital and Parametric testing
• Need very low leakage current when in response mode (<2nA)
  ✓ M9195B low leakage mode
• High DC accuracy below 150 mV
• May need links to digital design systems for test generation
• Includes wafer-level testing
Hardware Features
High speed digital stimulus/response mode

- 16 bidirectional channels per module, up to 12 modules can be synchronized (192 channels)
- Programmable logic levels: -1.5V to +6.5V,
- Digital stimulus mode output impedance 50ohm nominal
- Up to 250MHz pattern rate with RZ support for clock generation
- Independent signal waveform definitions for arbitrary per bit patterns
- Independent channels: per channel & per clock cycle IO control, per vector timing, per period timing, on-the-fly modification
- 1ns\(^1\) edge placement resolution for per period vector control
- Per channel programmable stimulus/response compensation delay
- Real-time response comparison
- Up to 125M vectors per channel memory
- Multiple Triggers/Markers for use with front panel or backplane

Note \(^1\) 1ns edge placement user settable, minimum value = 1ns
M9195B Digital Stimulus Response

PPMU mode, HV and Open Drain features

PPMU Features

- Each channel can be used for parametric measurements or patterns
- Each channel can be programmed independently
- Shared 16-bit measurement subsystem

4 high voltage IO for flash/polyfuse programming
- Outputs shared with pattern channels 2, 6, 10, 14
- Output voltage range: 0-13.5V
- Maximum data rate: 3MHz

4 open drain output ports for fixture relay control
- Open drain, 10k pullup to +5V
- Current sink: 1A
Per Pin Parametric Measurements

- Test modes: FVMI, FVMV, FIMV, FIMI, MV
- Voltage range: -1.5V to +6.5V
- 5 current ranges: 40mA, 1mA, 100uA, 10uA, 2uA
- Maximum source current is ± 40mA, Selectable remote sense
- Built in windowing: Average 64, Window 50Hz, Window 60Hz
- High throughput: 1ms for first channel. Multiple channels add 250us ea.
M9195B Triggers and Markers

Triggers

- Used to trigger the M9195B from a DUT or other instrumentation
- Triggers can be used with dynamic sites or watchloops and up to 6 triggers per site are supported
- They are edge triggered (not level) and with either rising (positive) or falling (negative) edges
- HW triggers sources include: front panel (GPIO1 and GPIO2), PXI_TRIG0-7, PXI_STAR, and PXIe_DSTARB
- Trigger latency is expected to be up to 4 x EPR (TBR)

Markers

- Allow the M9195B to trigger other instruments
- Up to 3 markers per site are supported
- Markers are placed within a pattern and generally initiated coincident with a particular vector
- Polarity of pulse and pulse width is programmable
- HW marker destinations include: front panel (GPIO1 and GPIO2), PXI_TRIG0-7, PXI_STAR, and PXIe_DSTARC
Multi-Module Mode

• Multi-module sync for systems up to 192 channels (single site only)

• Sync cables available for four and twelve modules
M9195B Block Diagram

Changes from M9195A to M9195B shown in red
Multi-sequencer or Single-sequencer Application Modes

- Two modes:
  - 1 bank: 16 synchronized channels with 1 virtual sequencer
  - 4 bank: 4 independent pattern sequencers with up to 4 synchronized channels
  - Mode is automatically selected based upon the site definition.
  - Modes can be changed on a per test basis
  - Mode type is driven by dynamic site definition
  - Multi-module systems operate in a single-sequencer mode
  - Multi-Sequencer Option: S04
M9195B Digital Stimulus Response

Front panel and DSR connector

GPIO1: used for triggers/markers

Sync: used to synchronize matchloops across multiple modules

Clk In: input for 10 MHz or 100 MHz reference clock

GPIO2: used for triggers/markers

Multi-module Sync: Interconnect multiple modules using this connector
M9195B Accessories

- 0.5m, 1m, and 2m single-site DSR cable
- 1m and 2m multi-site DSR cable
- 1m and 2m SMA breakout cable
- Multi-module sync cables
- Evaluation Board
- DSR adjustment cable
Vector Definition and Timing Model
IEEE 1450 (STIL) Pattern Model

Signal0
1 0 1 1 1 1 1 1 1 0 1

Signal1
1 0 1 Z 1 1 R 0 1

Signal2
H L H H H H H H X X

Clock
T T T T T T T T T T
IEEE 1450 Pattern Model

Signal0

1 0 1 1 1 1 0 1

Signal1

1 0 1 Z 1 1 R 0

Signal2

H L H H H H H X X

Clock

T T T T T T T T

Waveform

Receive Edge

Vector

Includes the period and the times for all possible waveform characters.
Flexible Waveform Bit Cyclization

Specifying timing

- **32 WaveformTables** define the period and times at which stimulus and response events occur within a period.
- **Pattern period**: Supports up to 250MHz clock and data.
- **User defined variable** for flexible timing equations
- **M9195A Stimulus/Response Actions**:
  - **U**: Force the channel to a high
  - **D**: Force the channel to a low
  - **Z**: Stop forcing
  - **P**: Force the channel to the previous state
  - **L**: Compare the channel to a low
  - **H**: Compare the channel to a high
  - **T**: Compare the channel to a Tri-state
  - **X**: Do not compare
- **15 User definable waveform characters** for each signal in the pattern: A-Z, 0-9

```
Timing basic {
  WaveformTable one {
    Period 'clock';
    Waveforms {
      Sclk { 01 { '0ns' D/U; }}
      Sclk { T { '0ns' U; 'clock/2' D; }}
      Sdata { 01 { '0ns' D/U; }}
      Sdata { T { '0ns' U; 'clock/2' D; }}
      Sdata { LH { '0ns' Z; 'clock/2' L/H; }}
      Sdata { Z { '0ns' Z; }}
      VIO { 01 { '0ns' D/U; }}
      Synch { 01 { '0ns' D/U; }}
      } // end waveforms
  } // end waveform table one
```
Timing control within period

2 Drive & 1 Response events available every period

Resolution = 24ps, range up to 0-255 ns
Example: Edge placement using Cyclizer

- Flexible edge placement using Cyclizer
- Seamless vector to vector timing changes using Cyclizer
- Edge Placement Resolution (EPR) 1nS typical

```plaintext
waveform table two {
  period '100ns';
  waveforms {
    sclk 0  '0ns' 'D/U';
    sclk 1  '0ns' 'D/U';
    vio 0  '0ns' 'D/U';
    vio 1  '0ns' 'D/U';
    sdata 0  '10ns' 'L/H';
    sdata 1  '10ns' 'L/H';
    sdata 0  '20ns' 'D/U';
    sdata 1  '20ns' 'D/U';
    sdata 0  '30ns' 'D/U';
    sdata 1  '30ns' 'D/U';
    sdata 0  '40ns' 'D/U';
    sdata 1  '40ns' 'D/U';
    sdata 0  '50ns' 'D/U';
    sdata 1  '50ns' 'D/U';
    synch 0  '0ns' 'D/U';
    synch 1  '0ns' 'D/U';
  }
}
```
Example: Period adjustment using Cyclizer

Adjust Period seamlessly from vector to vector

- Seamless vector to vector period changes using Cyclizer
- Period is declared in Waveform table
- Waveform tables can be associated with pattern on vector by vector basis

This example
- WFT1 = 50nS period
- WFT2 = 100nS period
Channel to Channel timing adjustment

Stimulus Delay feature provides precision ch-ch delays for de-skew and shmoo testing

- Stimulus Delay provides channel to channel timing offset
- Applied to ALL the vectors in the pattern
- Allows channel to channel de-skew and shmoo testing

Stimulus delay example +/-4nS with 50pS steps
Cable Compensation

- Per channel Stimulus Delay is used to synchronize the signals arrival at the DUT.

- Per channel Response Delay Compensation makes the stimulus path to the DUT, the same duration as the “Expected” path internal to the DSR.

- Adjust Response Delay Compensation to align the short internal path with the external path out to the DUT and back.

- Automatic Response Delay Compensation measurement

---

Pattern Generation → Stimulus Delay → Response Delay Compensation → DUT → Response Comparison → Result

### Table: Keysight DSR Module Configuration

<table>
<thead>
<tr>
<th>Channel</th>
<th>Clamp Voltage Minimum (V)</th>
<th>Clamp Voltage Maximum (V)</th>
<th>Clamp Voltage Ppmu Enable</th>
<th>Stimulus Delay (S)</th>
<th>Response Delay Compensation (S)</th>
<th>Line Drive Compensation</th>
<th>Line Receive Compensation</th>
<th>Remote Sense Enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-2.000</td>
<td>7.000</td>
<td>False</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>1</td>
<td>-2.000</td>
<td>7.000</td>
<td>False</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>2</td>
<td>-2.000</td>
<td>7.000</td>
<td>False</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>3</td>
<td>-2.000</td>
<td>7.000</td>
<td>False</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>4</td>
<td>-2.000</td>
<td>7.000</td>
<td>False</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>5</td>
<td>-2.000</td>
<td>7.000</td>
<td>False</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>6</td>
<td>-2.000</td>
<td>7.000</td>
<td>False</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>7</td>
<td>-2.000</td>
<td>7.000</td>
<td>False</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>8</td>
<td>-2.000</td>
<td>7.000</td>
<td>False</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>9</td>
<td>-2.000</td>
<td>7.000</td>
<td>False</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>10</td>
<td>-2.000</td>
<td>7.000</td>
<td>False</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>11</td>
<td>-2.000</td>
<td>7.000</td>
<td>False</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>12</td>
<td>-2.000</td>
<td>7.000</td>
<td>False</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>13</td>
<td>-2.000</td>
<td>7.000</td>
<td>False</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>14</td>
<td>-2.000</td>
<td>7.000</td>
<td>False</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>15</td>
<td>-2.000</td>
<td>7.000</td>
<td>False</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>False</td>
</tr>
</tbody>
</table>

---

Measure And Apply Response Delay Compensation
Programming the M9195B Using STIL
M9195B programming efficiency and ease-of-use

IEEE1450 based Standard Test Interface Language (STIL)

• STIL is a standard method for representing patterns and attributes required for digital IO pattern generation
• STIL has been adopted for use with the M9195B both for its simplicity and ease of use.
• Patterns are constructed from building blocks that are collectively referred to as STIL Components.
• STIL Components are populated by either text file import, Open XML import or IVI driver – or a combination of these three methods
  • Complete STIL files can also be created from digital test design/development tools using the M9192A/M9193A
STIL Components

- **Signal and Site blocks**: declare signals and map to physical channels
- **DCLevel block**: defines signal (logic) levels and response thresholds
- **Waveform tables**: timing information for pattern generation.
- **Pattern Burst and Pattern Exec**: Group into pattern sequences and bind DCLevel blocks to patterns.
- **Patterns**: basic pattern information
- **Other blocks**: SignalGroups and Macros

```plaintext
IEEE 1450: STIL (Standard Test Interface Language)
A standard method to define digital pattern attributes

/** STIL example */
/////////////////////////////////////////////////////////////////////////////
STIL 3.0
Signals {
  DB InOut;
  DE InOut;
  DI InOut;
  DO InOut;
} // end Signals

Site MySite {
  Channels {
    DB 0 0;
    DE 2 2;
    DI 1 1;
    DO 3 3;
  } // end Channels
} // end Site

DCLevels MyLevels {
  DB (VIL "1.8V"; VIL "0V"; VOH "1.8V"; VOL "0V");
  DE (VIL "1.8V"; VIL "0V"; VOH "1.8V"; VOL "0V");
  DI (VIL "1.8V"; VIL "0V"; VOH "1.8V"; VOL "0V");
  DO (VIL "1.8V"; VIL "0V"; VOH "1.8V"; VOL "0V");
}

Timing basic {
  WaveformTable MyWave {
    Period "100ns";
    Waveforms {
      DB ( 01 ( VIL* D1));
      DE ( 01 ( VIL* D2));
      DI ( 01 ( VIL* D3));
      DO ( 01 ( VIL* D4));
    } // end waveforms
  } // end waveform table MyWave
} // end timing basic

PatternBurst MyBurst {
  Patlist ( "MyPatterns" );
}

PatternExec MyPatternExec {
  Timing basic;
  PatternBurst "MyBurst";
  DCLevels MyLevels;
}

Pattern "MyPattern" {
  MyWave;
  Loop 4 (
    V ("0302345678" );
    V ("0302345678" );
    V ("0302345678" );
    V ("0302345678" );
    V ("0302345678" );
    V ("0302345678" );
    V ("0302345678" );
    V ("0302345678" );
  ) // End Loop
} // end Pattern
```
M9195B Flexible programming model

STIL Components entered various methods:

Application Programming Interfaces:

- STIL (Standard Test Interface Language IEEE 1450)
  - Define vectors and related timing
- IVI-COM, IVI-C drivers, LabVIEW driver (later date)
- Easy-to-use, Open XML (.xlsx) programming interface
- Soft-Front Panel for diagnostic and debug
M9195B Flexible programming model

STIL Components may be entered using various methods:

- ASCII text file
- Use Soft Front Panel
- Or Use IVI driver (IVI-COM or IVI-C)

```c
// filename for STIL (.stil) and bulk data loads (.vec)
// Note files travel with C# project
string STILFile = "C:\\RFRE_SetSignalTemplate.stil";
string BulkDataSimple_FileName = "C:\\BulkData_Simple.vec";
string BulkData_RZ_FileName = "C:\\BulkData_RZ.vec";
string BulkData_NAC_FileName = "C:\\BulkData_NAC.vec";

try
{
    // Load STIL file - a convenient method to get initial setups loaded
    M9195A.PatternFile.Load(STILFile, "");
}
```
M9195B Flexible programming model

STIL Components may be entered using various methods:

- Open XML (Excel) file
- Use Soft Front Panel
- Or Use IVI driver (IVI-COM or IVI-C)
Open XML (.xlsx) Templates

Microsoft Excel may also be used for defining pattern attributes

• Simple, easy to use and learn templates using common Open XML compliant commercial products
• Support all the basic DSR functions
• Example templates as a starting point
• Data input error checking

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>ScanLength</th>
</tr>
</thead>
<tbody>
<tr>
<td>Out[1]</td>
<td>out</td>
<td></td>
</tr>
<tr>
<td>Out[2]</td>
<td>out</td>
<td></td>
</tr>
<tr>
<td>Out[3]</td>
<td>out</td>
<td></td>
</tr>
<tr>
<td>Out[4]</td>
<td>out</td>
<td></td>
</tr>
<tr>
<td>Out[5]</td>
<td>out</td>
<td></td>
</tr>
<tr>
<td>Out[6]</td>
<td>out</td>
<td></td>
</tr>
<tr>
<td>In[1]</td>
<td>in</td>
<td></td>
</tr>
<tr>
<td>In[2]</td>
<td>in</td>
<td></td>
</tr>
<tr>
<td>In[3]</td>
<td>in</td>
<td></td>
</tr>
<tr>
<td>In[4]</td>
<td>in</td>
<td></td>
</tr>
<tr>
<td>In[5]</td>
<td>in</td>
<td></td>
</tr>
<tr>
<td>In[6]</td>
<td>in</td>
<td></td>
</tr>
<tr>
<td>In[7]</td>
<td>in</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Character</th>
<th>time</th>
<th>action</th>
<th>time</th>
<th>action</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>1.001-08</td>
<td>D</td>
<td>1 1.001-08</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1.001-07</td>
<td>L</td>
<td>1 1.001-07</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>20ns</td>
<td>D</td>
<td>0/1/T 20ns</td>
<td>D/O/T</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>100ns</td>
<td>M/L</td>
<td>100ns</td>
<td>M/L</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>X</td>
<td>0ns</td>
<td>X</td>
<td>0ns</td>
<td>X</td>
</tr>
<tr>
<td>11</td>
<td>6</td>
<td>0ns</td>
<td>U 100ns</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>0ns</td>
<td>D 100ns</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>X</td>
<td>0ns</td>
<td>X</td>
<td>0ns</td>
<td>X</td>
</tr>
<tr>
<td>14</td>
<td>A</td>
<td>0ns</td>
<td>U 75ns</td>
<td>D</td>
<td>175ns</td>
</tr>
<tr>
<td>15</td>
<td>B</td>
<td>0ns</td>
<td>U 75ns</td>
<td>D</td>
<td>175ns</td>
</tr>
</tbody>
</table>
M9195B Flexible programming model

STIL Components may be entered using various methods:

Direct IVI-COM or IVI-C driver generate and load
M9195B Flexible programming model:

Import Bulk vector data: support for legacy and tool generated patterns

- Bulk data import: a method to load basic pattern data
- Bind pattern data to previously defined Signal and Waveform STIL Components
- Optionally import Signal names in the first row and Waveform Table information on vector by vector basis
- Accepts common "vector" files previously generated by tools or legacy projects
Static Digital IO

- Programmatically write and read static binary values to/from digital pins
- Use API (such as Soft front panel) to control or validate fixture status
- Use API reads and writes for direct access to channels
- Use API to validate cabling without requiring the user to create a pattern
DSR Pattern Editor and Data Converters
M9192A DSR Pattern Editor

- Provides a graphical user interface for M9195B test development and debug
  - Graphical waveform editing
  - Pattern generation from scratch
- Keysight DSR STIL in-converter specially designed for use with M9195B
- Keysight DSR tester bridge works with M9195B SFP
- Drag and drop an M9195B error log file into pattern editor to get a visual of the failures

Based upon the Solstice-TDS platform from TSSI, the leading provider of design-to-test software solutions
M9193A DSR Pattern Editor with Data Converters

- WGL in-converter supports TSSI Waveform Generation Language
- Standard STIL in-converter supports ATPG files from Synopsys TetraMAX, Cadence Encounter, and Mentor Graphics FastScan
- Verilog in-converter for reading vsd/ecvd files
  - Includes a complete set of Cyclization tools for adding timing
- ASCII In-converter allows simple text file input of patterns

Fast test development using Automatic Test Program Generators, with customized output for the M9195B
Product Structure and Pricing
## Product Structure & Price

<table>
<thead>
<tr>
<th>Product Number</th>
<th>Product Description</th>
<th>US List Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>M9195B</td>
<td>PXIe Digital Stimulus/Response with PPMU: 250 MHz, 16ch</td>
<td>$11,500</td>
</tr>
<tr>
<td>Memory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M9195B-M01</td>
<td>Memory, 16 Mvectors/ch</td>
<td>$0</td>
</tr>
<tr>
<td>M9195B-M06</td>
<td>Memory, 64 Mvectors/ch</td>
<td>$500</td>
</tr>
<tr>
<td>M9195B-M12</td>
<td>Memory, 125 Mvectors/ch</td>
<td>$1500</td>
</tr>
<tr>
<td>Multi-Site</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M9195B-S01</td>
<td>Single Site Enabled</td>
<td>$0</td>
</tr>
<tr>
<td>M9195B-S04</td>
<td>Multi-Site Enabled</td>
<td>$1000</td>
</tr>
<tr>
<td>Sample Rate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M9195B-SR1</td>
<td>Max Clock Rate, 125 MHz</td>
<td>$0</td>
</tr>
<tr>
<td>M9195B-SR2</td>
<td>Max Clock Rate, 250 MHz</td>
<td>$2000</td>
</tr>
<tr>
<td>Performance Options</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M9195B-MMS</td>
<td>Multiple Module Synch</td>
<td>$3,000</td>
</tr>
</tbody>
</table>
# Product Structure & Price

## Accessories and Software

<table>
<thead>
<tr>
<th>Product Number</th>
<th>Product Description</th>
<th>US List Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y1245A</td>
<td>Single-site Digital Stimulus/Response Cable: 0.5m</td>
<td>$700</td>
</tr>
<tr>
<td>Y1246A</td>
<td>Single-site Digital Stimulus/Response Cable: 1m</td>
<td>$700</td>
</tr>
<tr>
<td>Y1247A</td>
<td>Single-site Digital Stimulus/Response Cable: 2m</td>
<td>$750</td>
</tr>
<tr>
<td>Y1248A</td>
<td>Multi-site Digital Stimulus/Response Cable: 1m</td>
<td>$1000</td>
</tr>
<tr>
<td>Y1249A</td>
<td>Multi-site Digital Stimulus/Response Cable: 2m</td>
<td>$1200</td>
</tr>
<tr>
<td>Y1250A</td>
<td>Four module synch cable</td>
<td>$500</td>
</tr>
<tr>
<td>Y1251A</td>
<td>Twelve module synch cable</td>
<td>$1500</td>
</tr>
<tr>
<td>Y1252A</td>
<td>Digital Stimulus/Response Calibration Fixture</td>
<td>$1200</td>
</tr>
<tr>
<td>Y1253A</td>
<td>Digital Stimulus/Response Evaluation and Prototyping Board</td>
<td>$350</td>
</tr>
<tr>
<td>Y1254A</td>
<td>Digital Stimulus/Response SMA Breakout Cable: 1m</td>
<td>$1500</td>
</tr>
<tr>
<td>Y1255A</td>
<td>Digital Stimulus/Response SMA Breakout Cable: 2m</td>
<td>$2000</td>
</tr>
<tr>
<td>M9192A</td>
<td>DSR Pattern Editor Software</td>
<td>$0</td>
</tr>
<tr>
<td>M9192A-1TP</td>
<td>DSR Pattern Editor Software, transportable perpetual license</td>
<td>$4,000</td>
</tr>
<tr>
<td>M9192A-1TP-12R</td>
<td>DSR Pattern Editor Software, annual SW updates, 1 year renewal</td>
<td>$600</td>
</tr>
<tr>
<td>M9193A</td>
<td>DSR Pattern Editor with Data Conversion Software</td>
<td>$0</td>
</tr>
<tr>
<td>M9193A-1TP</td>
<td>DSR Pattern Editor with Data Conversion Software, transportable perpetual license</td>
<td>$14,000</td>
</tr>
<tr>
<td>M9193A-1TP-12R</td>
<td>DSR Pattern Editor with Data Conversion Software, annual SW updates, 1 year renewal</td>
<td>$2100</td>
</tr>
</tbody>
</table>

*New with M9195B*
Advanced Topic: Programming Techniques for RFFE
Including on-the-fly Programming
On-the-fly Vector Modification

Operate on existing setups for fast, efficient pattern attribute modification

Use these 4 API commands to:

1) **SetSignal**  modify the pattern for a specific signal over a series of clock periods
2) **SetVector**  completely replace the vector at a given location with the new pattern
3) **SetDCLevel** modify the signal amplitude after the pattern has been downloaded to the DSR
4) **SetWaveformTable**  change the timing of a single bit period. Successive vectors will return to the WaveformTable.

Example API commands

```csharp
Pattern Test_pattern { 
    waveformTable one;
    
    V {'Sc1k+SData+VIO+Synch' =0000; }
    V {'Sc1k+SData+VIO+Synch' =1001; }
    V {'Sc1k+SData+VIO+Synch' =0001; }

    p1: V {'Sc1k+SData+VIO+Synch' =0101; }
    p2: V {'Sc1k+SData+VIO+Synch' =0000; }
    p3: V {'Sc1k+SData+VIO+Synch' =0100; }
    p4: V {'Sc1k+SData+VIO+Synch' =0000; }
    p5: V {'Sc1k+SData+VIO+Synch' =0100; }
    p6: V {'Sc1k+SData+VIO+Synch' =0000; }
} // end Pattern
```
Set Signal for flexible, on the fly programming model

How Set Signal is used to update serial command information

```
M9195A.PatternSites.Item[site].SetSignal("RFFE_Register_IO", "RWO", 0, "SDATA", RFFECmd);
M9195A.PatternSites.Item[site].Initiate();
M9195A.PatternSites.Item[site].WaitForAcquisitionComplete(1000);
```

Pattern RFFE_Register_IO {
    // Generic pattern for RFFE Register Write, Register A4-A0 oddParityBit Data
    // Depending on type of _io either
    // For Register Write: Data[7:0] oddParityBit Bus
    // For Register Read: BusParK Data[7:0] oddPar
    // For Register 0 Write: oddParityBit BusParK
    W WFT1;
    RWO:
    V { VIO+SDATA+SLCK'='100; } // Ensure that Sdata
    V { VIO+SDATA+SLCK'='110; } // ssc start (1) bit
    V { VIO+SDATA+SLCK'='100; } // ssc start (0) bit
    V { VIO+SDATA+SLCK'='10T; } // Command Frame
    V { VIO+SDATA+SLCK'='10T; } // 2
    V { VIO+SDATA+SLCK'='10T; } // 3
    V { VIO+SDATA+SLCK'='10T; } // 4
    V { VIO+SDATA+SLCK'='10T; } // 5
    V { VIO+SDATA+SLCK'='10T; } // 6
    V { VIO+SDATA+SLCK'='10T; } // 7
    V { VIO+SDATA+SLCK'='10T; } // 9
    V { VIO+SDATA+SLCK'='10T; } // 10
    V { VIO+SDATA+SLCK'='10T; } // 11
    V { VIO+SDATA+SLCK'='10T; } // 12
    V { VIO+SDATA+SLCK'='10T; } // 13
    V { VIO+SDATA+SLCK'='10T; } // Data frame
    V { VIO+SDATA+SLCK'='10T; } // 2
    V { VIO+SDATA+SLCK'='10T; } // 3
    V { VIO+SDATA+SLCK'='10T; } // 4
    V { VIO+SDATA+SLCK'='10T; } // 5
    V { VIO+SDATA+SLCK'='10T; } // 6
    V { VIO+SDATA+SLCK'='10T; } // 7
    V { VIO+SDATA+SLCK'='10T; } // 8
    V { VIO+SDATA+SLCK'='10T; } // 9
    V { VIO+SDATA+SLCK'='10T; } // 10
    V { VIO+SDATA+SLCK'='10T; } // 11
    V { VIO+SDATA+SLCK'='100; } // 11
    // Total vector count = 27 (3 + 13 + 11)
```
Set Signal for flexible, on the fly programming model

Constructing RFFE commands for RFIC test

**RFFE_RegWrite command:**  \( SA = 0xFF, \text{Address}= 0x00, \text{Data} = 0x2C \)

- Format Command Frame and calculate parity
- Format Data Frame and calculate Parity
- Assembly string RFFE_Cmd
- Pass string to M9195B memory using Set Signal, initiate and wait for completion
Combined Set-Signal & Bulk Data programming models

Flexible, efficient programming: RFFE bus example

```c
while (ent.ToChar() != 'k') {
    // Demo 5.1 General RFFE programming:
    // Using On-the-fly programming,
    // Using imported bulk data; both NR and NZ clocking
    //
    // M9195B Overview
    // Timing benchmark
    int time_base = 1000 * (double)(ti - te) / (double)frequency;
    sCndWindowReport = String.Format("Completed On-The-Fly IO mode (1$ Coded) Time = \(8\text{ ms}\)\), Time,

    // Register write examples:
    utilities.RFFE_Register(0x1054, "RFFEIO", 0x0, 0x0A, 0x01);
    utilities.RFFE_Register(0x1054, "RFFEIO", 0x0, 0x09, 0x02);
    utilities.RFFE_Register(0x1054, "RFFEIO", 0x0, 0x04, 0x03);
    utilities.RFFE_Register(0x1054, "RFFEIO", 0x0, 0x02, 0x04);
    utilities.RFFE_Register(0x1054, "RFFEIO", 0x0, 0x01, 0x05);
    utilities.RFFE_Register(0x1054, "RFFEIO", 0x0, 0x08, 0x06);
    utilities.RFFE_Register(0x1054, "RFFEIO", 0x0, 0x06, 0x07);

    // Register read examples:
    if (hr.RFFEEndian(0x1054, "RFFEIO", 0x0, 0x04, 0x01, ref m_Serial));
    if (hr.RFFEEndian(0x1054, "RFFEIO", 0x0, 0x02, 0x01, ref m_Serial));
    if (hr.RFFEEndian(0x1054, "RFFEIO", 0x0, 0x01, 0x01, ref m_Serial));

    // register and extended register write examples:
    utilities.RFFE_Register(0x1054, "RFFEIO", 0x0, 0x01);
    utilities.RFFE_Register(0x1054, "RFFEIO", 0x0, 0x02);
    utilities.RFFE_Register(0x1054, "RFFEIO", 0x0, 0x03);
    utilities.RFFE_Register(0x1054, "RFFEIO", 0x0, 0x04);

    M9195B.PatternSites.Inactivate("RFFEIO");

    // Timing benchmark
    int time_base = 1000 * (double)(ti - te) / (double)frequency;
    sCndWindowReport = String.Format("Completed On-The-Fly IO mode (1$ Coded) Time = \(8\text{ ms}\)\), Time,

    // bulk data with NZ Clocking (Uses DTutters write timing table)
    //
    // M9195B Overview
    // Timing benchmark
    int time_base = 1000 * (double)(ti - te) / (double)frequency;
    sCndWindowReport = String.Format("Completed On-The-Fly IO mode (1$ Coded) Time = \(8\text{ ms}\)\), Time,

    // bulk data with NZ Clocking (Uses DTutters write timing table)
    //
    // M9195B Overview
    // Timing benchmark
    int time_base = 1000 * (double)(ti - te) / (double)frequency;
    sCndWindowReport = String.Format("Completed On-The-Fly IO mode (1$ Coded) Time = \(8\text{ ms}\)\), Time,
```

M9195B Overview
RFFE VIO Power Considerations

VIO powers the interface portion of the RFFE I/O it may draw some current.

Table from RFFE spec is shown below. $I_{VIO-IN}^{Average}$ current draw from a slave device $<1.25\,mA$

$VIO = 1.8\,V +/- 150\,mV$ or $1.2\,V +/- 100\,mV$.

Driver with $R_s = 50ohms$ @ $2mA$ Vdrops $100\,mV$

If current draw is too high use PMU mode (low Source resistance)
Advanced Topic: Margin testing for Design Validation Test
Margin testing for Design Validation

**Use Reevaluatevariables for time margin test**

- No need to modify pattern
- Change individual edges on vector-vector basis
- Edge Placement Resolution 1ns minimum
- Can use `reevaluatevariables` API for fast edge timing adjustment

```csharp
// Margin testing using Reevaluate function to change timing
while (ent.Tocover() != "x")
{
    t0 = MyTimer.Elapsedticks;
    for (iCntr = 0; iCntr < 36; iCntr++)
    {
        MS9595A.PatternSites.ActivateFromCache("RF8EID0");
        MS9595A.PatternSites.get_item("RF8EID0").Reevaluatevariables(String.Format("RF8EID0x{0}mS", iCntr));
        // MS9595A.RegWrite(RF8EID0, 0x00, 0x00, 0x00);
        //iCntr += 5;
    }
    t1 = MyTimer.Elapsedticks;
    MS9595A.PatternSites.Inactivate("RF8EID0");
    Console.WriteLine("Margin test time, 36 loop with 1ns steps: {0:0000} ms", t1-t0/(Stopwatch.Frequency)
    Console.WriteLine("Press enter to loop, 'x' to continue to parametric measurements");
    ent = Console.ReadKey();
}
```

*Typical time to modify edge and restart pattern ~3.3ms*
Margin testing for Design Validation

Use Stimulus delay for channel-channel time margin test

- No need to modify pattern
- Change channel to channel timing with high resolution

```c
// Margin testing using stimulus delay for high resolution ch-ch margin
//
double stimDelay = 0;
int n = 0;
while (n < num_channels) {
    // No need to modify pattern
    // Change channel to channel timing with high resolution
    int delay = 500; // Typical time to change delay and restart pattern ~500us
    stimDelay += delay;
    // Code to update channel timing
}
```

**Close-up view:**
- SDATA (green) varying in time vs SCLK (yellow) in 200pS steps
Margin testing for Design Validation

Use ReevaluateDCLevels for logic level margin test

- No need to modify pattern
- Change applied pattern or response threshold logic levels using ReevaluateDCLevels API

Typical time to change level and restart pattern ~2ms
Summary and questions...
Backup slides
Single-bank Application Modes

Single sequencer for pattern generation/acquisition

- Single site cable
  - 50 ohm, high performance ribbonized coax
  - Samtec Edge Rate connectors
  - 1 or 2m options
  - Standard or thumbscrew latching housing

- 1 bank mode used for wider parallel vector applications
  - Single sequencer controls all channels in a multi-module system
Multi-bank Application Mode

4 Independent sequencers for asynchronous test flow

- Four site cable
  - 50 ohm, high performance ribbonized coax
  - Samtec Edge Rate connectors
  - 1 or 2m options
  - Standard or thumbscrew latching housing

- A single module with 4 Independent sequencers
  - Can run same or different tests simultaneously
  - 4 asynchronous clocks
  - Use multiple modules for more sites, modules run independently

4 bank - mode

Channels 0-3
Channels 4-7
Channels 8-11
Channels 12-15
Open Drain 0-3
IEEE 1450 STIL format

Standard Test Interface Language

- STIL standards based terminology and methods to define attributes for generating and acquiring large complex patterns
- Simple text scripting combined with SFP supports quick pattern prototyping and development.
- After initial development use IVI driver to call STIL
Using re-evaluate to adjust waveform timing table

```csharp
driver.PatternSites.Activate("Bank1", "Basic_test_with_HSL", true, edgePlacement);
for (iCntr = 10; iCntr < 21; iCntr++) {
    iPeriodNs = iCntr*10;
    driver.PatternSites.get_Item("Bank1").ReevaluateVariables("MyPeriod=" + iPeriodNs.ToString() + "ns");
    driver.PatternSites.get_Item("Bank1").Initiate();
    driver.PatternSites.get_Item("Bank1").WaitForAcquisitionComplete(1000);
    Console.Write(". ");
    Thread.Sleep(10);
}
driver.PatternSites.InactivateAndDisable("Bank1");
```

Using re-evaluate to adjust waveform DC Levels table

```csharp
driver.PatternSites.get_Item("Bank1").Initiate(); //Do first init to setup the default DC valuses
for (iCntr = 0; iCntr < 11; iCntr++) {
    VIH_OffsetV = -0.5 + ((double)iCntr)/10; //step VIH from Vnom +/-500mV in 100mV steps
    driver.PatternSites.get_Item("Bank1").ReevaluateDcLevels("VIH_OffsetV=" + VIH_OffsetV.ToString());
    driver.PatternSites.get_Item("Bank1").Initiate();
    driver.PatternSites.get_Item("Bank1").WaitForAcquisitionComplete(1000);
    Console.Write(". ");
    Thread.Sleep(400);
}
```
Combined Set-Signal & Bulk Data programming models

Flexible, efficient programming: SPI bus example

- 1st 5 commands using Set Signal method
- Last 2 commands from imported RZ and NRZ bulk data
Summary RFIC test applications

Summary of M9195A DSR implementation and advantages

- RFIC parametric tests using PMU feature:
  - Input leakage using FVMI – 2uA range
  - Clamp measurements using FIMV
- RFIC SPI or MIPI RFFE commands
  - On-the-fly programming for fast, flexible test flow and DUT specific commands
  - Bulk data for legacy vector file re-use
    - Waveform Table for RZ and NRZ format support
  - PMU for VIO power – up to 40mA
- SPI or MIPI RFFE margin testing
  - Vector to vector timing control for margin test
  - Channel to Channel timing control for margin test
  - Logic level margin testing
## Edge Placement Optimizer

This tool displays the actual times used by the DSR given an edge placement resolution and target values.

**Edge Placement Resolution:** 1 ns

### Target Values

<table>
<thead>
<tr>
<th>Weight</th>
<th>Type</th>
<th>Waveform Table</th>
<th>Target (ns)</th>
<th>Actual</th>
<th>Absolute Error</th>
<th>Relative Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stim. Delay</td>
<td>Resp. Delay Comp.</td>
<td>WFT1</td>
<td>38.46</td>
<td>40.000 ns</td>
<td>1.540 ns</td>
<td>4.0%</td>
</tr>
<tr>
<td>Stim. Delay</td>
<td>Resp. Delay Comp.</td>
<td>WFT1</td>
<td>10</td>
<td>10.000 ns</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Stim. Delay</td>
<td>Resp. Delay Comp.</td>
<td>WFT1</td>
<td>20</td>
<td>20.000 ns</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Stim. Delay</td>
<td>Resp. Delay Comp.</td>
<td>WFT1</td>
<td>19.23</td>
<td>19.000 ns</td>
<td>0.230 ns</td>
<td>1.2%</td>
</tr>
<tr>
<td>Stim. Delay</td>
<td>Resp. Delay Comp.</td>
<td>WFT2</td>
<td>50.9</td>
<td>50.900 ns</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Stim. Delay</td>
<td>Resp. Delay Comp.</td>
<td>WFT2</td>
<td>2000</td>
<td>2.000 us</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Stim. Delay</td>
<td>Resp. Delay Comp.</td>
<td>WFT2</td>
<td>1000</td>
<td>1000.000 ns</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Stim. Delay</td>
<td>Resp. Delay Comp.</td>
<td>WFT2</td>
<td>1955.3</td>
<td>1.952 us</td>
<td>3.300 ns</td>
<td>0.2%</td>
</tr>
<tr>
<td>Stim. Delay</td>
<td>Resp. Delay Comp.</td>
<td>WFT3</td>
<td>10000</td>
<td>9.984 us</td>
<td>16.000 ns</td>
<td>0.2%</td>
</tr>
<tr>
<td>Stim. Delay</td>
<td>Resp. Delay Comp.</td>
<td>WFT3</td>
<td>5000</td>
<td>4.992 us</td>
<td>8.000 ns</td>
<td>0.2%</td>
</tr>
<tr>
<td>Stim. Delay</td>
<td>Resp. Delay Comp.</td>
<td>WFT3</td>
<td>28.1</td>
<td>28.100 ns</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Stim. Delay</td>
<td>Resp. Delay Comp.</td>
<td>WFT3</td>
<td>28.2</td>
<td>28.200 ns</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Stim. Delay</td>
<td>Resp. Delay Comp.</td>
<td>WFT3</td>
<td>28.3</td>
<td>28.300 ns</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Stim. Delay</td>
<td>Resp. Delay Comp.</td>
<td>WFT3</td>
<td>28.4</td>
<td>28.400 ns</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Stim. Delay</td>
<td>Resp. Delay Comp.</td>
<td>WFT3</td>
<td>28.5</td>
<td>28.500 ns</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Stim. Delay</td>
<td>Resp. Delay Comp.</td>
<td>WFT3</td>
<td>28.6</td>
<td>28.600 ns</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Stim. Delay</td>
<td>Resp. Delay Comp.</td>
<td>WFT3</td>
<td>28.7</td>
<td>28.700 ns</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Stim. Delay</td>
<td>Resp. Delay Comp.</td>
<td>WFT3</td>
<td>28.8</td>
<td>28.800 ns</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Images:**
- [image][518x71]
- [image][60x84]
Multi-sequencer or Single-sequencer Application Modes

2G memory used for patterns, responses, and pattern sequencing
Multi-sequencer or Single-sequencer Application Modes

2G memory used for patterns, responses, and pattern sequencing
Multi-sequencer or Single-sequencer Application Modes

- Two modes:
  - 1 bank: 16 synchronized channels with 1 virtual sequencer
  - 4 bank: 4 independent pattern sequencers with up to 4 synchronized channels
- Mode is automatically selected based upon the site definition.
- Modes can be changed on a per test basis
- Mode type is driven by dynamic site definition
- Multi-module systems operate in a single-sequencer mode
- Multi-Sequencer Option: S04
**Single Sequencer Definition Example**

- All 4 sequencers are synchronized into a single virtual sequencer because of the site assignment which crosses the internal boundaries.

- Dynamic Digital (Ch 0-4, 6, 12-13)
  - Synchronous, deterministic patterns taken from memory and controlled by the sequencer.

- Unused (Ch 5)

- Ch 6

- PMU site (Ch 7-8)

- Static Digital (9-11)
  - Defined in the API. Not stored in memory.

- Ch 12-13

- PMU site (Ch 14-15)
  - Able to perform FVMI, FIMV tests
**Multi-sequencer Definition Example**

<table>
<thead>
<tr>
<th>4 sequencer operation</th>
<th>Seq 1</th>
<th>Seq 2</th>
<th>Seq 3</th>
<th>Seq 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>4</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>6</td>
<td>10</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>7</td>
<td>11</td>
<td>15</td>
</tr>
</tbody>
</table>

- All 4 sequencers are able to operate independently

- Dynamic Digital (Ch 0-3)
- Dynamic Digital (4-7)
- Dynamic Digital (8-11)
- Dynamic Digital (Ch 12-15)
Invalid Multi-sequencer Definition Example

- Dynamic Digital (0-7)
- Dynamic Digital (Ch 8-15)

Can only synchronize all 4 sequencers
Multi-sequencer Definition Example – 2 Sequencers

- 2 sequencers are able to operate independently. Site assignment didn’t cross pin boundaries

- Dynamic Digital (0-3)

- PMU site (4-7)

- Dynamic Digital (8-11)

- PMU Site (Ch 12-15)