Using the Pattern Generator

Online Help

Agilent Technologies

Notices

Trademarks

Microsoft®, MS-DOS®, Windows®, Windows 2000®, and Windows XP® are U.S. registered trademarks of Microsoft Corporation.

Adobe®, Acrobat®, and the Acrobat Logo® are trademarks of Adobe Systems Incorporated.

Manual Part Number

Version 03.70.0000

Edition

December 1, 2007

Available in electronic format only

Agilent Technologies, Inc.
1900 Garden of the Gods Road
Colorado Springs, CO 80907 USA

Warranty

The material contained in this document is provided "as is," and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

Restricted Rights Legend

If software is for use in the performance of a U.S. Government prime contract or subcontract, Software is delivered and licensed as "Commercial computer software" as defined in DFAR 252.227-7014 (June 1995), or as a "commercial item" as defined in FAR 2.101(a) or as "Restricted computer software" as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Agilent Technologies' standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

Safety Notices

CAUTION

A **CAUTION** notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a **CAUTION** notice until the indicated conditions are fully understood and met.

WARNING

A WARNING notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in personal injury or death. Do not proceed beyond a WARNING notice until the indicated conditions are fully understood and met.

Pattern Generator—At a Glance

The Agilent Technologies 16720A pattern generator is used by digital design teams to emulate digital signals in circuits under development. The pattern generator can take the place of missing devices, or can act as a stimulus to functionally test prototypes.

Contents

1
Pattern Generator Overview

The Agilent Technologies 16720A pattern generator is a tool that generates digital signals. It is used in applications that require an external source to simulate digital circuitry or generate digital signals for functionally testing prototype hardware.

Combined with the analog and digital measurement capabilities of the logic analysis system, you have a tightly integrated solution to your digital stimulus and response measurement needs.

Steps in Using the Pattern Generator

The exact output pattern, clock type and speed, and number of required signals depends on your specific application. How you configure the pattern generator and what kind of signal generation sequence you create will vary. However, from a procedural standpoint, the steps are the same each time to set up, create a sequence, and start the pattern generator.

1. Select the probing (see page 12) that is compatible with your device under test.
2. Set the Output Mode (see page 36) and the Clock Source (see page 37) parameters.
3. Connect the probes (see page 32) to your circuit and define buses and signals (see page 39) in the pattern generator user interface.
4. Create a sequence of test vectors (see page 53) to generate the desired output signals.
5. Run (see page 99) the pattern generator and measure the device under test for the desired results.

Re-using Pattern Generator Programs

After you set up a pattern generator configuration, you may want to store it away so you can use it again. Perhaps you want to create a set of test routines or circuit simulators. There are three ways to handle re-usable configurations.

- You can reload previously saved (see page 101) pattern generator configurations. Macros and loops are restored from ALA format configuration files as originally defined. The compiled vector sequence (without loops and macros) is restored from XML format configuration files.

- You can import previously exported data from comma-separated value (CSV) format files (see page 105). CSV format files contain a *compiled* version of the data; in other words, macros are expanded and loops are shown as a repeated sequence of vectors.
- You can import data from PattGen Binary (PGB) format files (see page 116).

NOTE

You can convert 16522A pattern generator ASCII format files (see page 132) to comma-separated value (CSV) and XML format files containing vector data and setup information, respectively.

NOTE

When 16700-series logic analysis system configuration files contain pattern generator data that you want to reuse, you can export the data to PattGen Binary (PGB) format files (from within the 16700-series logic analysis system) and convert them to to the 16900-series PGB format (see page 140).

See Also

- Key Characteristics (see page 144)

2
Connecting Output Signals to the Device Under Test

Selecting the Correct Probe Pod

The following equivalent circuit information is provided to help you select the appropriate clock and data pods for your application.

- Data Pod Descriptions (see page 12)
  - 10461A TTL Data Pod (see page 13)
  - 10462A 3-State TTL/CMOS Data Pod (see page 14)
  - 10464A ECL Data Pod (terminated) (see page 14)
  - 10465A ECL Data Pod (unterminated) (see page 15)
  - 10466A 3-State TTL/3.3 V Data Pod (see page 16)
  - 10469A PECL Data Pod (see page 16)
  - 10471A LVPECL Data Pod (see page 17)
  - 10473A 3-State 2.5 V Data Pod (see page 18)
  - 10476A 3-State 1.8 V Data Pod (see page 19)
  - 10483A 3-State 3.3 V Data Pod (see page 20)
  - E8141A LVDS Data Pod (see page 20)
- Clock Pod Descriptions (see page 21)
  - 10460A TTL Clock Pod (see page 22)
  - 10463A ECL Clock Pod (see page 23)
  - 10468A PECL Clock Pod (see page 24)
  - 10470A LVPECL Clock Pod (see page 25)
  - 10472A 2.5 V Clock Pod (see page 26)
  - 10475A 1.8 V Clock Pod (see page 27)
  - 10477A 3.3 V Clock Pod (see page 28)
  - E8140A LVDS Clock Pod (see page 29)
- Pod Dimensions (see page 30)
- Data Cable Characteristics without a Data Pod (see page 30)
- Clock Cable Characteristics without a Clock Pod (see page 31)

See Also

- Connecting the Probe Pods (see page 32)

Data Pod Descriptions

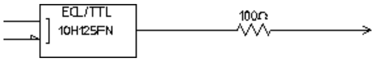
- 10461A TTL Data Pod (see page 13)
- 10462A 3-State TTL/CMOS Data Pod (see page 14)
- 10464A ECL Data Pod (terminated) (see page 14)
- 10465A ECL Data Pod (unterminated) (see page 15)

- 10466A 3-State TTL/3.3 V Data Pod (see page 16)
- 10469A PECL Data Pod (see page 16)
- 10471A LVPECL Data Pod (see page 17)
- 10473A 3-State 2.5 V Data Pod (see page 18)
- 10476A 3-State 1.8 V Data Pod (see page 19)
- 10483A 3-State 3.3 V Data Pod (see page 20)
- E8141A LVDS Data Pod (see page 20)

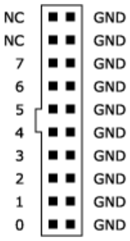
See Also

- Clock Pod Descriptions (see page 21)
- Connecting the Probe Pods (see page 32)

10461A TTL Data Pod

Output type:	10H125 with 100 ohm in series 
Maximum clock:	200 MHz
Skew:	Typical less than 2 ns; worst case 4 ns (see About Data Pod Skew Characteristics (see page 21))
Recommended lead set:	10474A (see page 33)

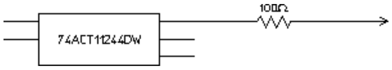
Pinout



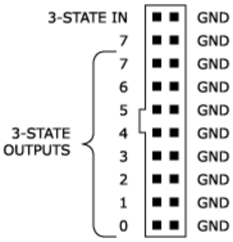
See Also

- Pod Dimensions (see page 30)
- Clock Pod Descriptions (see page 21)
- Connecting the Probe Pods (see page 32)

10462A 3-State TTL/CMOS Data Pod

Output type:	74ACT11244 with 100 ohm in series; 10H125 on non-3-state channel 7 (see Using the "3-STATE IN" Enable Input (see page 21)) 
3-State enable:	Negative true, 100K ohm to GND, enabled on no connect
Maximum clock:	100 MHz
Skew:	Typical less than 4 ns; worst case 12 ns (see About Data Pod Skew Characteristics (see page 21))
Recommended lead set:	10474A (see page 33)

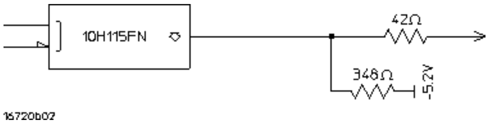
Pinout



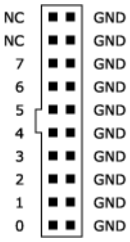
See Also

- Pod Dimensions (see page 30)
- Clock Pod Descriptions (see page 21)
- Connecting the Probe Pods (see page 32)

10464A ECL Data Pod (terminated)

Output type:	10H115 with 348 ohm pulldown, 42 ohm in series 
Maximum clock:	300 MHz
Skew:	Typical less than 1 ns; worst case 2 ns (see About Data Pod Skew Characteristics (see page 21))
Recommended lead set:	10474A (see page 33)

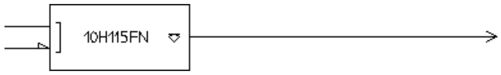
Pinout



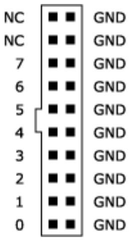
See Also

- Pod Dimensions (see page 30)
- Clock Pod Descriptions (see page 21)
- Connecting the Probe Pods (see page 32)

10465A ECL Data Pod (unterminated)

Output type:	10H115 (no termination) 
Maximum clock:	300 MHz
Skew:	Typical less than 1 ns; worst case 2 ns (see About Data Pod Skew Characteristics (see page 21))
Recommended lead set:	10347A (see page 33)

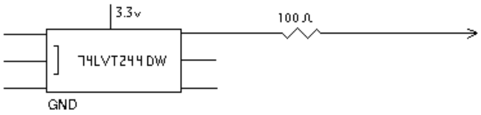
Pinout



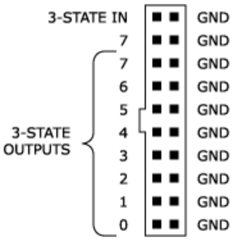
See Also

- Pod Dimensions (see page 30)
- Clock Pod Descriptions (see page 21)
- Connecting the Probe Pods (see page 32)

10466A 3-State TTL/3.3 V Data Pod

Output type:	74LVT244 with 100 ohm in series; 10H125 on non-3-state channel 7 (see Using the "3-STATE IN" Enable Input (see page 21)) 
3-State enable:	negative true, 100K ohm to GND, enabled on no connect
Maximum clock:	200 MHz
Skew:	Typical less than 3 ns; worst case 7 ns (see About Data Pod Skew Characteristics (see page 21))
Recommended lead set:	10474A (see page 33)

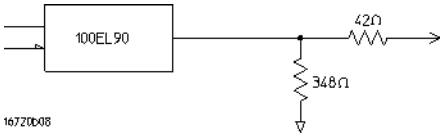
Pinout



See Also

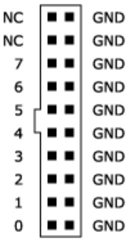
- Pod Dimensions (see page 30)
- Clock Pod Descriptions (see page 21)
- Connecting the Probe Pods (see page 32)

10469A PECL Data Pod

Output type:	100EL09 (5V) with 348 ohm pulldown to ground and 42 ohm in series 
Maximum clock:	300 MHz

Skew:	Typical less than 500 ps; worst case 1 ns (see About Data Pod Skew Characteristics (see page 21))
Recommended lead set:	10498A (see page 33)

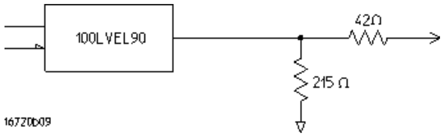
Pinout



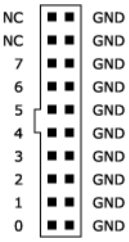
See Also

- Pod Dimensions (see page 30)
- Clock Pod Descriptions (see page 21)
- Connecting the Probe Pods (see page 32)

10471A LVPECL Data Pod

Output type:	100LVEL90 (3.3V) with 215 ohm pulldown to ground and 42 ohm in series 
Maximum clock:	300 MHz
Skew:	Typical less than 500 ps; worst case 1 ns (see About Data Pod Skew Characteristics (see page 21))
Recommended lead set:	10498A (see page 33)

Pinout

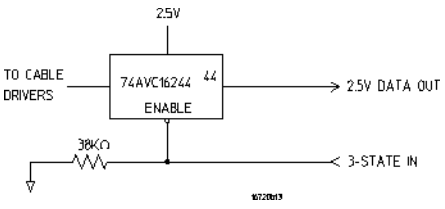


See Also

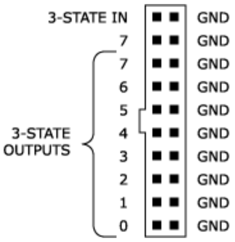
- Pod Dimensions (see page 30)

- Clock Pod Descriptions (see page 21)
- Connecting the Probe Pods (see page 32)

10473A 3-State 2.5 V Data Pod

Output type:	74AVC16244  2.5V TO CABLE DRIVERS 74AVC16244 44 ENABLE 2.5V DATA OUT 38KΩ 3-STATE IN
3-State enable:	negative true, 38K ohm to GND, enabled on no connect (see Using the "3-STATE IN" Enable Input (see page 21))
Maximum clock:	300 MHz
Skew:	Typical less than 1 ns; worst case 2 ns (see About Data Pod Skew Characteristics (see page 21))
Recommended lead set:	10498A (see page 33)

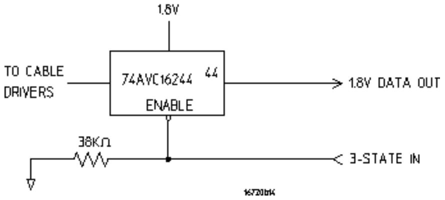
Pinout



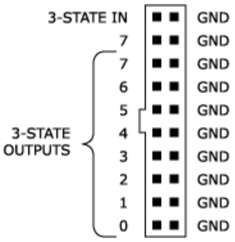
See Also

- Pod Dimensions (see page 30)
- Clock Pod Descriptions (see page 21)
- Connecting the Probe Pods (see page 32)

10476A 3-State 1.8 V Data Pod

Output type:	74AVC16244  1.8V TO CABLE DRIVERS 74AVC16244 44 ENABLE 1.8V DATA OUT 38KΩ 3-STATE IN
3-State enable:	negative true, 38K ohm to GND, enabled on no connect (see Using the "3-STATE IN" Enable Input (see page 21))
Maximum clock:	300 MHz
Skew:	Typical less than 1.5 ns; worst case 2.5 ns (see About Data Pod Skew Characteristics (see page 21))
Recommended lead set:	10498A (see page 33)

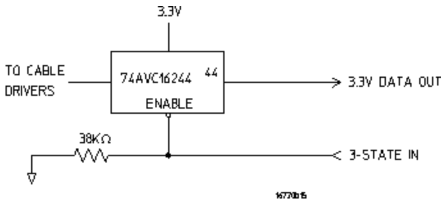
Pinout



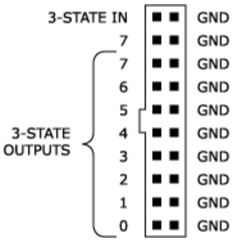
See Also

- Pod Dimensions (see page 30)
- Clock Pod Descriptions (see page 21)
- Connecting the Probe Pods (see page 32)

10483A 3-State 3.3 V Data Pod

Output type:	74AVC16244  3.3V TO CABLE DRIVERS 74AVC16244 44 ENABLE 3.3V DATA OUT 38KΩ 3-STATE IN 16720b15
3-State enable:	negative true, 38K ohm to GND, enable on no connect (see Using the "3-STATE IN" Enable Input (see page 21))
Maximum clock:	300 MHz
Skew:	Typical less than 1 ns; worst case 2 ns (see About Data Pod Skew Characteristics (see page 21))
Recommended lead set:	10498A (see page 33)

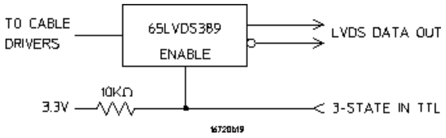
Pinout



See Also

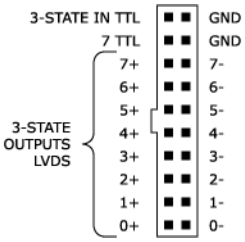
- Pod Dimensions (see page 30)
- Clock Pod Descriptions (see page 21)
- Connecting the Probe Pods (see page 32)

E8141A LVDS Data Pod

Output type:	65LVDS389 (LVDS data lines); 10H125 (TTL non-3-state channel 7, see Using the "3-STATE IN" Enable Input (see page 21))  TO CABLE DRIVERS 65LVDS389 ENABLE LVDS DATA OUT 10KΩ 3.3V 3-STATE IN TTL 16720b19
3-State enable:	positive true TTL; no connect=enabled

Maximum clock:	300 MHz
Skew:	Typical less than 1 ns; worst case 2 ns (see About Data Pod Skew Characteristics (see page 21))
Recommended lead set:	E8142A (see page 33)

Pinout



See Also

- Pod Dimensions (see page 30)
- Clock Pod Descriptions (see page 21)
- Connecting the Probe Pods (see page 32)

About Data Pod Skew Characteristics

Typical skew measurements made at the pod connector with approximately 10 pF/50K ohm load to GND; worst case skew numbers are a calculation of worst case conditions through circuits. Both numbers apply to any channel within a single or multiple module system.

Using the "3-STATE IN" Enable Input

The pattern generator always holds the last set of state vectors when it is stopped.

When a logic "1" level is applied to the "3-STATE IN" input, the 3-stateable outputs are placed into a "high impedance" state. When a logic "0" level is applied to the "3-STATE IN" input, or when the input is not connected, the 3-stateable outputs are active, and the levels of the last vector are output.

Data pods with 3-state control provide a parallel channel 7 output that is non-3-stateable. By looping this output back into the "3-STATE IN" input, the pattern generator's channel 7 output can be used as a 3-state control signal.

Clock Pod Descriptions

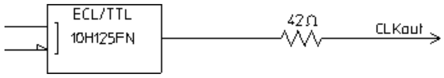
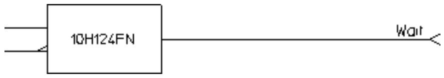
- 10460A TTL Clock Pod (see page 22)
- 10463A ECL Clock Pod (see page 23)
- 10468A PECL Clock Pod (see page 24)

- 10470A LVPECL Clock Pod (see page 25)
- 10472A 2.5 V Clock Pod (see page 26)
- 10475A 1.8 V Clock Pod (see page 27)
- 10477A 3.3 V Clock Pod (see page 28)
- E8140A LVDS Clock Pod (see page 29)

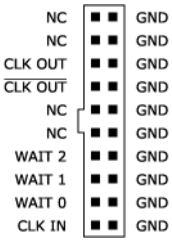
See Also

- Data Pod Descriptions (see page 12)
- Connecting the Probe Pods (see page 32)

10460A TTL Clock Pod

Clock output type:	10H125 with 42 ohm series; true & inverted 
Clock output rate:	100 MHz maximum
Clock out delay:	Approximately 8 ns total in 14 steps
Clock input type:	TTL - 10H124 
Clock input rate:	DC to 100 MHz
Pattern input type:	TTL - 10H124 (no connect is logic 1)
Clock-in to clock-out:	Approximately 30 ns
Pattern-in to recognition:	Approximately 15 ns + 1 clk period
Recommended lead set:	10474A (see page 33)

Pinout

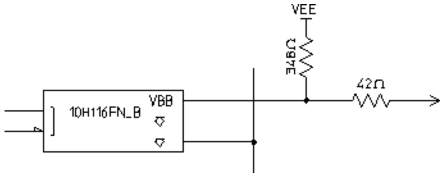
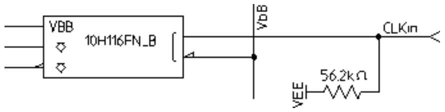


See Also

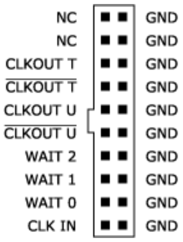
- Pod Dimensions (see page 30)
- Data Pod Descriptions (see page 12)

- Connecting the Probe Pods (see page 32)

10463A ECL Clock Pod

Clock output type:	10H116 differential unterminated; differential with 348 ohm to -5.2v and 42 ohm series 
Clock output rate:	300 MHz maximum
Clock out delay:	Approximately 8 ns total in 14 steps
Clock input type:	ECL - 10H116 with 50K ohm to -5.2v 
Clock input rate:	DC to 300 MHz
Pattern input type:	ECL - 10H116 with 50K ohm (no connect is logic 0)
Clock-in to clock-out:	Approximately 30 ns
Pattern-in to recognition:	Approximately 15 ns + 1 clk period
Recommended lead set:	10474A (see page 33)

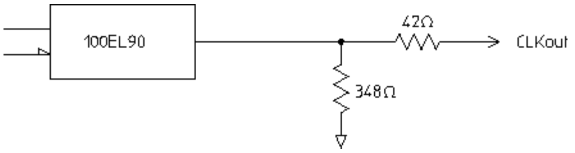
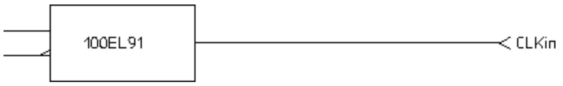
Pinout



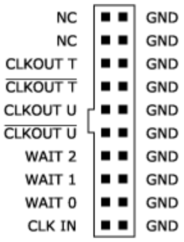
See Also

- Pod Dimensions (see page 30)
- Data Pod Descriptions (see page 12)
- Connecting the Probe Pods (see page 32)

10468A PECL Clock Pod

Clock output type:	100EL90 (5V) with 348 ohm pulldown to ground and 42 ohm in series 
Clock output rate:	300 MHz maximum
Clock out delay:	Approximately 8 ns total in 14 steps
Clock input type:	100EL91 PECL (5V), no termination 
Clock input rate:	DC to 300 MHz
Pattern input type:	100EL91 PECL (5V), no termination (no connect is logic 0)
Clock-in to clock-out:	Approximately 30 ns
Pattern-in to recognition:	Approximately 15 ns + 1 clk period
Recommended lead set:	10498A (see page 33)

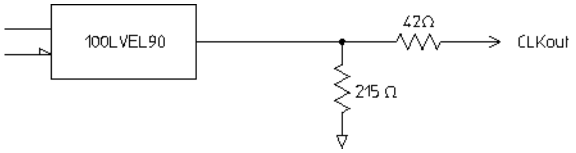
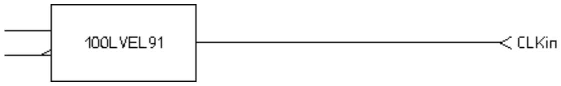
Pinout



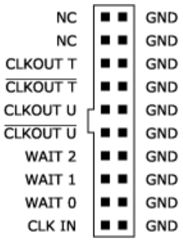
See Also

- Pod Dimensions (see page 30)
- Data Pod Descriptions (see page 12)
- Connecting the Probe Pods (see page 32)

10470A LVPECL Clock Pod

Clock output type:	100LVEL90 (3.3V) with 215 ohm pulldown to ground and 42 ohm in series 
Clock output rate:	300 MHz maximum
Clock out delay:	Approximately 8 ns total in 14 steps
Clock input type:	100LVEL91 PECL (3.3V), no termination 
Clock input rate:	DC to 300 MHz
Pattern input type:	100LVEL91 PECL (3.3V), no termination (no connect is logic 0)
Clock-in to clock-out:	Approximately 30 ns
Pattern-in to recognition:	Approximately 15 ns + 1 clk period
Recommended lead set:	10498A (see page 33)

Pinout

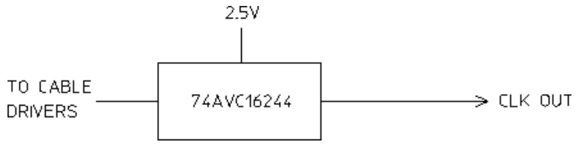
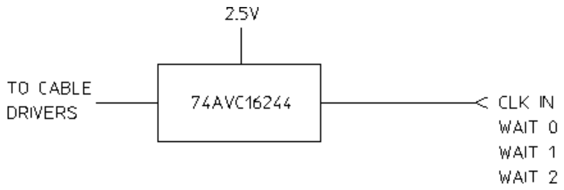


NC	GND
NC	GND
CLKOUT T	GND
$\overline{\text{CLKOUT T}}$	GND
CLKOUT U	GND
$\overline{\text{CLKOUT U}}$	GND
WAIT 2	GND
WAIT 1	GND
WAIT 0	GND
CLK IN	GND

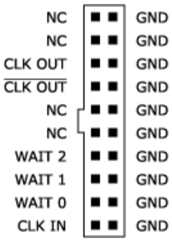
See Also

- Pod Dimensions (see page 30)
- Data Pod Descriptions (see page 12)
- Connecting the Probe Pods (see page 32)

10472A 2.5 V Clock Pod

Clock output type:	74AVC16244 
Clock output rate:	200 MHz maximum
Clock out delay:	Approximately 8 ns total in 14 steps
Clock input type:	74AVC16244 (3.6 V maximum)  2.5V TO CABLE DRIVERS 74AVC16244 CLK IN WAIT 0 WAIT 1 WAIT 2
Clock input rate:	DC to 200 MHz
Pattern input type:	74AVC16244 (3.6 V maximum; no connect is logic 0)
Clock-in to clock-out:	Approximately 30 ns
Pattern-in to recognition:	Approximately 15 ns + 1 clk period
Recommended lead set:	10498A (see page 33)

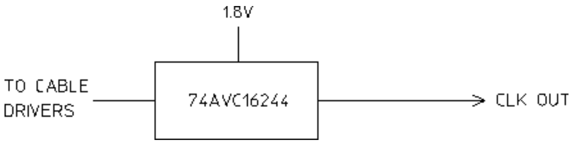
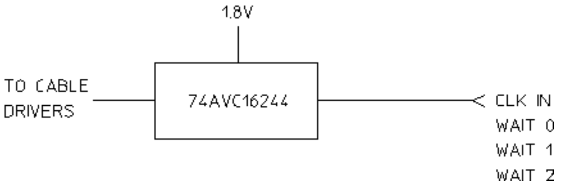
Pinout



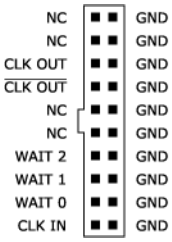
See Also

- Pod Dimensions (see page 30)
- Data Pod Descriptions (see page 12)
- Connecting the Probe Pods (see page 32)

10475A 1.8 V Clock Pod

Clock output type:	74AVC16244 
Clock output rate:	200 MHz maximum
Clock out delay:	Approximately 8 ns total in 14 steps
Clock input type:	74AVC16244 (3.6 V maximum)  1.8V TO CABLE DRIVERS 74AVC16244 CLK IN WAIT 0 WAIT 1 WAIT 2
Clock input rate:	DC to 200 MHz
Pattern input type:	74AVC16244 (3.6 V maximum; no connect is logic 0)
Clock-in to clock-out:	Approximately 30 ns
Pattern-in to recognition:	Approximately 15 ns + 1 clk period
Recommended lead set:	10498A (see page 33)

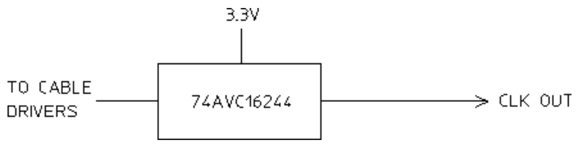
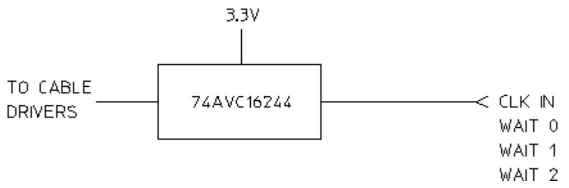
Pinout



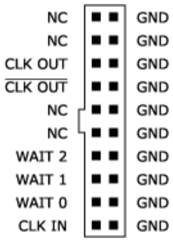
See Also

- Pod Dimensions (see page 30)
- Data Pod Descriptions (see page 12)
- Connecting the Probe Pods (see page 32)

10477A 3.3 V Clock Pod

Clock output type:	74AVC16244 
Clock output rate:	200 MHz maximum
Clock out delay:	Approximately 8 ns total in 14 steps
Clock input type:	74AVC16244 (3.6 V maximum)  3.3V TO CABLE DRIVERS 74AVC16244 CLK IN WAIT 0 WAIT 1 WAIT 2
Clock input rate:	DC to 200 MHz
Pattern input type:	74AVC16244 (3.6 V maximum; no connect is logic 0)
Clock-in to clock-out:	Approximately 30 ns
Pattern-in to recognition:	Approximately 15 ns + 1 clk period
Recommended lead set:	10498A (see page 33)

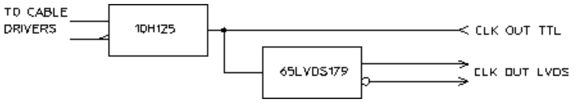
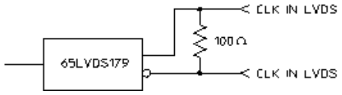
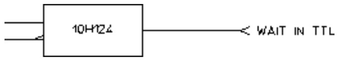
Pinout



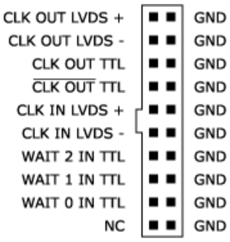
See Also

- Pod Dimensions (see page 30)
- Data Pod Descriptions (see page 12)
- Connecting the Probe Pods (see page 32)

E8140A LVDS Clock Pod

Clock output type:	65LVDS179 (LVDS) and 10H125 (TTL)  TO CABLE DRIVERS 10H125 65LVDS179 CLK OUT TTL CLK OUT LVDS
Clock output rate:	200 MHz maximum (LVDS and TTL)
Clock out delay:	Approximately 8 ns total in 14 steps
Clock input type:	65LVDS179 (LVDS with 100 ohm) 
Clock input rate:	DC to 150 MHz (LVDS)
Pattern input type:	10H124 (TTL) (no connect is logic 1) 
Clock-in to clock-out:	Approximately 30 ns
Pattern-in to recognition:	Approximately 15 ns + 1 clk period
Recommended lead set:	E8142A (see page 33)

Pinout

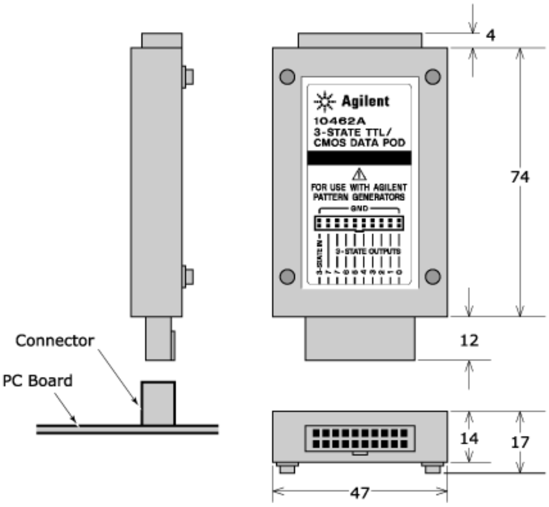


CLK OUT LVDS +	GND
CLK OUT LVDS -	GND
CLK OUT TTL	GND
$\overline{\text{CLK OUT}}$ TTL	GND
CLK IN LVDS +	GND
CLK IN LVDS -	GND
WAIT 2 IN TTL	GND
WAIT 1 IN TTL	GND
WAIT 0 IN TTL	GND
NC	GND

See Also

- Pod Dimensions (see page 30)
- Data Pod Descriptions (see page 12)
- Connecting the Probe Pods (see page 32)

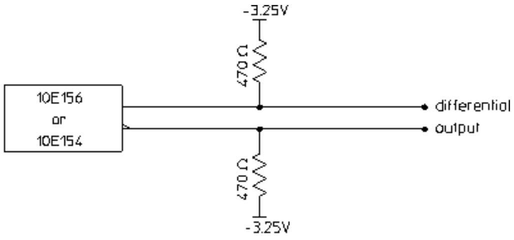
Pod Dimensions (in millimeters)

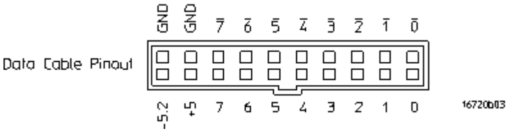


See Also

- Data Pod Descriptions (see page 12)
- Clock Pod Descriptions (see page 21)
- Connecting the Probe Pods (see page 32)

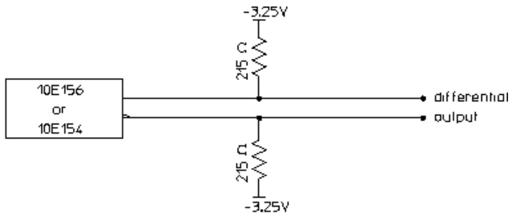
Data Cable Characteristics without a Data Pod

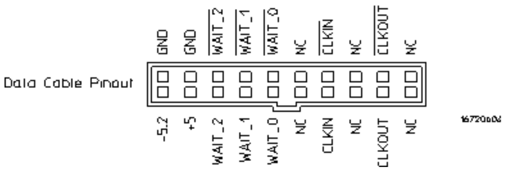




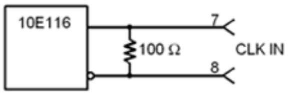
The data cables without a data pod provide an ECL-terminated (470 ohm; to -3.25V) differential signal. These signals are usable when received by a differential receiver, preferably with a 100 ohm termination across the lines. These signals should not be used single-ended due to the slow fall time and shifted voltage threshold; they are not ECL compatible.

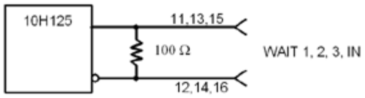
Clock Cable Characteristics without a Clock Pod

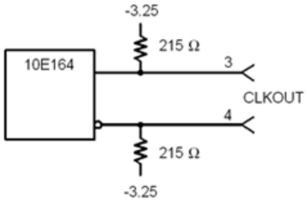




The clock out signals (CLKOUT and not-CLKOUT) without a clock pod provide an ECL-terminated (215 ohm to -3.25V) differential signal. These signals are usable when received by a differential receiver, preferably with a 100 ohm termination across the lines. These signals should not be used single-ended due to the slow fall time and shifted voltage threshold; they are not ECL compatible.







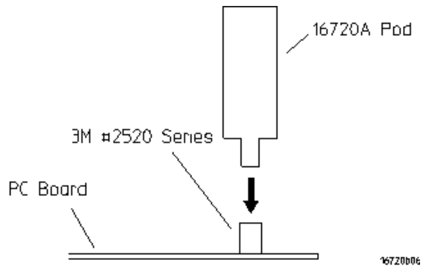
Connecting the Probe Pods

NOTE

Clock and Data pods are required for a proper signal interface. There are different types (see page 12) available, and they should match your device under test circuit characteristics. In addition, depending on the Output Mode (see page 36) selected, some pods may not be available for use.

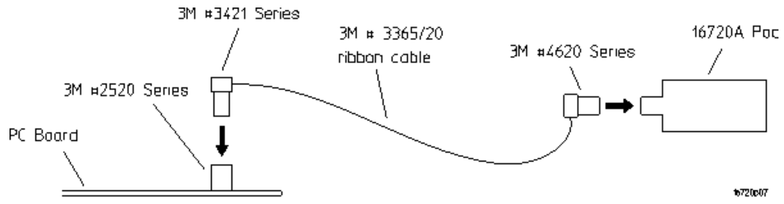
- Direct Pod-to-Board Connection (see page 32)
- Jumper Cable-to-Pod Connection (see page 32)
- Probe Lead Set to Board Pin Connection (see page 33)

Direct Pod-to-Board Connection



Plug the pod directly into the ©3M 2520-series, or similar alternative connector on the PC board.

Jumper Cable-to-Pod Connection

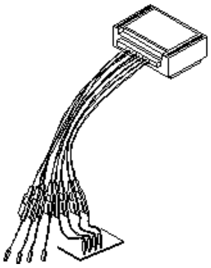


Use this method when you have clearance problems on the PC board. Construct a flat-ribbon cable and connect as shown above.

NOTE

You can obtain equivalent connectors from sources other than 3M.

Probe Lead Set to Board Pin Connection



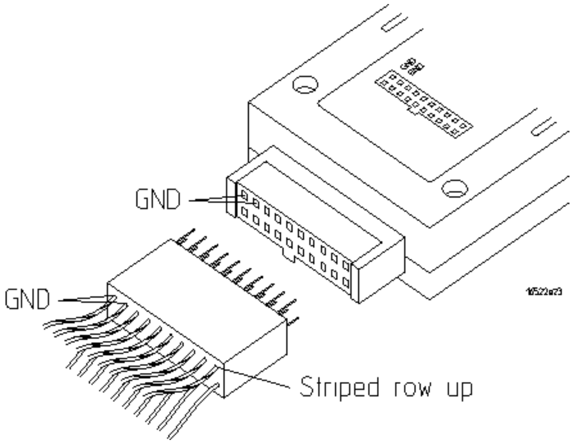
The following probe lead assemblies are available for connecting to PC board pins.

- E8142A 8-channel LVDS probe lead set (6-inch lead wire length).
- 10498A 8-channel probe lead set (6-inch lead wire length).
- 10474A 8-channel probe lead set (12-inch lead wire length).
- 10347A 8-channel probe lead set, 50-ohm coaxial for unterminated signals.

The probe tips of both lead sets plug directly into any 0.1-inch grid with 0.026- to 0.033-inch diameter round pins or 0.025-inch square pins. These probe tips work with the 5090-4356 surface mount grabbers and the 5959-0288 through-hole grabbers.

NOTE

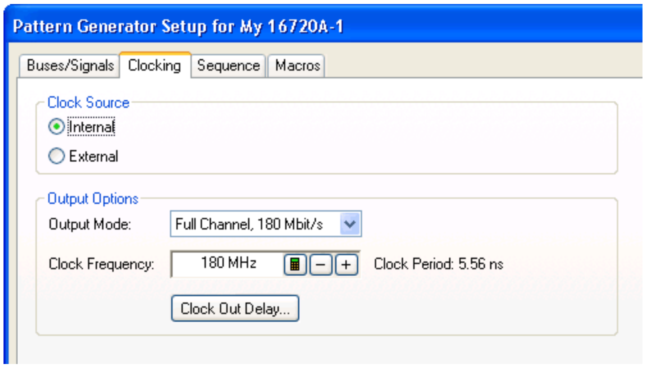
The LVDS Data Pod must be connected to the leads in such a way that the striped row of cables faces up.


GND
GND
Striped row up
16522e23

3
Configuring the Pattern Generator

- Selecting the Output Mode (Channels/Speed Trade-off) (see page 36)
- Selecting the Clock Source (see page 37)

Selecting the Output Mode (Channels/Speed Trade-off)



Pattern Generator Setup for My 16720A-1

Clock Source

● Internal

○ External

Output Options

Output Mode: Full Channel, 180 Mbit/s

Clock Frequency: 180 MHz Clock Period: 5.56 ns

The output mode determines the channel width, available pods, and the frequency range for both the internal and external clock. The choice you make may be determined by trade-offs between clock speed and channel width.

Because the output mode affects clock frequency ranges, available pods, and channel width, keep your mode selection in mind when designing the circuit's hardware interface and when mapping probe connections between the test circuit and the labels of the pattern generator.

This table shows the difference between the Full-Channel 180 Mbits/s mode and the Half-Channel 300 Mbits/s mode.

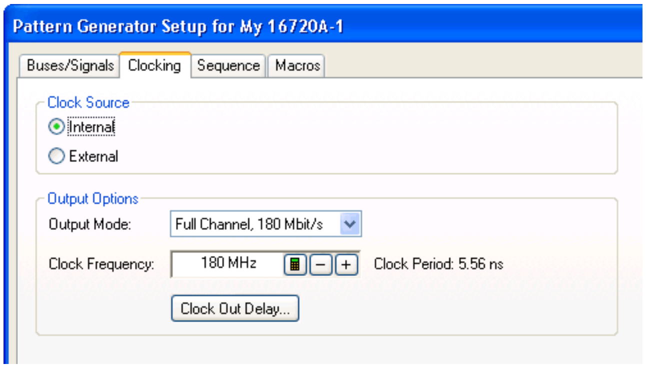
	Full Channel 180 Mbits/s	**Half Channel 300 Mbits/s**
Pods available	Pods 1, 2, 3, 4, 5, 6	Pods 1, 3, 5
Maximum channels	48; 8 per pod	24; 8 per pod
Maximum external clock frequency	180 MHz	300 MHz
Maximum internal clock frequency	180 MHz	300 MHz
Minimum external clock frequency	DC	DC
Minimum internal clock frequency	1 MHz	1 MHz
Maximum vectors	8,388,608	16,777,216

Selecting the Clock Source

The Clock Source field toggles between internal and external. The internal clock source is supplied by the pattern generator and controls the frequency used to output the vectors to the system under test. The external clock is provided by the user or the system under test, and is input to the pattern generator through the CLK IN probe of a clock pod.

An advantage of using an external clock is that you synchronize the vector output of the pattern generator to the system under test. No matter which clock source is used, vectors are always output on the rising edge of the clock.

Internal Clock Source



Clock Source
● Internal
○ External

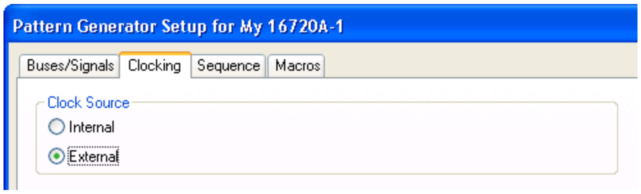
Output Options
Output Mode: Full Channel, 180 Mbit/s
Clock Frequency: 180 MHz
Clock Period: 5.56 ns

Use an internal clock source when you want to have control over the frequency of the output vectors and it is not important for the output vectors to be synchronized to the system under test.

You select clock frequencies in increments of 1 MHz. If you use the keypad to select a value between the step intervals, the value is rounded to the nearest interval.

	Full Channel 180 Mbits/s	Half Channel 300 Mbits/s
Maximum internal clock frequency	180 MHz	300 MHz
Minimum internal clock frequency	1 MHz	1 MHz

External Clock Source



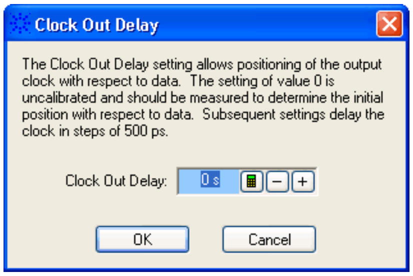
Clock Source
○ Internal
● External

Use an external clock source when you want to synchronize the frequency of the output vectors to the system under test. With this mode selected, you do not have direct control over the frequency of the output vectors. Output vector frequency will be the same as the external clock.

	Full Channel 180 Mbits/s	Half Channel 300 Mbits/s
Maximum external clock frequency	180 MHz	300 MHz
Minimum external clock frequency	DC	DC

CAUTION If the external clock is faster than the maximum period, the Agilent 16720A pattern generator will produce erroneous output vectors.

Clock Out Delay



The Clock Out Delay setting allows positioning of the output clock with respect to data. The setting of value 0 is uncalibrated and should be measured to determine the initial position with respect to data. Subsequent settings delay the clock in steps of 500 ps.

Clock Out Delay: 0 s

The clock out delay setting lets you position the output clock with respect to the data. The zero setting is uncalibrated and should be measured to determine the initial position with respect to the data. You can delay the clock in 500 ps increments.

4
Assigning Bus/Signal Names to Pattern Generator Probes

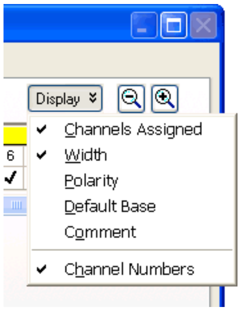
Before you can set up pattern generator vector sequences, you must define bus/signal names for the channels on which you want to output data.

The Buses/Signals tab is accessed through the menu bar's **Setup>(Pattern Generator Module)>Bus/Signal...** command. The Buses/Signals tab is used to map (assign) bus and signal names to the pod and channel connections of the probes. Through the **Display** button, you can select what bus/signal information is displayed.

The following tasks are performed in the Buses/Signals tab:

- To add a new bus or signal (see page 41)
- To delete a bus or signal (see page 42)
- To rename a bus or signal (see page 43)
- To assign channels in the default bit order (see page 44)
- To assign channels, selecting the bit order (see page 45)
- To reorder bits by editing the Channels Assigned string (see page 46)
- To set the default number base (see page 48)
- To set polarity (see page 49)
- To add user comments (see page 50)
- To add a folder (see page 51)
- To sort bus/signal names (see page 52)

Through the **Display** button, you can select what bus/signal setup information is displayed (channels assigned, width, polarity, default base, comment, or channel numbers).


Display
6
✓ Channels Assigned
✓ Width
Polarity
Default Base
Comment
✓ Channel Numbers

To add a new bus or signal

Before you can assign pattern generator probe channels to bus/signal names, you must first create the bus/signal names.

1 From the menu bar, select **Setup>(Pattern Generator Module)>Bus/Signal...**.

2 In the Buses/Signals tab of the Pattern Generator Setup dialog, select **Add Bus/Signal...**.

3 In the Add Bus/Signal dialog, enter the name of the bus/signal.

4 Click **OK**.

See Also

- To delete a bus or signal (see page 42)
- To rename a bus or signal (see page 43)
- To assign channels in the default bit order (see page 44)
- To assign channels, selecting the bit order (see page 45)

To delete a bus or signal

The delete bus or signal feature allows you to remove buses and signals individually or all at once.

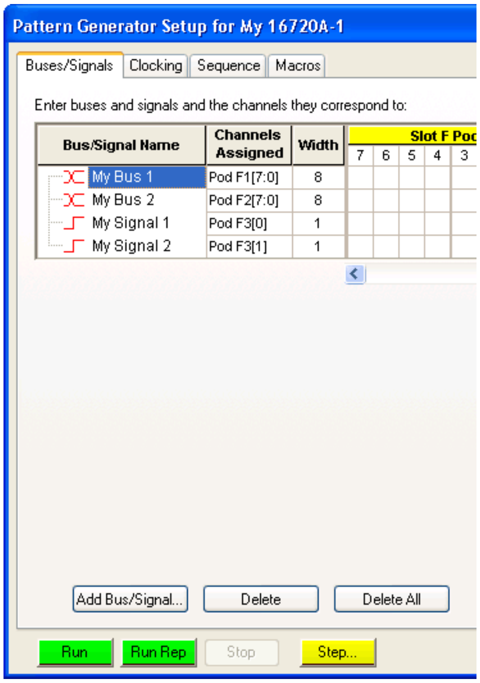
- To delete an individual bus or signal (see page 42)
- To delete all buses and signals (see page 42)

To delete an individual bus or signal

1 From the menu bar, select **Setup>(Pattern Generator Module)>Bus/Signal...**.

2 In the Buses/Signals tab of the Pattern Generator Setup dialog, highlight the bus or signal you want to delete.

3 Click **Delete.**



Enter buses and signals and the channels they correspond to:

Bus/Signal Name	Channels Assigned	Width	Slot F Poc				
			7	6	5	4	3
My Bus 1	Pod F1[7:0]	8					
My Bus 2	Pod F2[7:0]	8					
My Signal 1	Pod F3[0]	1					
My Signal 2	Pod F3[1]	1					

To delete all buses and signals

1 From the menu bar, select **Setup>(Pattern Generator Module)>Bus/Signal...**.

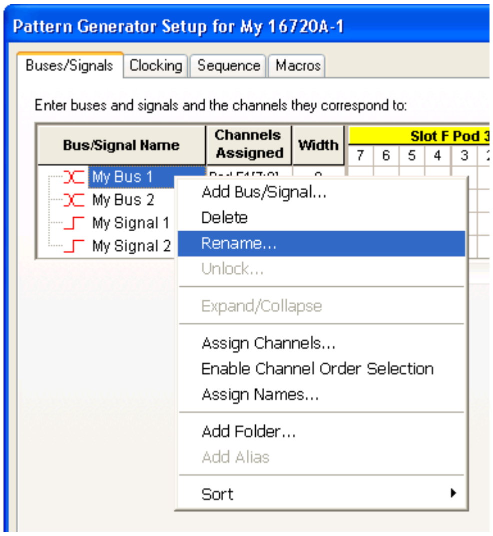
2 In the Buses/Signals tab of the Pattern Generator Setup dialog, click **Delete All**.

To rename a bus or signal

The rename bus/signal feature allows you to change bus and signal names. All pod channel assignments for the renamed bus/signal remain unchanged.

1 From the menu bar, select **Setup>(Pattern Generator Module)>Bus/Signal...**.

2 In the Buses/Signals tab of the Pattern Generator Setup dialog, right-click the bus or signal name and choose **Rename...**.



Enter buses and signals and the channels they correspond to:

3 Enter the new bus or signal name.

4 Click **OK**.

See Also

- To add a new bus or signal (see page 41)
- To delete a bus or signal (see page 42)

To assign channels in the default bit order

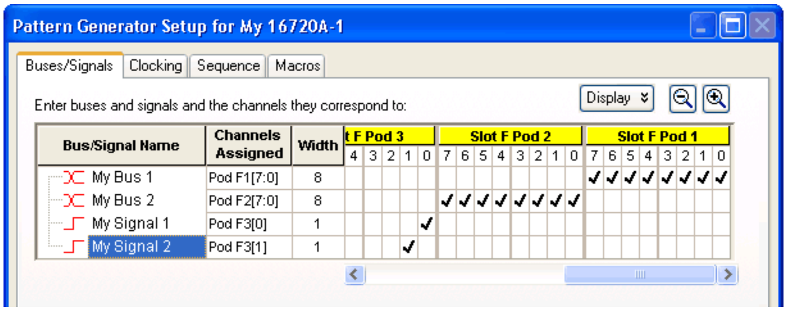
1 From the menu bar, select **Setup>(Pattern Generator Module)>Bus/Signal...**.

2 In the Buses/Signals tab of the Pattern Generator Setup dialog, select squares in the grid to assign channels to bus and signal names.

For each pattern generator output signal, you should have a black check mark mapping particular pod channels to bus/signal names. Unassigned channels are inactive.

Example: In the picture below:

- channels 0-7 on pod F1 are mapped to My Bus 1
- channels 0-7 on pod F2 are mapped to My Bus 2
- channel 0 on pod F3 is mapped to My Signal 1
- channel 1 on pod F3 is mapped to My Signal 2



Pattern Generator Setup for My 16720A-1

Enter buses and signals and the channels they correspond to:

Bus/Signal Name	Channels Assigned	Width	t F Pod 3					Slot F Pod 2								Slot F Pod 1							
			4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
My Bus 1	Pod F1[7:0]	8														✓	✓	✓	✓	✓	✓	✓	✓
My Bus 2	Pod F2[7:0]	8						✓	✓	✓	✓	✓	✓	✓	✓								
My Signal 1	Pod F3[0]	1					✓																
My Signal 2	Pod F3[1]	1				✓																	

NOTE You cannot assign any single output channel to more than one bus/signal.

See Also

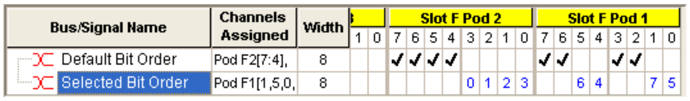
- To reorder bits by editing the Channels Assigned string (see page 46)
- To assign channels, selecting the bit order (see page 45)

To assign channels, selecting the bit order

In cases where pattern generator connectors have been designed into a device under test, and buses on those connectors are not wired to consecutive pattern generator channels, you can assign channels to a bus name in a selected bit order.

1. From the menu bar, select **Setup>(Pattern Generator Module)>Bus/Signal...**.
2. In the Buses/Signals tab, right-click the bus name, and choose **Enable Channel Order Selection**.
3. Start selecting squares in the grid to assign channels from the low order bit of the bus to the high order bit.

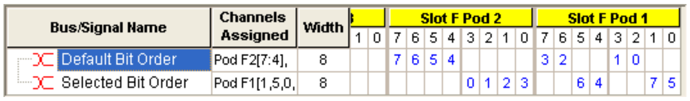
   The bit numbers are displayed as you select squares.



Bus/Signal Name	Channels Assigned	Width			Slot F Pod 2								Slot F Pod 1							
			1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Default Bit Order	Pod F2[7:4],	8			✓	✓	✓	✓					✓	✓			✓	✓		
Selected Bit Order	Pod F1[1,5,0,	8							0	1	2	3			6	4			7	5

To reset the default bit order

The default bit order of assigned channels has higher bits on the left and lower on the right (in the Bus/Signal Setup dialog).



Bus/Signal Name	Channels Assigned	Width			Slot F Pod 2								Slot F Pod 1							
			1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Default Bit Order	Pod F2[7:4],	8			7	6	5	4					3	2			1	0		
Selected Bit Order	Pod F1[1,5,0,	8							0	1	2	3			6	4			7	5

To reset to the default bit order:

- Right-click the bus name, and uncheck **Enable Channel Order Selection**.

See Also

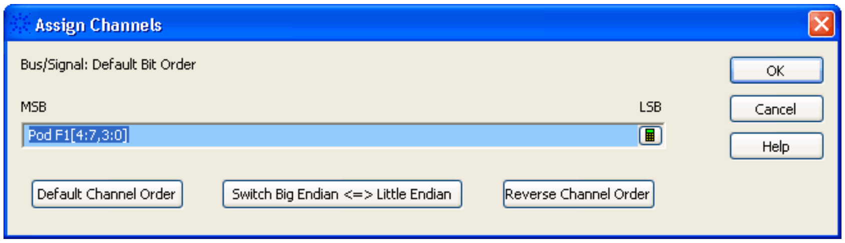
- To reorder bits by editing the Channels Assigned string (see page 46)
- To assign channels in the default bit order (see page 44)

To reorder bits by editing the Channels Assigned string

You can change the order of the bits in a bus name (assigned in either the default order (see page 44) or a selected order (see page 45)) by editing the **Channels Assigned** text string.

1 In the Buses/Signals tab of the Pattern Generator Setup dialog, click the **Channels Assigned** to the bus name.

2 In the Assign Channels dialog, enter the appropriate order of bits in the bus.



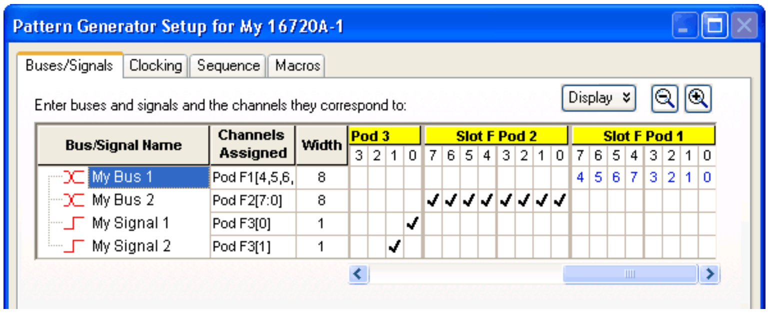
Bus/Signal: Default Bit Order

Pod F1[4:7,3:0]

Example	Description
Pod F1[0]	Signal consisting of the first channel in the first pod.
Pod F1[7:0]	Bus consisting of all eight channels in Pod F1 in default order.
Pod F1[3:0], Pod F2[7:4]	Bus with four channels from first pod followed by four channels from second pod.
Pod F1[7:0], Pod F2[7:0]	Big endian, little endian switch on a 16-bit bus.
Pod F1[0,1,2,3]	Bus with bits in reverse order.

3 Click **OK**.

Channel numbers are displayed for reordered bits in the Buses/Signals tab.



Pattern Generator Setup for My 16720A-1

Enter buses and signals and the channels they correspond to:

Bus/Signal Name	Channels Assigned	Width	Pod 3				Slot F Pod 2								Slot F Pod 1							
			3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
My Bus 1	Pod F1[4,5,6,	8													4	5	6	7	3	2	1	0
My Bus 2	Pod F2[7:0]	8					✓	✓	✓	✓	✓	✓	✓	✓								
My Signal 1	Pod F3[0]	1				✓																
My Signal 2	Pod F3[1]	1			✓																	

To reset the default bit order

Either:

- Right-click the bus name, and uncheck **Enable Channel Order Selection**.

Or:

1 Click the **Channels Assigned** to the bus name.

2 In the Assign Channels dialog, click **Default Channel Order**.

3 Click **OK**.

See Also

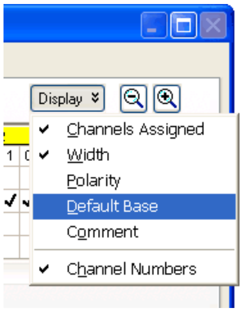
- To assign channels (see page 44)

To set the default number base

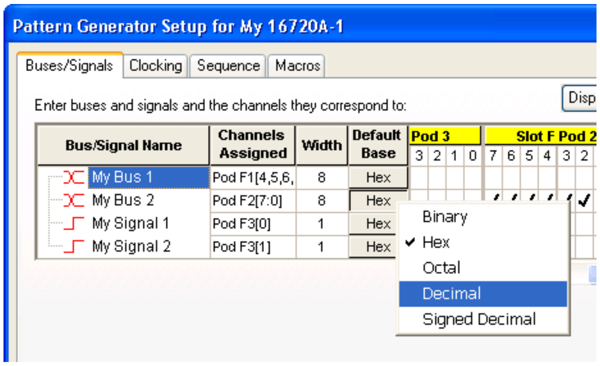
You can set the default number base for a bus when you create the bus. The default base is used in the Sequence and Macros tabs when inserting columns. Default base only affects new bus/signal columns; if you change default base for an existing bus or signal, you will not see the base change in the Sequence and Macros tabs.

1 From the main menu, select **Setup>(Pattern Generator Module)>Bus/Signals...**.

2 In the Buses/Signals tab of the Pattern Generator Setup dialog, click **Display** and choose **Default Base**.



3 To change the default base for a bus or signal, click the default base value and select the new default base.



Pattern Generator Setup for My 16720A-1

Enter buses and signals and the channels they correspond to:

Bus/Signal Name	Channels Assigned	Width	Default Base
My Bus 1	Pod F1[4,5,6,	8	Hex
My Bus 2	Pod F2[7:0]	8	Hex
My Signal 1	Pod F3[0]	1	Hex
My Signal 2	Pod F3[1]	1	Hex

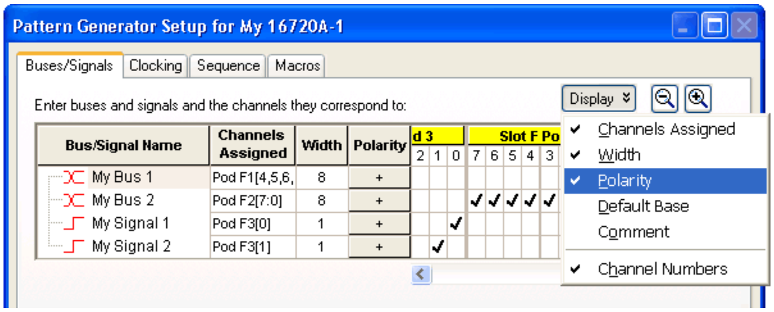
To set polarity

You can define buses and signals with negative or positive polarity. This only affects the values displayed by the pattern generator; it lets you view or enter vectors in a different polarity.

The default polarity is positive (1 = high).

1 From the main menu, select **Setup>(Pattern Generator Module)>Bus/Signals...**.

2 In the Buses/Signals tab of the Pattern Generator Setup dialog, click **Display** and choose **Polarity**.



Pattern Generator Setup for My 16720A-1

Buses/Signals | Clocking | Sequence | Macros

Enter buses and signals and the channels they correspond to:

Bus/Signal Name	Channels Assigned	Width	Polarity	d 3			Slot F Po				
				2	1	0	7	6	5	4	3
My Bus 1	Pod F1[4,5,6,	8	+								
My Bus 2	Pod F2[7:0]	8	+				✓	✓	✓	✓	✓
My Signal 1	Pod F3[0]	1	+			✓					
My Signal 2	Pod F3[1]	1	+		✓						

3 In the Polarity column that appears, toggle between + (positive) and — (negative).

To add user comments

You can attach comments to buses and signals. The comments appear in XML format configuration files.

1 From the main menu, select **Setup>(Pattern Generator Module)>Bus/Signals...**.

2 In the Buses/Signals tab of the Pattern Generator Setup dialog, click **Display** and choose **Comment**.

The Comment column appears.

3 In the Comment column, enter your comment for the bus or signal.

NOTE

Comments are intended as a descriptor to embellish a bus/signal name and not as a notepad. Comments can be up to 256 characters in length.

To add a folder

The **Add Folder...** feature adds a Windows-style folder to the bus/signal list. Use folders to help organize bus and signal names when using many bus/signal names.

1 From the main menu, select **Setup>(Pattern Generator Module)>Bus/Signals...**.

2 Right-click on a bus/signal name; then, choose **Add Folder...**.

3 In the Add Folder dialog, enter the folder name, and click **OK**.

The new folder appears directly below the highlighted bus/signal name. You can rename (see page 43) a folder just like you would a bus/signal name.

To sort bus/signal names

You can sort bus/signal names and folder names to help organize them.

1. From the main menu, select **Setup>(Pattern Generator Module)>Bus/Signals...**.
2. In the Buses/Signals tab of the Patter Generator Setup dialog, right-click on one of the bus/signal or folder names to be sorted; then, choose either **Sort>Ascending** or **Sort>Descending**.

See Also

- To add a folder (see page 51)

5
Creating Vector Sequences

Test vectors determine the pattern output at each clock cycle. Test vectors are positioned in a list called a sequence. When a sequence is run (see page 99), the list of vectors is executed in order of first vector to last vector. Vectors are always output on the rising edge of the clock.

In every pattern generator application, you have two sequences. An INIT SEQUENCE (initialization sequence) is used to place your circuit or subsystem in a known state. The initialization sequence is followed by the MAIN SEQUENCE. The main sequence is used for the actual pattern generation that stimulates your circuit under test. The INIT sequence is only executed once, while the MAIN sequence loops if you start repetitive execution.

Using Hardware and Software Instructions

In addition to test vectors, both INIT and MAIN sequences can include predefined instruction (see page 60) elements. Instructions can create Breaks, Loops, Wait for External Events or Arm, or Send an Arm signal to another instrument. The most useful instruction is the *User-Defined Macro* which lets you create reusable sequences that accept parameters. This flexibility is very useful in prototype turn-on and environmental testing.

For more information on INIT and MAIN sequences and how to create them, see the following topics.

- Creating the Initialization Sequence (see page 55)
- Creating the Main Sequence (see page 58)
- Inserting Vectors and Instructions (see page 60)
    - To insert Vectors (see page 61)
    - To insert Break instructions (see page 61)
    - To insert Wait External Event instructions (see page 62)
    - To insert Send Arm instructions (see page 64)
    - To insert Wait for Arm instructions (see page 65)
    - To insert Start/End Loop instructions (see page 65)
    - To insert User-Defined Macro instructions (see page 67)
- Filling Vectors with Automatically-Generated Patterns (see page 70)
    - To fill with Fixed patterns (see page 71)
    - To fill with Count patterns (see page 72)

    - To fill with Rotate patterns (see page 73)
    - To fill with Toggle patterns (see page 75)
    - To fill with Pseudo-Random patterns (see page 76)
- Finding Instructions or Vectors (see page 78)
- Editing Sequences (see page 79)
    - To delete lines (see page 79)
    - To cut, copy, and paste lines (see page 80)
    - To cut, copy, and paste cell text (see page 81)
    - To cut, copy, and paste columns (see page 82)
    - To go to a line number (see page 82)
    - To select lines (see page 83)
    - To navigate in vector sequences (see page 83)
- Editing Macros (see page 85)
    - To add a macro (see page 86)
    - To delete a macro (see page 86)
    - To rename a macro (see page 86)
    - To copy a macro (see page 87)
    - To define macro parameters (see page 88)
    - To use parameters in the macro sequence (see page 88)
- Setting Vector Sequence Display Options (see page 90)
    - To adjust column widths (see page 90)
    - To set the numeric base (see page 90)

Creating the Initialization Sequence

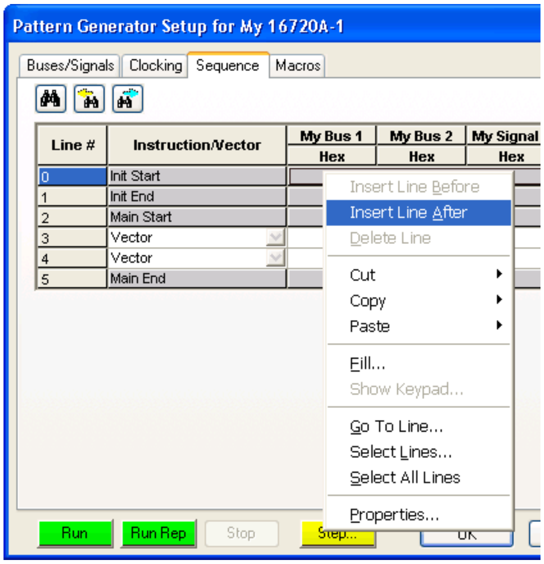
The initialization (INIT) sequence is the first of two vector sequences that appear in the Sequence display. Use the INIT sequence to put the circuit or subsystem into a known starting condition. You can also use the INIT sequence to arm a logic analyzer or oscilloscope with the Send Arm instruction to begin a measurement before the MAIN sequence begins. If you leave the INIT sequence empty, it will be ignored.

Creating the INIT Sequence

The INIT sequence can contain hardware and software instructions (see page 60) as well as vector data. However, instructions are not allowed on the first two vector lines.

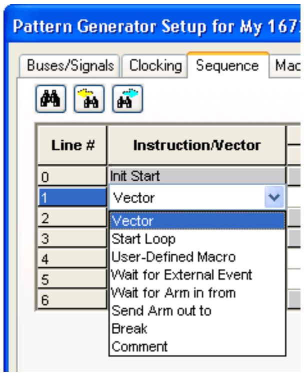
1 Select the *Sequence* tab, then select Init Start.

2 Select *Insert Line After.*



Pattern Generator Setup for My 16720A-1

Line #	Instruction/Vector	My Bus 1 Hex	My Bus 2 Hex	My Signal Hex
0	Init Start			
1	Init End			
2	Main Start			
3	Vector			
4	Vector			
5	Main End			

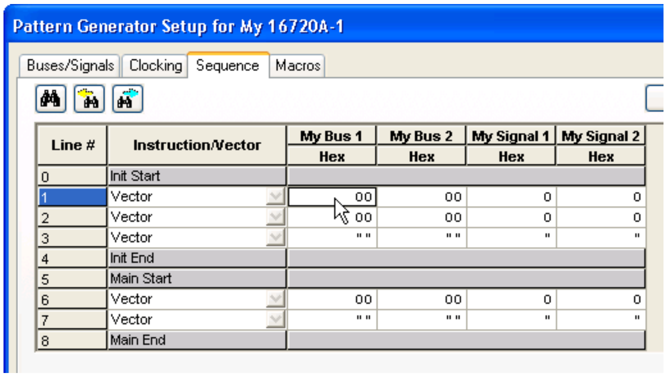


Pattern Generator Setup for My 167

Line #	Instruction/Vector
0	Init Start
1	Vector
2	
3	
4	
5	
6	

3 Repeat for each new vector line you want to insert.

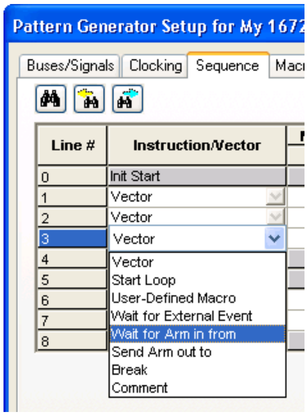
4 To specify the vector data, select the left-most character in the new vector line.



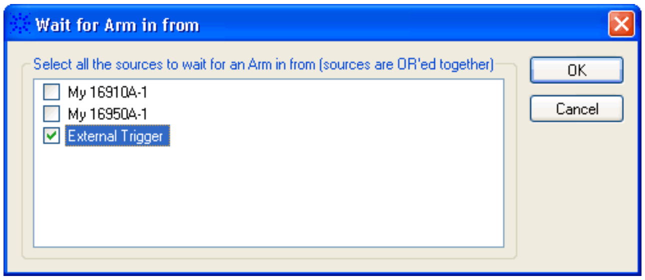
Line #	Instruction/Vector	My Bus 1 Hex	My Bus 2 Hex	My Signal 1 Hex	My Signal 2 Hex
0	Init Start				
1	Vector	00	00	0	0
2	Vector	00	00	0	0
3	Vector	""	""	"	"
4	Init End				
5	Main Start				
6	Vector	00	00	0	0
7	Vector	""	""	"	"
8	Main End				

5 Enter in the desired vector data. As you enter the information, the default cursor wrap setting will roll the cursor left-to-right.

6 Optional - If applicable, insert an instruction (see page 60) instead of entering vector data.



Line #	Instruction/Vector
0	Init Start
1	Vector
2	Vector
3	Vector
4	
5	
6	
7	
8	



Select all the sources to wait for an Arm in from (sources are OR'ed together)

- □ My 16910A-1
- □ My 16950A-1
- ☑ External Trigger

See Also

- Defining Buses and Signals (see page 39)
- Editing Sequences (see page 79)

- Inserting Vectors and Instructions (see page 60)
- Editing Macros (see page 85)
- Running and Stopping Pattern Generator Output (see page 99)

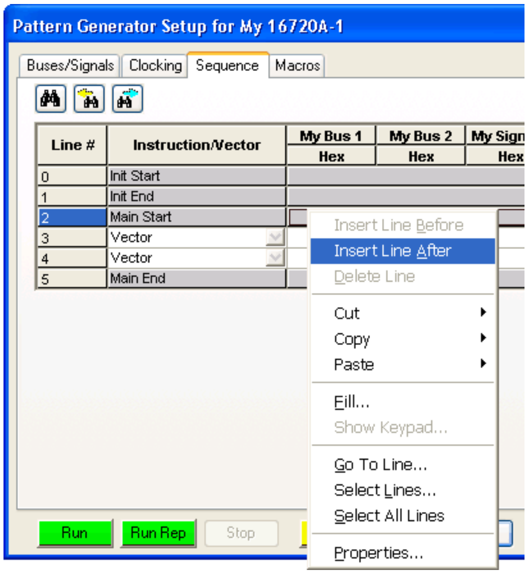
Creating the Main Sequence

The MAIN sequence is the second of two vector sequences that appear in Sequence. Use the MAIN sequence as the primary signal generation sequence. The MAIN sequence must contain at least two vectors to output.

The MAIN sequence can contain hardware and software instructions (see page 60) as well as vector data. However, instructions are not allowed on the first two vector lines or the last vector line.

1 Select the Sequence tab, then select Main Start.

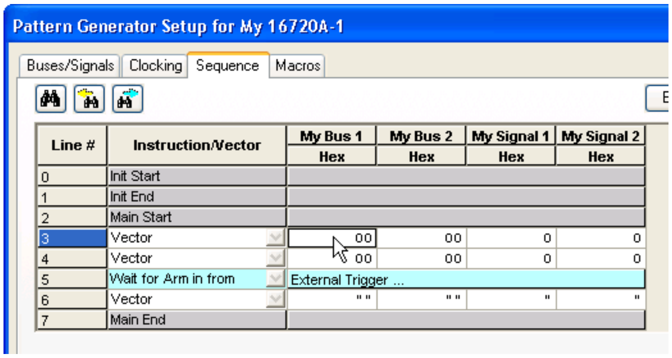
2 Select *Insert Line After*.



Line #	Instruction/Vector	My Bus 1 Hex	My Bus 2 Hex	My Sign Hex
0	Init Start			
1	Init End			
2	Main Start			
3	Vector			
4	Vector			
5	Main End			

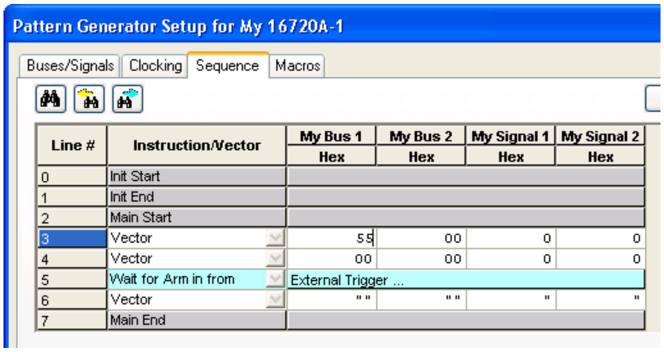
3 Repeat for each new vector line you want to insert.

4 To specify the vector data, select the left-most character in the new vector line.



Line #	Instruction/Vector	My Bus 1 Hex	My Bus 2 Hex	My Signal 1 Hex	My Signal 2 Hex
0	Init Start				
1	Init End				
2	Main Start				
3	Vector	00	00	0	0
4	Vector	00	00	0	0
5	Wait for Arm in from	External Trigger ...			
6	Vector	" "	" "	"	"
7	Main End				

5 Enter in the desired vector data. As you enter the information, the default cursor wrap setting will roll the cursor left-to-right.



Pattern Generator Setup for My 16720A-1

Line #	Instruction/Vector	My Bus 1	My Bus 2	My Signal 1	My Signal 2
		Hex	Hex	Hex	Hex
0	Init Start				
1	Init End				
2	Main Start				
3	Vector	55	00	0	0
4	Vector	00	00	0	0
5	Wait for Arm in from	External Trigger ...			
6	Vector	" "	" "	"	"
7	Main End				

6 Optional - If applicable, insert an instruction (see page 60) instead of entering vector data.

See Also

- Defining Buses and Signals (see page 39)
- Editing Sequences (see page 79)
- Inserting Vectors and Instructions (see page 60)
- Editing Macros (see page 85)
- Running and Stopping Pattern Generator Output (see page 99)

Inserting Vectors and Instructions

Vectors specify values that are output by the pattern generator. Instructions are like programming commands; they control the flow of vector output. There are two types of instructions: hardware and software.

When you insert a line, it is a Vector with *ditto* data values (that is, values that are the same as the previous vector). You can edit the data values or change the Vector to one of the other instructions.

Inserting Vectors

- To insert Vectors (see page 61)

Inserting Hardware Instructions

Hardware instructions affect pattern generator or external hardware.

- To insert Break instructions (see page 61)
- To insert Wait External Event instructions (see page 62)
- To insert Send Arm instructions (see page 64)
- To insert Wait for Arm instructions (see page 65)

Inserting Software Instructions

Software instructions only affect the execution flow of the currently running sequence (although a User-Defined Macro will likely include hardware instructions).

- To insert Start/End Loop instructions (see page 65)
- To insert User-Defined Macro instructions (see page 67)

Instruction Rules

The valid instructions vary between the Sequence tab and the Macro tab:

Valid Instructions	
Sequence Tab	**Macro Tab**
Start Loop/Stop Loop	Start Loop/Stop Loop
Break	Break
Macro	Macro
Wait for External Event	Wait for External Event
Send Arm	
Wait for Arm	

There are also rules for when each type of instruction can and cannot be used:

- Instructions cannot be in the next two lines after Init Start or Main Start, and they cannot be in the next line after Macro Start.
- Instructions cannot be in the previous line before Main End.

- Wait for External Events, Wait for Arm, Send Arm, and Break instructions must be preceded and followed by a vector line.
- There can be at most one Wait for Arm instruction and one Send Arm instruction in the Sequence tab. The Macro tab does not allow the Wait for Arm instruction or the Send Arm instruction.

See Also

- Creating the Initialization Sequence (see page 55)
- Creating the Main Sequence (see page 58)
- Filling Vectors with Automatically-Generated Patterns (see page 70)
- Editing Sequences (see page 79)
- Editing Macros (see page 85)

To insert Vectors

A Vector tells the pattern generator to create the specified patterns. The cells in a vector line are editable.

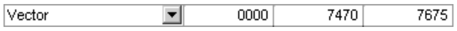
1 In the Sequence or Macros tab, select the vector that you want to insert the new vector before or after.

2 Click **Edit** and choose **Insert Line Before** or **Insert Line After**.

  Pressing the Insert key on the keyboard is the same as choosing **Insert Line Before**.

3 If the vector bus/signal data values should be the same as the previous vector, leave the *ditto* values; otherwise, enter the desired bus/signal patterns.

  You can return an edited data value to the *ditto* value by cutting the contents of the data value cell.



Vector 0000 7470 7675

See Also

- Filling Vectors with Automatically-Generated Patterns (see page 70)

To insert Break instructions

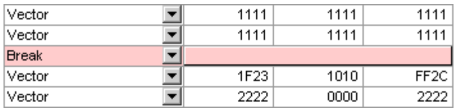
When the Break instruction is encountered, the pattern generator stops the sequence and holds the outputs of the previous vector until you select resume or stop.

1 In the Sequence or Macros tab, select the vector that you want to insert the instruction before or after.

2 Click **Edit** and choose **Insert Line Before** or **Insert Line After**.

  Pressing the Insert key on the keyboard is the same as choosing **Insert Line Before**.

3 Select the **Vector** and choose **Break**.



Vector	1111	1111	1111
Vector	1111	1111	1111
Break			
Vector	1F23	1010	FF2C
Vector	2222	0000	2222

Restrictions on Use

The Break instruction must follow data vectors. Also, the Break instruction is not allowed on the following vector lines of a sequence:

- The first or second vector of the INIT sequence.
- The first or second vector of the MAIN sequence.
- The last vector of the MAIN sequence.
- The first vector of a Macro sequence.

To insert Wait External Event instructions

The Wait External Event instruction halts the execution of the program sequence until one of four designated events (A-D) is received by the pattern generator.

When this instruction is encountered, the pattern generator stops until the specified external event happens. The external events all relate to patterns on the Wait 2, Wait 1, and Wait 0 input channels on the clock pod. Patterns on these channels are received from an external source (for example, the device under test).

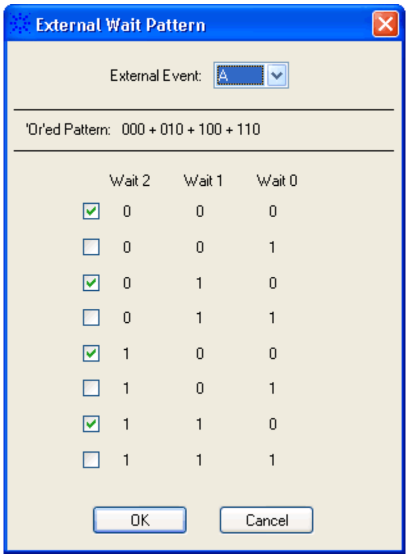
1 In the Sequence or Macros tab, select the vector that you want to insert the instruction before or after.

2 Click **Edit** and choose **Insert Line Before** or **Insert Line After**.

Pressing the Insert key on the keyboard is the same as choosing **Insert Line Before**.

3 Select the **Vector** and choose **Wait for External Event**.

4 In the External Event dialog, select the **External Event** (A-D).



External Event: A

'Or'ed Pattern: 000 + 010 + 100 + 110

	Wait 2	Wait 1	Wait 0
☑	0	0	0
□	0	0	1
☑	0	1	0
□	0	1	1
☑	1	0	0
□	1	0	1
☑	1	1	0
□	1	1	1

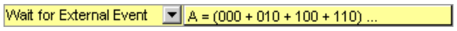
5 To define the external event, check the boxes corresponding to the Wait input patterns that can signal the event.

If no check boxes are selected, this is the "None" or "Always Wait" case which causes the pattern generator to wait without any possibility of resuming.

CAUTION You cannot resume pattern generator output after an "Always Wait" event. Use the Break instruction instead of the "Always Wait" case because you can resume pattern generator output after a break.

If all of the check boxes are selected, this is the "Any" or "Never Wait" case which does not cause the pattern generator to wait at all.

6 Click **OK**.


Wait for External Event A = (000 + 010 + 100 + 110) ...

Restrictions on Use

The Wait External Event instruction, as with all hardware instructions, must follow data vectors. Wait External Events are not allowed on the following vector lines of a sequence:

- The first or second vector of the INIT sequence.
- The first or second vector of the MAIN sequence.
- The last vector of the MAIN sequence.
- The first vector of a Macro sequence.

To insert Send Arm instructions

The Send Arm instruction creates an arming signal when the instruction is executed. This arming signal can be used by other modules in the frame or by external instruments.

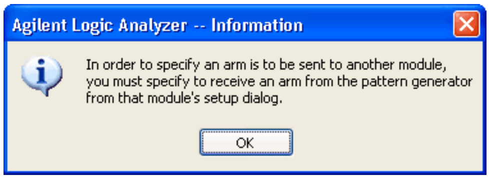
1 In the Sequence tab, select the vector that you want to insert the instruction before or after.

2 Click **Edit** and choose **Insert Line Before** or **Insert Line After**.

  Pressing the Insert key on the keyboard is the same as choosing **Insert Line Before**.

3 Select the **Vector** and choose **Send Arm out to**.

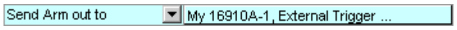
  The following dialog tells you that you must set up modules, or the external trigger, to receive the arm signal separately.



In order to specify an arm is to be sent to another module, you must specify to receive an arm from the pattern generator from that module's setup dialog.

4 Click **OK** to close the information dialog.

5 Set up the arm signal recipients (see "To arm one module with another module's trigger" (in the online help) or "To trigger other instruments - trigger out" (in the online help)).



Send Arm out to | My 16910A-1, External Trigger ...

As an example, you could set up a logic analyzer to trigger when the pattern generator reaches a particular row. In this case, you could set up that line with a Send Arm instruction and set up the logic analyzer to trigger when it receives an arm signal from the pattern generator.

Restrictions on Use The Send Arm instruction, as with all hardware instructions, must follow data vectors. Also, the Send Arm instruction can only be used one time in a sequence, and consequently it is not allowed in a user macro or a repeat loop. The Send Arm instruction is not allowed on the following vector lines of a sequence:

- The first or second vector of the INIT sequence.
- The first or second vector of the MAIN sequence.
- The last vector of the MAIN sequence.

See Also

- "Triggering From, and Sending Triggers To, Other Modules/Instruments" (in the online help)

To insert Wait for Arm instructions

This instruction causes the pattern generator to pause until an arm signal is received from the specified module.

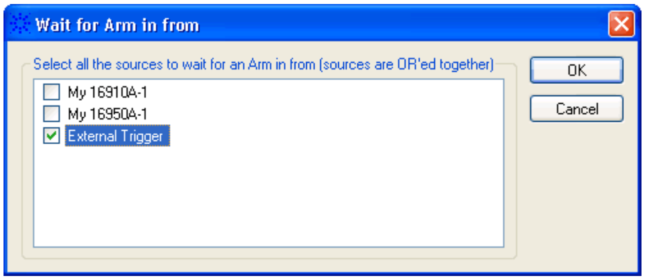
1 In the Sequence tab, select the vector that you want to insert the instruction before or after.

2 Click **Edit** and choose **Insert Line Before** or **Insert Line After**.

   Pressing the Insert key on the keyboard is the same as choosing **Insert Line Before**.

3 Select the **Vector** and choose **Wait for Arm in from**.

4 In the "Wait for Arm in from" dialog, select the module from which the arm signal should be received.

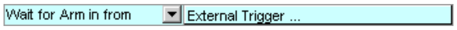


Wait for Arm in from

Select all the sources to wait for an Arm in from (sources are OR'ed together)

□ My 16910A-1
□ My 16950A-1
☑ External Trigger

5 Click **OK**.



Wait for Arm in from | External Trigger ...

Restrictions on Use The Wait for Arm instruction, as with all hardware instructions, must follow data vectors. Also, the Wait for Arm instruction can only be used one time in a sequence, and consequently it is not allowed in a user macro or a repeat loop. The Wait for Arm instruction is not allowed on the following vector lines of a sequence:

- The first or second vector of the INIT sequence.
- The first or second vector of the MAIN sequence.
- The last vector of the MAIN sequence.

To insert Start/End Loop instructions

The Start Loop instruction inserts the start and end vectors of a repeat loop, along with one blank data vector row.

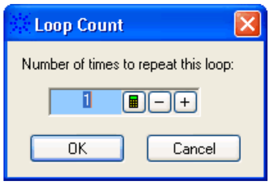
1 In the Sequence or Macros tab, select the vector that you want to insert the instruction before or after.

2 Click **Edit** and choose **Insert Line Before** or **Insert Line After**.

Pressing the Insert key on the keyboard is the same as choosing **Insert Line Before**.

3 Select the **Vector** and choose **Start Loop**.

4 In the Loop Count dialog, enter the number of times to repeat the loop.



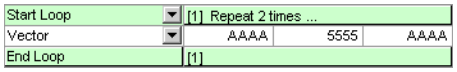
Number of times to repeat this loop: 1

At run time, the loop count determines the number of times to expand the loop's body into individual vectors.

5 Click **OK**.

6 Edit the blank data vector line.

7 Insert or copy vectors and instructions into the loop.

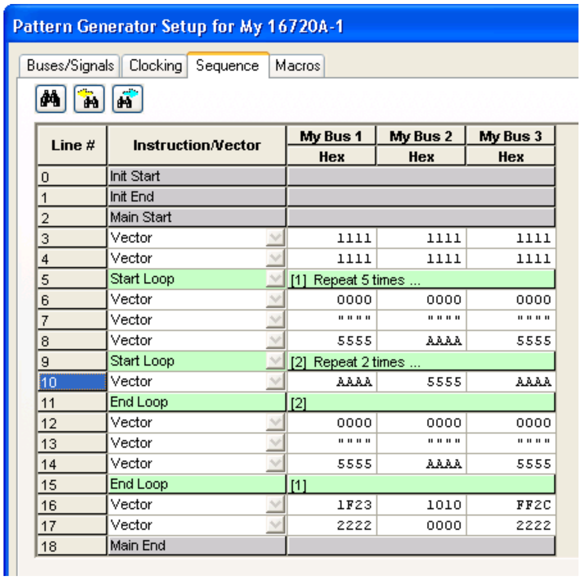


Start Loop	[1] Repeat 2 times ...		
Vector	AAAA	5555	AAAA
End Loop	[1]		

Both the start and end vectors of a repeat loop are removed from the sequence if either one is included in a delete operation.

Nested Start/End Loop Instructions

This example shows a nested loop. Loop[1] is a walking ones pattern that is repeated five times. Loop[2] is a bit pattern of all ones that is repeated twice directly in the middle of the walking ones pattern of Loop[1].



Line #	Instruction/Vector	My Bus 1	My Bus 2	My Bus 3
		Hex	Hex	Hex
0	Init Start			
1	Init End			
2	Main Start			
3	Vector	1111	1111	1111
4	Vector	1111	1111	1111
5	Start Loop	[1] Repeat 5 times ...		
6	Vector	0000	0000	0000
7	Vector	""""	""""	""""
8	Vector	5555	AAAA	5555
9	Start Loop	[2] Repeat 2 times ...		
10	Vector	AAAA	5555	AAAA
11	End Loop	[2]		
12	Vector	0000	0000	0000
13	Vector	""""	""""	""""
14	Vector	5555	AAAA	5555
15	End Loop	[1]		
16	Vector	1F23	1010	FF2C
17	Vector	2222	0000	2222
18	Main End			

Restrictions on Use

The following instructions cannot be used in a repeat loop:

- The Send Arm instruction.
- The Wait for Arm instruction.

The Start/End Loop instruction is not allowed on the following vector lines of a sequence:

- The first or second vector of the INIT sequence.
- The first or second vector of the MAIN sequence.
- The last vector of the MAIN sequence.
- The first vector of a Macro sequence.

To insert User-Defined Macro instructions

Macros are similar to subroutines in a programming language. Macros let you define a group of vectors by name so they can be referred to at different points in in the sequence.

1 In the Sequence or Macros tab, select the vector that you want to insert the instruction before or after.

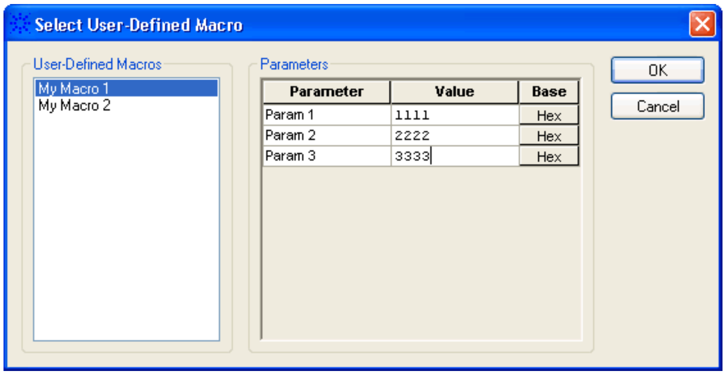
2 Click **Edit** and choose **Insert Line Before** or **Insert Line After**.

   Pressing the Insert key on the keyboard is the same as choosing **Insert Line Before**.

3 Select the **Vector** and choose **User-Defined Macro**.

4 In the Select User-Defined Macro dialog, select the macro to call, and enter the parameter values.

NOTE Parameters must be added to the macro before values can be assigned to them in the call.

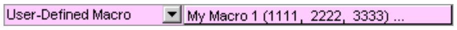


User-Defined Macros: My Macro 1, My Macro 2

Parameter	Value	Base
Param 1	1111	Hex
Param 2	2222	Hex
Param 3	3333	Hex

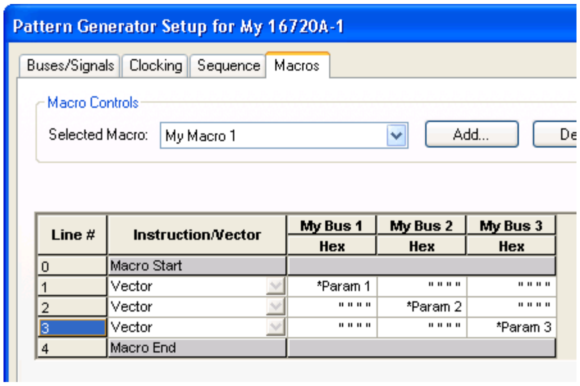
When entering a parameter value, the number of digits is not limited. For example, if you enter 24 bits of data for a parameter and that parameter is used on a 8-bit bus, the most significant 16 bits are ignored.

5 Click **OK**.



User-Defined Macro | My Macro 1 (1111, 2222, 3333) ...

At run time, the user-defined macro is expanded into its corresponding vectors.



Selected Macro: My Macro 1

Line #	Instruction/Vector	My Bus 1	My Bus 2	My Bus 3
		Hex	Hex	Hex
0	Macro Start			
1	Vector	*Param 1	" " " "	" " " "
2	Vector	" " " "	*Param 2	" " " "
3	Vector	" " " "	" " " "	*Param 3
4	Macro End			

Restrictions on Use

The User-Defined Macro instruction is not allowed on the following vector lines of a sequence:

- The first or second vector of the INIT sequence.
- The first or second vector of the MAIN sequence.

- The last vector of the MAIN sequence.
- The first vector of a Macro sequence.

See Also

- Editing Macros (see page 85)

Filling Vectors with Automatically-Generated Patterns

There are five automatic pattern fill utilities available to quickly generate and insert signal patterns into your sequences and user-defined macros. By using pattern fills, you not only save time generating generic patterns, but you also guarantee the accuracy of the sequence.

1 In the Sequence or Macros tab, click **Edit** and choose **Fill...**.

2 In the Fill dialog, select the **Type of Fill** and pattern generation options for that type. See:

   - To fill with Fixed patterns (see page 71)
   - To fill with Count patterns (see page 72)
   - To fill with Rotate patterns (see page 73)
   - To fill with Toggle patterns (see page 75)
   - To fill with Pseudo-Random patterns (see page 76)

3 Select the **Bus/Signal** on which to fill.

4 Enter the **Starting Line**.

5 Enter the number of **Vectors to Generate**.

6 Select the **Edit Mode**:

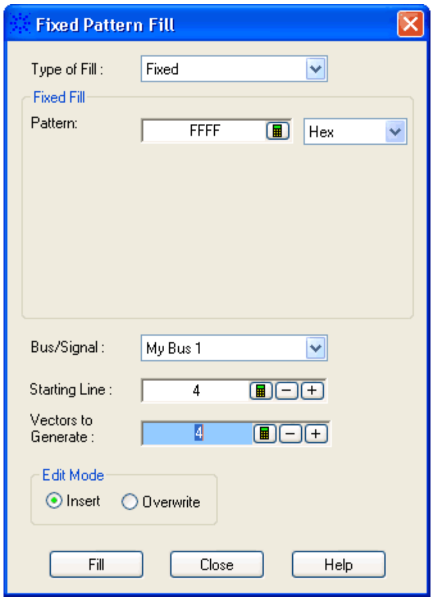
   - **Insert** - inserts at or before the selected sequence line.
   - **Overwrite** - overwrites existing vectors starting at the selected sequence line.

7 Click **Fill**.

To fill with Fixed patterns

The Fixed fill simply takes the given pattern and puts into the Sequence spreadsheet the number of times specified by the number of vectors.

1 In the Fill dialog, select the **Fixed** type of fill.



Fixed Pattern Fill

Type of Fill : Fixed

Fixed Fill

Pattern: FFFF Hex

Bus/Signal : My Bus 1

Starting Line : 4

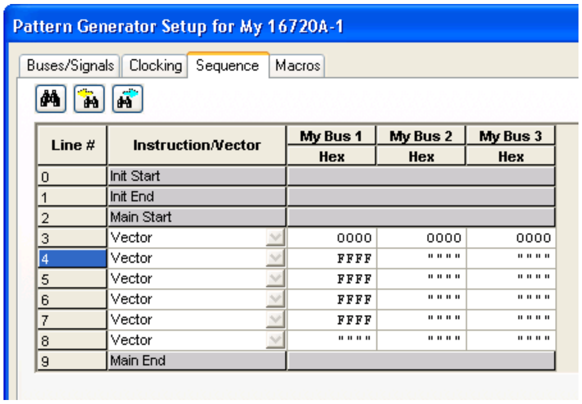
Vectors to Generate : 4

Edit Mode

● Insert ○ Overwrite

2 In the Fixed Fill box, select the number base, and enter the pattern to be generated.

3 Specify the other fill options (see Filling Vectors with Automatically-Generated Patterns (see page 70)).



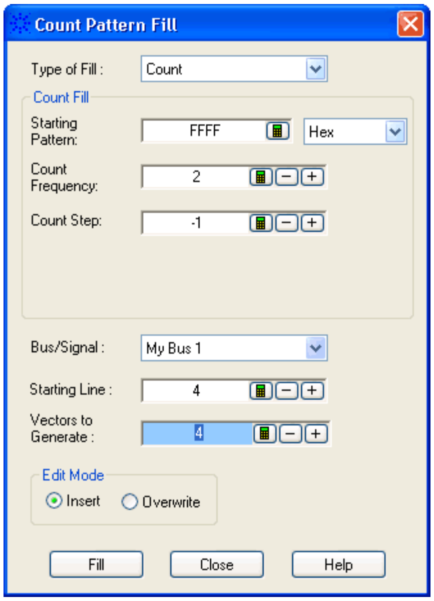
Pattern Generator Setup for My 16720A-1

Line #	Instruction/Vector	My Bus 1	My Bus 2	My Bus 3
		Hex	Hex	Hex
0	Init Start			
1	Init End			
2	Main Start			
3	Vector	0000	0000	0000
4	Vector	FFFF	""""	""""
5	Vector	FFFF	""""	""""
6	Vector	FFFF	""""	""""
7	Vector	FFFF	""""	""""
8	Vector	""""	""""	""""
9	Main End			

To fill with Count patterns

The Count fill is used to create vectors that increase or decrease.

1 In the Fill dialog, select the **Count** type of fill.



Count Pattern Fill

Type of Fill : Count

Count Fill

Starting Pattern: FFFF Hex

Count Frequency: 2

Count Step: -1

Bus/Signal : My Bus 1

Starting Line : 4

Vectors to Generate : 4

Edit Mode

● Insert ○ Overwrite

2 In the Count Fill box:

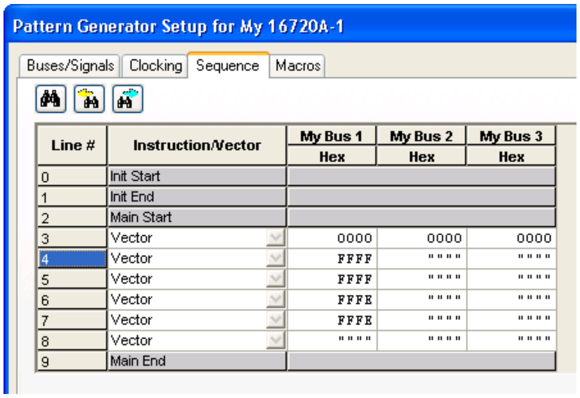
a Enter the **Starting Pattern** to begin counting from.

b Enter the **Count Frequency** which specifies how often the step should be applied.

For example, if the frequency is 4, there will be four consecutive vectors with the same pattern, and the next four vectors after that will differ by the amount of the step.

c Enter the **Count Step** which specifies the amount that each vector increases or decreases.

3 Specify the other fill options (see Filling Vectors with Automatically-Generated Patterns (see page 70)).


Pattern Generator Setup for My 16720A-1

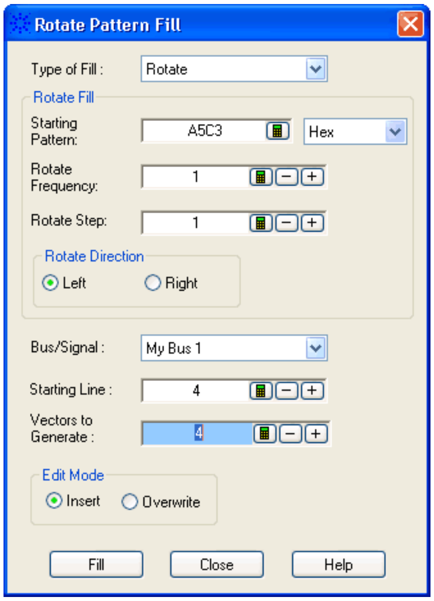
Line #	Instruction/Vector	My Bus 1	My Bus 2	My Bus 3
		Hex	Hex	Hex
0	Init Start			
1	Init End			
2	Main Start			
3	Vector	0000	0000	0000
4	Vector	FFFF	""""	""""
5	Vector	FFFF	""""	""""
6	Vector	FFFE	""""	""""
7	Vector	FFFE	""""	""""
8	Vector	""""	""""	""""
9	Main End			

To fill with Rotate patterns

The Rotate fill is best understood as changing binary values. For example, if the starting value is 0000 0001 in binary and the step is one, then the next value will be 0000 0010, and the value after that will be 0000 0100. If the starting value is 0000 0001 and the step is two, then the next value will be 0000 0100.

If the starting pattern is all 1's or all 0's, the "rotation" has no effect, and the Rotate fill is the same as a Fixed fill.

1 In the Fill dialog, select the **Rotate** type of fill.


Rotate Pattern Fill

Type of Fill : Rotate

Rotate Fill

Starting Pattern: A5C3 Hex

Rotate Frequency: 1

Rotate Step: 1

Rotate Direction

● Left ○ Right

Bus/Signal : My Bus 1

Starting Line : 4

Vectors to Generate : 4

Edit Mode

● Insert ○ Overwrite

2 In the Rotate Fill box:

   a Enter the **Starting Pattern** to begin rotating from.

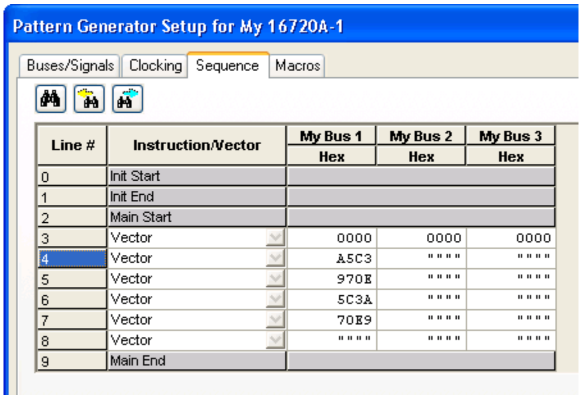
   b Enter the **Rotate Frequency** which specifies how often the rotate should occur, in other words, how many values will be the same before the rotation causes the values to change.

   For example, if the frequency is 4, there will be four consecutive vectors with the same pattern, and the next four vectors after that will be rotated by the amount of the step.

   c Enter the **Rotate Step** which specifies the number of digits to rotate.

   d Select the left or right **Rotate Direction**.

3 Specify the other fill options (see Filling Vectors with Automatically-Generated Patterns (see page 70)).

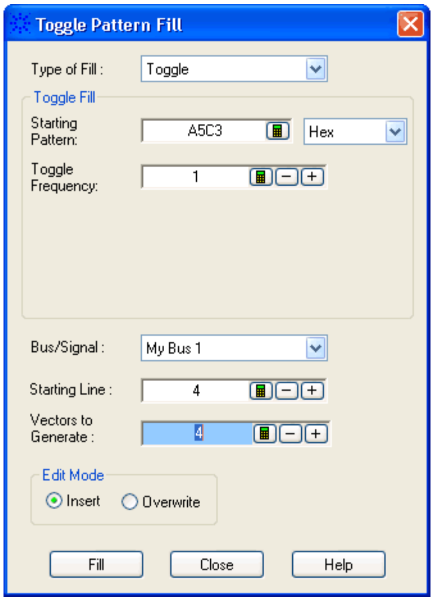


Line #	Instruction/Vector	My Bus 1	My Bus 2	My Bus 3
		Hex	Hex	Hex
0	Init Start			
1	Init End			
2	Main Start			
3	Vector	0000	0000	0000
4	Vector	A5C3	"""""	"""""
5	Vector	970E	"""""	"""""
6	Vector	5C3A	"""""	"""""
7	Vector	70E9	"""""	"""""
8	Vector	"""""	"""""	"""""
9	Main End			

To fill with Toggle patterns

The Toggle fill is best understood as changing binary values. If the starting value is 0101 0101, and the step is one, then the next value is 1010 1010. In other words, all of the 1's are turned into 0's and vice versa.

1 In the Fill dialog, select the **Toggle** type of fill.



Toggle Pattern Fill

Type of Fill : Toggle

Toggle Fill

Starting Pattern: A5C3 Hex

Toggle Frequency: 1

Bus/Signal : My Bus 1

Starting Line : 4

Vectors to Generate : 4

Edit Mode

● Insert ○ Overwrite

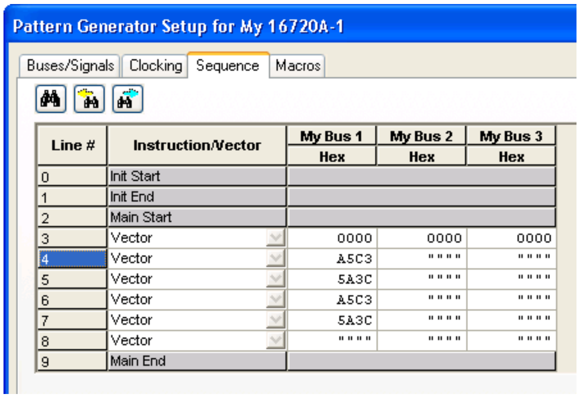
2 In the Toggle Fill box:

a Enter the **Starting Pattern** to begin toggling from.

b Enter the **Toggle Frequency** which specifies how often the toggle should occur, in other words, how many values will be the same before the toggle causes the values to change.

For example, if the frequency is 4, there will be four consecutive vectors with the same pattern, and the next four vectors after that will be toggled.

3 Specify the other fill options (see Filling Vectors with Automatically-Generated Patterns (see page 70)).



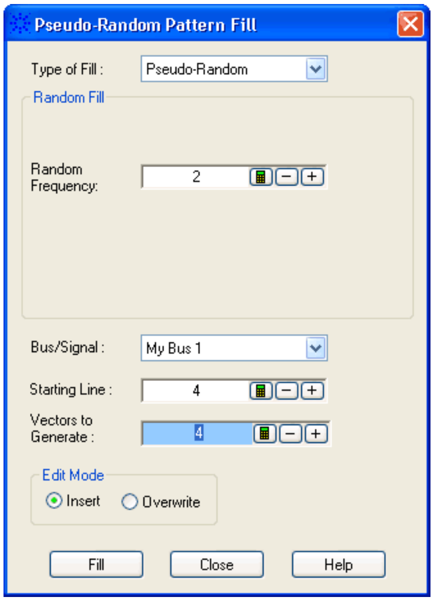
Line #	Instruction/Vector	My Bus 1	My Bus 2	My Bus 3
		Hex	Hex	Hex
0	Init Start			
1	Init End			
2	Main Start			
3	Vector	0000	0000	0000
4	Vector	A5C3	""""	""""
5	Vector	5A3C	""""	""""
6	Vector	A5C3	""""	""""
7	Vector	5A3C	""""	""""
8	Vector	""""	""""	""""
9	Main End			

To fill with Pseudo-Random patterns

The Pseudo-Random fill simply generates pseudo-random numbers and inserts them into the sequence spreadsheet.

The frequency indicates how often a new random number should be generated.

1 In the Fill dialog, select the **Pseudo-Random** type of fill.



Type of Fill : Pseudo-Random

Random Frequency: 2

Bus/Signal : My Bus 1

Starting Line : 4

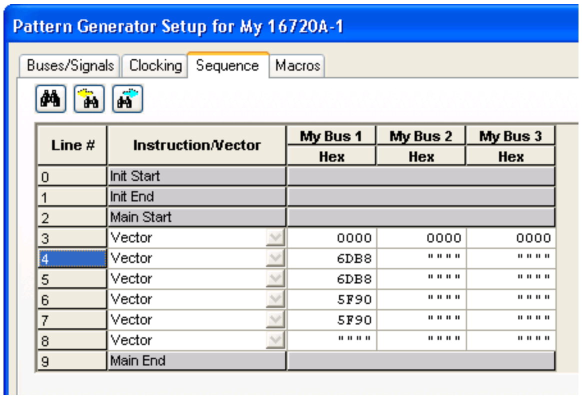
Vectors to Generate : 4

● Insert ○ Overwrite

2 In the Random Fill box, enter the **Random Frequency** which specifies how often a new random number should be generated.

For example, if the frequency is 4, there will be four consecutive vectors with the same pattern, and the next four vectors after that will be the next random number.

3 Specify the other fill options (see Filling Vectors with Automatically-Generated Patterns (see page 70)).



Pattern Generator Setup for My 16720A-1

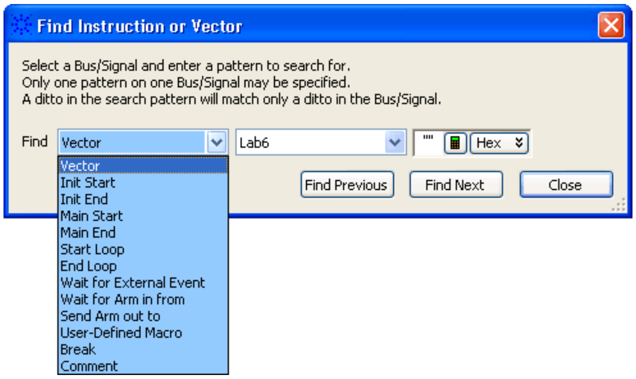
Line #	Instruction/Vector	My Bus 1	My Bus 2	My Bus 3
		Hex	Hex	Hex
0	Init Start			
1	Init End			
2	Main Start			
3	Vector	0000	0000	0000
4	Vector	6DB8	""""	""""
5	Vector	6DB8	""""	""""
6	Vector	5F90	""""	""""
7	Vector	5F90	""""	""""
8	Vector	""""	""""	""""
9	Main End			

Finding Instructions or Vectors

In the Pattern Generator Setup dialog's Sequence tab, you can search for instructions or vectors in the sequence.

1 Click the Find Instruction/Vector button.

2 In the Find Instruction or Vector dialog, select the type of instruction or vector you want to find.



Find Instruction or Vector

Select a Bus/Signal and enter a pattern to search for.
Only one pattern on one Bus/Signal may be specified.
A ditto in the search pattern will match only a ditto in the Bus/Signal.

Find: Vector | Lab6 | "" | Hex

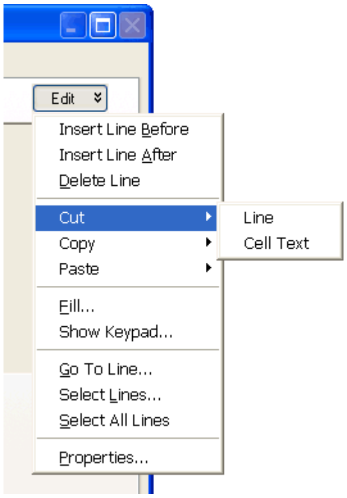
- If you selected **Vector**, also select the label and enter the value to find.
- If you selected **User-Defined Macro** or **Comment**, you can also enter a case-sensitive string to find. If you don't enter a string, any macro or comment will be found.

3 Click **Find Previous** or **Find Next** to begin the search.

4 Click **Close** to close the Find Instruction or Vector dialog.

When the dialog is closed, you can click the Find Previous Instruction/Vector button or the Find Next Instruction/Vector button to continue searching.

Editing Sequences



- To delete lines (see page 79)
- To cut, copy, and paste lines (see page 80)
- To cut, copy, and paste cell text (see page 81)
- To cut, copy, and paste columns (see page 82)
- To go to a line number (see page 82)
- To select lines (see page 83)
- To navigate in vector sequences (see page 83)

To delete lines

CAUTION When you delete sequence lines, they are permanently removed. If you want to place them in a temporary storage buffer, cut (see page 80) them instead.

1. In the Sequence or Macro tab, select the vector sequence lines to be deleted (see To select lines (see page 83)).
2. Either: press the Delete key on the keyboard, right-click and choose **Delete Line(s)**, or click **Edit** and choose **Delete Line(s)**.

Restrictions on Use

You are not allowed to delete the following sequence lines:

- The Init Start line.
- The Init End line.
- The Main Start line.
- The Main End line.
- The Macro Start line.
- The Macro End line.

A delete will not be performed if the result of the delete places a non-Vector instruction on one of the following lines:

- The first or second vector of the INIT sequence.
- The first or second vector of the MAIN sequence.

- The last vector of the MAIN sequence.
- The first vector of a Macro sequence.

A delete will not be performed if the result of the delete does either of the following:

- Places a hardware instruction immediately after another non-Vector instruction.
- Causes the MAIN sequence to contain fewer than two data vectors.

See Also

- To select lines (see page 83)

To cut, copy, and paste lines

NOTE When you use the **Cut** and **Copy** operations, you are placing the sequence lines in a temporary storage buffer. All subsequent **Paste** operations will insert sequence lines from the storage buffer until new lines are cut or copied. When you use the **Delete** operation, the sequence lines are not placed in the temporary buffer; they are just deleted.

To cut lines

1 In the Sequence or Macro tab, select the vector sequence lines to be cut (see To select lines (see page 83)).

2 Either: press Ctrl+x, right-click and choose **Cut>Line(s)**, or click **Edit** and choose **Cut>Line(s)**.

NOTE Cutting (or deleting) all the sequence lines causes the sequence to be reset to the power-up state.

To copy lines

1 In the Sequence or Macro tab, select the vector sequence lines to be copied (see To select lines (see page 83)).

2 Either: press Ctrl+c, right-click and choose **Copy>Line(s)**, or click **Edit** and choose **Copy>Line(s)**.

To paste lines

1 In the Sequence or Macro tab, position the cursor on the line at which lines are to be pasted. (If multiple lines are selected, the cursor is positioned on the last one.)

2 Either: press Ctrl+v, right-click and choose **Paste>Line(s)**, or click **Edit** and choose **Paste>Line(s)**.

Restrictions on Use

The summary rules are:

- Main Start, Main End, Init Start, Init End, Macro Start and Macro End lines cannot be cut, copied, pasted or deleted. These are completely non-editable lines.

See Also

- To select lines (see page 83)

- To delete lines (see page 79)
- To cut, copy, and paste cell text (see page 81)
- To cut, copy, and paste columns (see page 82)

To cut, copy, and paste cell text

To cut cell text

1 In the Sequence or Macro tab, highlight the cells to be cut by dragging, clicking, Shift-clicking, or Ctrl-clicking them.

2 Either: press Ctrl+x, right-click and choose **Cut>Cell Text**, or click **Edit** and choose **Cut>Cell Text**.

To copy cell text

1 In the Sequence or Macro tab, highlight the cells to be copied by dragging, clicking, Shift-clicking, or Ctrl-clicking them.

2 Either: press Ctrl+c, right-click and choose **Copy>Cell Text**, or click **Edit** and choose **Copy>Cell Text**.

To paste cell text

1 In the Sequence or Macro tab, select the cell at which cell text is to be pasted. (If multiple cells are selected, the upper left-most cell is where cell text will be pasted.)

2 Either: press Ctrl+v, right-click and choose **Paste>Cell Text**, or click **Edit** and choose **Paste>Cell Text**.

Restrictions on Use

The summary rules are:

- Cells copied from non-vector lines become ditto characters when they are pasted.
- Cells on the Main Start, Main End, Init Start, Init End, Macro Start and Macro End lines cannot be cut, copied, pasted or deleted. These are completely non-editable lines.
- If you cut the contents of a cell, its value reverts to all quotes (ditto characters).
- When a value is copied from one column to another, the value is shown in the base of the pasted column even if this makes the displayed digits different. For example, if the value "1111" is copied from a binary column and pasted into a hex column, it is shown as "F".
- When a value is copied from one column to another and the value is outside of the range of the pasted column, most significant bits from the paste buffer will be truncated. For example, if the value "1234 5678" is copied from a binary column eight bits wide is pasted into a binary column four bits wide, the value pasted is "5678".

See Also

- To cut, copy, and paste lines (see page 80)
- To cut, copy, and paste columns (see page 82)

To cut, copy, and paste columns

NOTE

When you use the **Cut** and **Copy** operations, you are placing the columns in a temporary storage buffer. All subsequent **Paste** operations will insert columns from the storage buffer until new columns are cut or copied. When you use the **Delete** operation, the columns are not placed in the temporary buffer; they are just deleted.

To cut columns

1. In the Sequence or Macro tab, select the columns to be cut by clicking, Shift-clicking, or Ctrl-clicking the bus/signal names at the top of the column.
2. Either: press Ctrl+x, right-click and choose **Cut**, or click **Edit** and choose **Cut**.

To copy columns

1. In the Sequence or Macro tab, select the columns to be copy by clicking, Shift-clicking, or Ctrl-clicking the bus/signal names at the top of the column.
2. Either: press Ctrl+c, right-click and choose **Copy**, or click **Edit** and choose **Copy**.

To paste columns

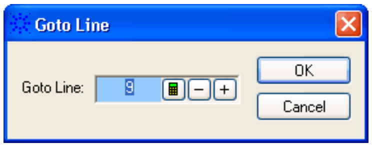
1. In the Sequence or Macro tab, select the column at which columns are to be pasted by clicking the bus/signal name at the top of the column. (If multiple columns are selected, the right-most selected column is where columns will be pasted.)
2. Either: press Ctrl+v, right-click and choose **Paste**, or click **Edit** and choose **Paste**.

To delete columns

1. In the Sequence or Macro tab, select the columns to be deleted by clicking, Shift-clicking, or Ctrl-clicking the bus/signal names at the top of the column.
2. Either: press the Delete key on the keyboard, right-click and choose **Delete>Columns**, or click **Edit** and choose **Delete>Columns**.

To go to a line number

1. In the Sequence or Macro tab, either: right-click on any part of a line and choose **Go To Line...**, or click **Edit** and choose **Go To Line....**
2. In the Goto Line dialog that appears, enter the desired line number.



Goto Line: 9

3. Click **OK**.

See Also

- To navigate in vector sequences (see page 83)

To select lines

You can select vector sequence lines in order to perform an editing function on them (like delete, cut, copy, or paste).

To select multiple lines

1 In the Line # column, highlight the lines you want to select (by clicking, Shift-clicking, or Ctrl-clicking the line numbers).

To select all lines

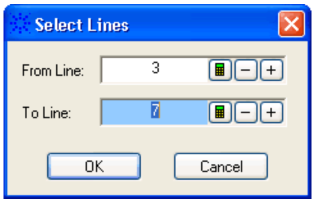
1 In the Sequence or Macro tab, click **Edit** and choose **Select All Lines**.

To select a range of lines

1 Use the Select Lines function to select a number of lines at once:

   a In the Sequence or Macro tab, click **Edit** and choose **Select Lines**.

   b In the Select Lines dialog, enter the starting and ending lines of the range to select.



From Line: 3

To Line: 7

   c Click **OK**.

See Also

- To delete lines (see page 79)
- To cut, copy, and paste lines (see page 80)
- To navigate in vector sequences (see page 83)

To navigate in vector sequences

- In the Sequence or Macros tab, press one of the following keys on the keyboard:

Key Pressed	Action Taken
Enter Key	Move to cell immediately underneath currently selected cell. If the current cell is at the bottom of the sequence, a new line is automatically inserted.
Tab Key	Move to cell immediately to the right of the currently selected cell. If the cell is at the far right, move to the leftmost cell in the line underneath.

Shift+Tab Key	Move to cell immediately to the left of the currently selected cell. If the cell is at the far left, move to the rightmost cell in the line above.
Arrow Keys	Move in the direction of the key.
Home	Move to the first line of the sequence.
End	Move to the last line of the sequence.
Page Up	Scroll upward by one screen worth of data.
Page Down	Scroll downward by one screen worth of data.
Insert	Causes the same action as selecting Insert Line Before.
Delete	Causes the same action as selecting Delete Line(s).

See Also

- To go to a line number (see page 82)

Editing Macros

"*Macros*" (in the online help) are like subroutines in a programming language. If you'd like to use the same set of vectors at multiple places in a sequence, you can define them as a macro.

You can pass parameter values to a macro. This lets a macro output different vectors at different points in the sequence (depending on the parameter values passed in the call). Parameters are like variables that can be used in place of vector values in the macro sequence.

NOTE

The number of bits in macro parameters is not limited. If the bus/signal that uses the parameter has fewer bits, the most significant bits of the parameter value are truncated.

Macros are called using the User-Defined Macro (see page 67) instruction. Macros may be inserted into the INIT or MAIN sequences of the vectors in the Sequence tab or into other macros (in other words, nested).

NOTE

Take care to avoid infinite loops (macro 0 calls macro 1, and macro 1 calls macro 0).

Inserting instructions, filling vectors, and editing sequences in the Macros tab is the same as in the Sequence tab, except that:

- Parameters can be set and cleared in vector data fields.
- The Send Arm and Wait for Arm instructions cannot be used. (These instructions can only be used once in a sequence, and a macro may be called many times in a sequence.)

A macro sequence can contain hardware and software instructions (see page 60) as well as vector data. However, instructions are not allowed on the first vector line.

When the pattern generator compiles the test vectors, macros and parameters are expanded into a flattened list of test vectors.

- To add a macro (see page 86)
- To delete a macro (see page 86)
- To rename a macro (see page 86)
- To copy a macro (see page 87)
- To define macro parameters (see page 88)
- To use parameters in the macro sequence (see page 88)

To use macros

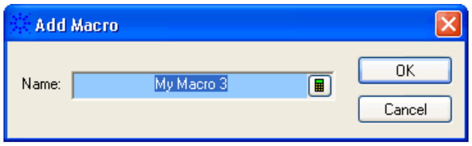
1. Create the macro, and define its parameters (if any).
2. Edit the macro sequence. If there are parameters, set them in the desired vector data fields.
3. Insert calls to the macro (in the main sequence or in other macro sequences) using the User-Defined Macro (see page 67) instruction.

See Also

- Inserting Vectors and Instructions (see page 60)
- Filling Vectors with Automatically-Generated Patterns (see page 70)
- Editing Sequences (see page 79)

To add a macro

1. In the Macro tab, click **Add...**.
2. In the Add Macro dialog, enter the name of the new macro.



Name: My Macro 3

3. Click **OK**.

See Also

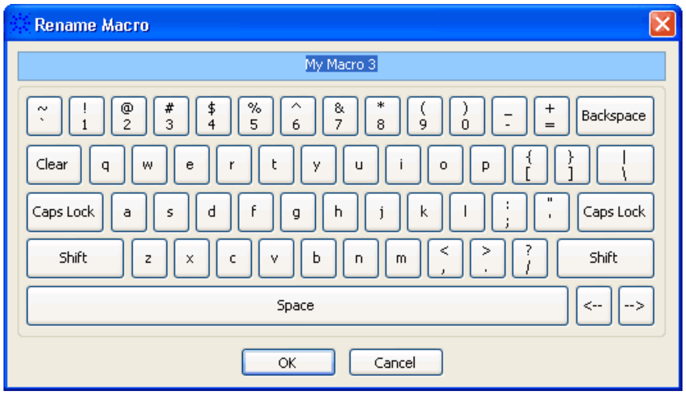
- To rename a macro (see page 86)
- To delete a macro (see page 86)
- To copy a macro (see page 87)
- To define macro parameters (see page 88)
- To use parameters in the macro sequence (see page 88)

To delete a macro

1. In the Macro tab, select the **Macro Name** to be deleted.
2. Click **Delete**.

To rename a macro

1. In the Macro tab, select the **Macro Name** to rename.
2. Click **Rename...**.
3. In the Rename Macro dialog, enter the new name of the macro.



My Macro 3

4 Click **OK**.

See Also

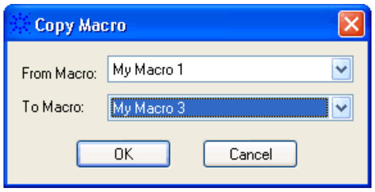
- To add a macro (see page 86)
- To delete a macro (see page 86)
- To copy a macro (see page 87)
- To define macro parameters (see page 88)
- To use parameters in the macro sequence (see page 88)

To copy a macro

You can overwrite one macro with the sequence and parameters from another macro. Use this feature to create a new, but slightly different macro from the original macro.

1 In the Macro tab, click **Copy...**.

2 In the Copy Macro dialog, select the names of the macro to copy from and the macro to copy to.



From Macro: My Macro 1

To Macro: My Macro 3

3 Click **OK**.

See Also

- To add a macro (see page 86)
- To delete a macro (see page 86)
- To rename a macro (see page 86)
- To define macro parameters (see page 88)
- To use parameters in the macro sequence (see page 88)

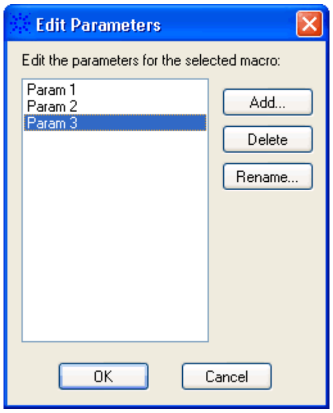
To define macro parameters

1 In the Macro tab, select the **Macro Name** whose parameters you wish to define.

2 Click **Parameters...**.

3 In the Parameters dialog:

- Click **Add...** to add a parameter and specify its name.
- Click **Delete...** to delete a parameter.



Edit the parameters for the selected macro:

Param 1
Param 2
Param 3

4 When you are done defining parameters, click **OK**.

See Also

- To add a macro (see page 86)
- To delete a macro (see page 86)
- To rename a macro (see page 86)
- To copy a macro (see page 87)
- To use parameters in the macro sequence (see page 88)

To use parameters in the macro sequence

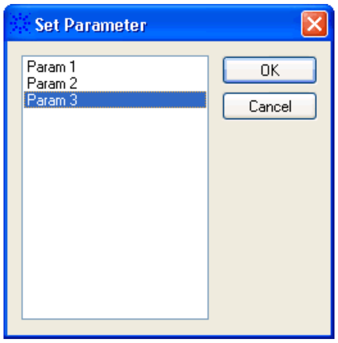
NOTE Because parameters replace a data field within data vectors, you can only set them on an existing data vector. They cannot be set on instruction vectors.

To set a parameter

1 In the Macro tab, select the data field that should use the parameter. (This positions the cursor.)

2 Click **Edit** and choose **Set Parameter**.

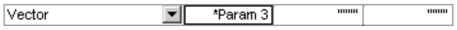
3 In the select Select Parameter dialog, select the parameter to be used in the data field.



Param 1
Param 2
Param 3

4 Click **OK**.

Once the parameter has been set, it is displayed in the corresponding cell. Notice that the parameter begins with an asterisk "*" to distinguish it from non-parameter values.



*Param 3	"""""""	"""""""

To clear a parameter

1 In the Macro tab, select the data field that contains the parameter to be removed.

2 Click **Edit** and choose **Clear Parameter**.

When you clear a parameter, the currently selected cell reverts from being a parameter to being a value. (By default, the new value is all quotes).

See Also

- To insert User-Defined Macro instructions (see page 67)

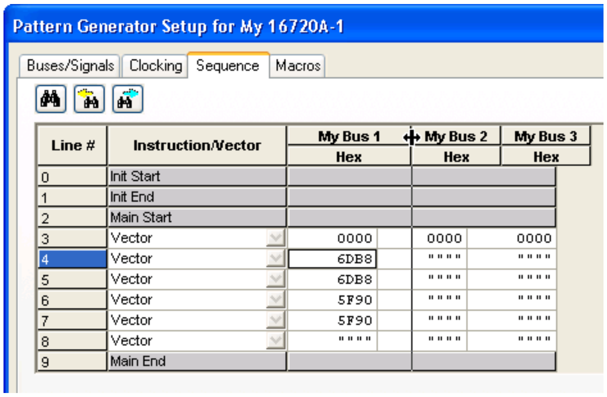
Setting Vector Sequence Display Options

These display options are available in both the Sequence and Macros tabs.

- "To adjust column widths" on page 90
- "To set the numeric base" on page 90
- "To change column colors" on page 91
- "To change column data alignment" on page 92
- "To change the font size" on page 93
- "To change instruction colors" on page 94
- "To specify a macro's color" on page 96

To adjust column widths

1 In the Sequence or Macros tab, drag a column divider to the desired column width.



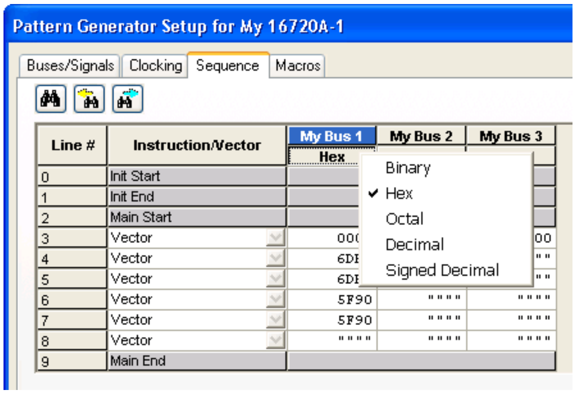
Line #	Instruction/Vector	My Bus 1	My Bus 2	My Bus 3
		Hex	Hex	Hex
0	Init Start			
1	Init End			
2	Main Start			
3	Vector	0000	0000	0000
4	Vector	6DB8	""""	""""
5	Vector	6DB8	""""	""""
6	Vector	5F90	""""	""""
7	Vector	5F90	""""	""""
8	Vector	""""	""""	""""
9	Main End			

To set the numeric base

In the Sequence and Macros tabs, the numeric base of a bus/signal is displayed in the column heading.

To change the numeric base setting:

1 In the Sequence or Macros tab, click the numeric base button in the column you want to change, and select the new numeric base.

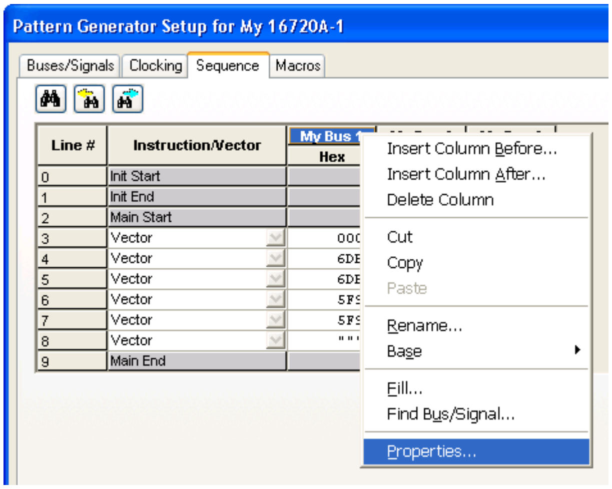


Line #	Instruction/Vector	My Bus 1 Hex	My Bus 2	My Bus 3
0	Init Start			
1	Init End			
2	Main Start			
3	Vector	000		00
4	Vector	6DE		" "
5	Vector	6DE		" "
6	Vector	5F90	" " " "	" " " "
7	Vector	5F90	" " " "	" " " "
8	Vector	" " " "	" " " "	" " " "
9	Main End			

To change column colors

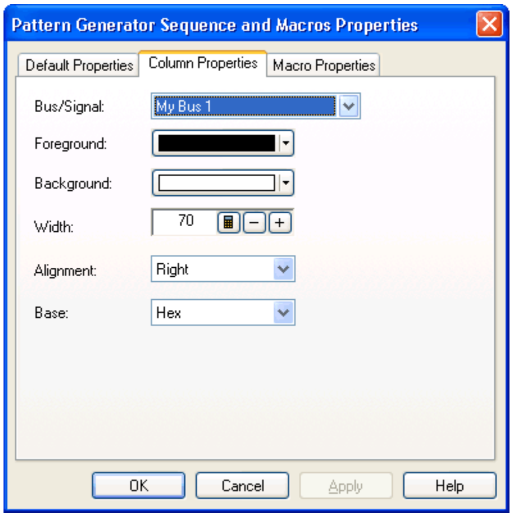
In the Sequence and Macros tabs, you can change the foreground and background colors used for vector sequence columns.

1 In the Sequence or Macros tab, right-click on a bus/signal column heading, and choose **Properties...**.



Line #	Instruction/Vector	My Bus 1 Hex
0	Init Start	
1	Init End	
2	Main Start	
3	Vector	000
4	Vector	6DE
5	Vector	6DE
6	Vector	5F9
7	Vector	5F9
8	Vector	" " "
9	Main End	

2 In the Column Properties tab of the Pattern Generator Properties dialog:



Bus/Signal: My Bus 1

Foreground:

Background:

Width: 70

Alignment: Right

Base: Hex

a Select the **Bus/Signal** column that the changes apply to.

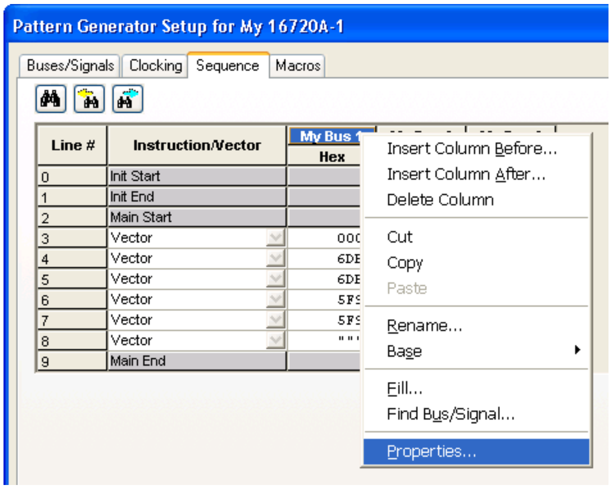
b Select the desired **Foreground** and **Background** colors.

c Click **OK**.

To change column data alignment

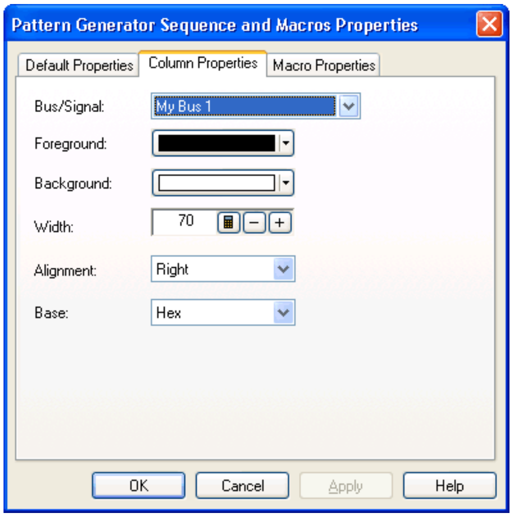
In the Sequence and Macros tabs, you can change the alignment of data in the vector sequence columns.

1 In the Sequence or Macros tab, right-click on a bus/signal column heading, and choose **Properties...**.



Line #	Instruction/Vector	My Bus 1 Hex
0	Init Start	
1	Init End	
2	Main Start	
3	Vector	000
4	Vector	6DE
5	Vector	6DE
6	Vector	5F
7	Vector	5F
8	Vector	"""
9	Main End	

2 In the Column Properties tab of the Pattern Generator Properties dialog:



Bus/Signal: My Bus 1
Foreground:
Background:
Width: 70
Alignment: Right
Base: Hex

a Select the **Bus/Signal** column that the changes apply to.

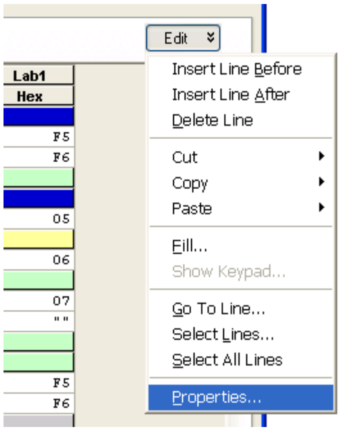
b Select the desired **Alignment** of the data in the column.

c Click **OK**.

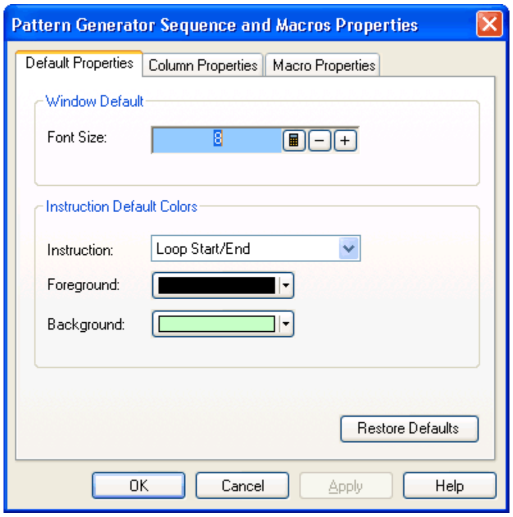
To change the font size

In the Sequence and Macros tabs, you can change the size of the text.

1 In the Sequence or Macros tab, click **Edit**, and choose **Properties...**.



2 In the Default Properties tab of the Pattern Generator Sequence and Macros Properties dialog:



Font Size: 8

Instruction: Loop Start/End

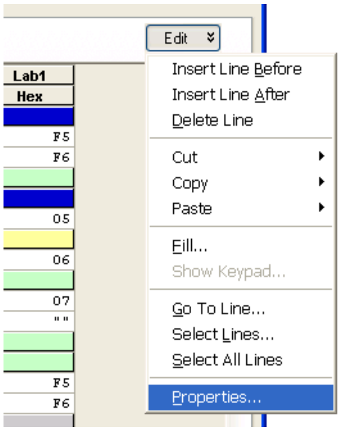
a Enter the **Font Size**.

b Click **OK**.

To change instruction colors

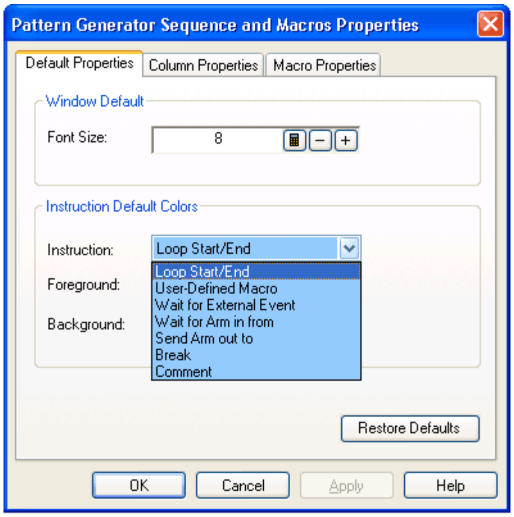
In the Sequence and Macros tabs, you can change the size of the text.

1 In the Sequence or Macros tab, click **Edit**, and choose **Properties...**.



2 In the Default Properties tab of the Pattern Generator Sequence and Macros Properties dialog:

a Select the type of **Instruction** whose color you want to change.



Pattern Generator Sequence and Macros Properties

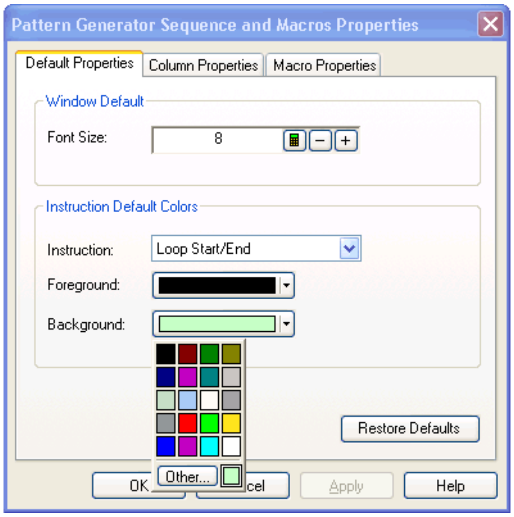
Window Default

Font Size: 8

Instruction Default Colors

Instruction: Loop Start/End

b Select the **Foreground** and **Background** colors.



Pattern Generator Sequence and Macros Properties

Window Default

Font Size: 8

Instruction Default Colors

Instruction: Loop Start/End

Foreground: ■

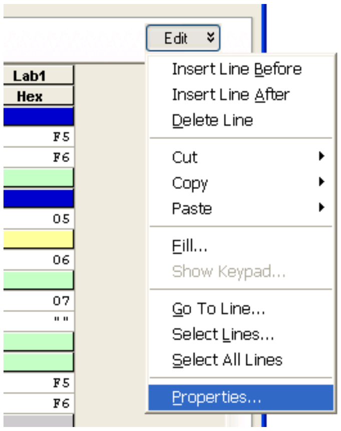
Background: 🟩

c Click **OK**.

To specify a macro's color

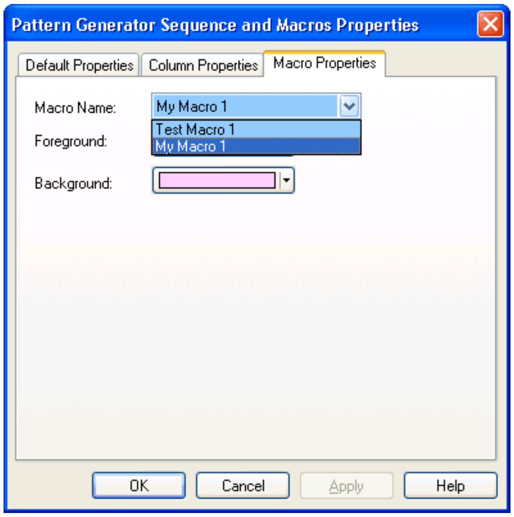
In the Sequence and Macros tabs, you can change the size of the text.

1 In the Sequence or Macros tab, click **Edit**, and choose **Properties...**.

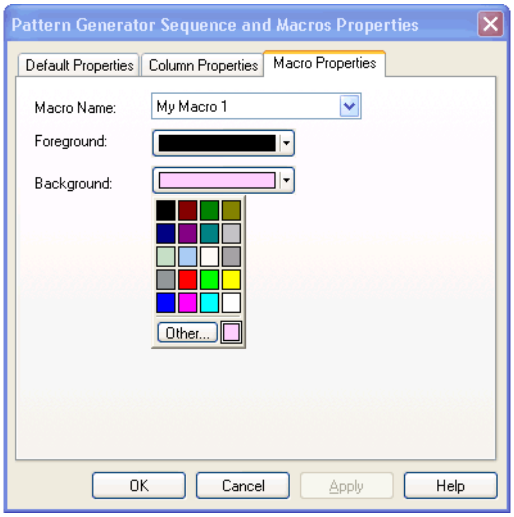


2 In the Macro Properties tab of the Pattern Generator Sequence and Macros Properties dialog:

a Select the name of the **Macro** whose color you want to change.



b Select the **Foreground** and **Background** colors.



Macro Name: My Macro 1

Foreground:

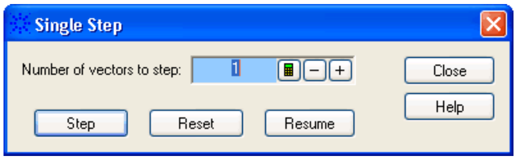
Background:

c Click **OK**.

6
Running and Stopping Pattern Generator Output

When controlling the pattern generator vector output, you can:

- **Run** - Causes the pattern generator to output vectors starting at the first line of the Initialization Sequence. "Run" executes the Initialization Sequence and Main Sequence one time only, then stops. Pattern generator outputs are help at the last vector until you reset or run again.
- **Run Rep** - Causes the pattern generator to output vectors starting at the first line of the Initialization Sequence. When all of the vectors in the Main Sequence have been output, the execution continues with the Main Start (not the Init Start). The Initialization Sequence is output once, and the Main Sequence is output repetitively, continuing until you stop.
- **Stop** - Causes the current execution to stop at the current vector. Pattern generator outputs are held at the current vector until you resume, step, reset, or run again.
- **Step...** - Opens the Single Step dialog.



Number of vectors to step: 1

In the Single Step dialog, you can:

- **Step** - Causes the pattern generator to output a specified number of vectors.
- **Reset** - Causes the pattern generator to reset to the first line of the Initialization Sequence. This only impacts "Step" and "Resume" because "Run" always begins at the first line of the Initialization Sequence.
- **Resume** - Causes the pattern generator to continue outputting vectors after the vector that was last output. If the system is stopped at the first vector, "Run" and "Resume" perform the same function; otherwise, they are different.

There are several ways to control the pattern generator vector output:

To:	Click in toolbar:	Click in Setup dialog:	Choose from Run/Stop>(PattGen Module) menu:
run once	⤓	**Run**	**Run**
run repetitive	↻	**Run Rep**	**Run Repetitive**
stop	■	**Stop**	**Stop**
step	≡	**Step** (in Single Step dialog)	**Step**
resume	⇣	**Resume** (in Single Step dialog)	**Resume**
reset	↰	**Reset** (in Single Step dialog)	**Reset**

7
Saving and Loading Pattern Generator Configurations

You can save pattern generator settings (including vectors) to ALA format or XML format configuration files. Later, you can restore your settings by opening the configuration file.

- "To open a configuration file" (in the online help)
- "To save a configuration file" (in the online help)

ALA Format Configuration Files

When you save configurations to ALA format files, all of the setup information is saved to a file with the .ala extension. Saved are: buses/signals, clocking options, the vector sequence, and macros.

XML Format Configuration Files

You can also save configurations to XML format files. Saved are: buses/signals, clocking options, and the compiled vector sequence. Loops, macros, and ditto characters in the vector sequence are not saved. (The compiled vector sequence is actually saved to a PattGen Binary (PGB) file that the XML format configuration file references.)

When loading a pattern generator configuration from an XML format file, it is just like opening a configuration and importing a PGB file; that is, if you want to edit the vector sequence, you must click **Enable Sequence Editing** in the Clocking tab of the Pattern Generator Setup dialog.

8
Exporting and Importing Vector Sequences

Exporting Vector Sequences to CSV Format Files

Exporting pattern generator vector sequences to CSV format files is just like exporting data from any of the logic analyzer display windows (Listing, Waveform, etc.). In fact, the same Export dialog is used; you just select the pattern generator as the source of the export.

However, there are some things to note:

- A pattern generator CSV file only describes the *compiled* data; it does not contain Loops, User-Defined Macros, or ditto characters.
- The first row exported has the bus/signal names. The second row has the pod index and channel numbers. Then, the initialization sequence rows are exported, followed by the main sequence rows.
- Delimiter and instruction rows begin with an asterisk (*).
- All vector data is exported as hexadecimal.

For example:

```
"MY BUS 1","MY BUS 2","MY SIGNAL 1","MY SIGNAL 2"
"Pod 1[7:0]","Pod 3[7:0], Pod 2[7:0]","Pod 4[0]","Pod 4[1]"
*Init Start,,,
FF,FFFF,1,1
EE,EEEE,0,0
*Wait for Arm in from "External Trigger",,,
DD,DDDD,1,1
CC,CCCC,0,0
*Init End,,,
*Main Start,,,
BB,BBBB,1,1
AA,AAAA,0,0
*Break,,,
99,9999,1,1
88,8888,0,0
*Wait for External Event A = (000 + 010 + 100 + 110),,,
77,7777,1,1
66,6666,0,0
*Send Arm out to "My 16910A-1" , "External Trigger",,,
55,5555,1,1
44,4444,0,0
*Main End,,,
```

See Also

- Pattern Generator CSV File Format (see page 106)

Importing Vector Sequences from CSV Format Files

You can now import pattern generator CSV files with Loops, User Macros, and Comment instructions. Pattern generator CSV files also contain bus/signal setup information; however, they do not contain clock source or output mode/frequency setup information.

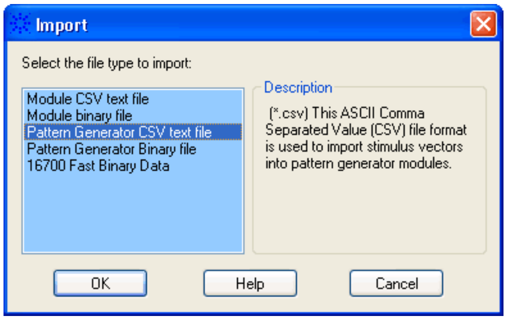
Because the CSV format is the same across the pattern generator and display windows (Waveform, Listing, etc.), you can convert captured data into vectors for the pattern generator by exporting data from a display window to a CSV format file and then importing the data back into the pattern generator module as vectors.

You can also convert old 16522A pattern generator ASCII format files to CSV format files (and XML format setup files) and import those CSV format vector sequences.

To import vector sequences from a comma-separated value (CSV) format file:

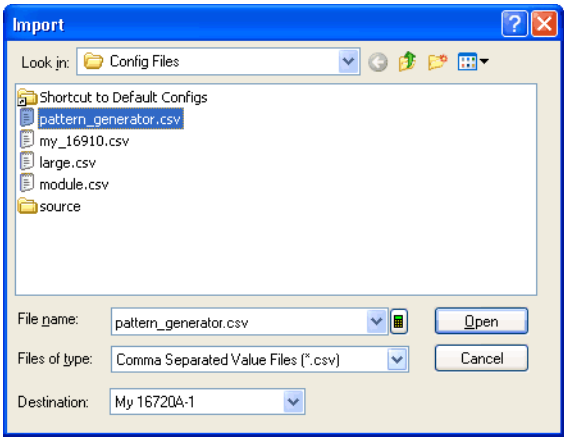
1 From the *Agilent Logic Analyzer* application's main menu, choose **File>Import...**.

2 In the Import dialog, select **CSV Data For Pattern Generator**, and click **OK**.


Import
Select the file type to import:
Module CSV text file
Module binary file
Pattern Generator CSV text file
Pattern Generator Binary file
16700 Fast Binary Data
Description
OK
Help
Cancel

(*.csv) This ASCII Comma Separated Value (CSV) file format is used to import stimulus vectors into pattern generator modules.

3 In the next Import dialog, select or enter the **File name** of the CSV file you want to import.



Look in: Config Files

File name: pattern_generator.csv

Files of type: Comma Separated Value Files (*.csv)

Destination: My 16720A-1

4 Select the **Destination** pattern generator module.

5 Click **Open**.

See Also

- Exporting Vector Sequences to CSV Format Files (see page 104)
- Converting Agilent 16522A ASCII Files (see page 132)
- Pattern Generator CSV File Format (see page 106)

Pattern Generator CSV File Format

The pattern generator supports the loading of Comma Seperated Value (CSV) formatted files. The CSV file format is compatible with other applications such as spreadsheets like Microsoft Excel and with other *Agilent Logic Analyzer* application windows like the Listing display. For example, you can export the data displayed in a Listing window as a CSV file and then, with a few small changes, import that captured data into the pattern generator.

You can now import pattern generator CSV files with Loops, User Macros, and Comment instructions.

When exporting pattern generator sequences to CSV format files, only the *compiled* data is output. The CSV file does not include Loops or User Macro instructions because these are *software* instructions and are not part of the compiled vector data. User Macros and Loops are expanded or *unrolled* by inserting the corresponding vectors directly into the CSV file. Also when exporting a CSV file, ditto characters are replaced with the actual corresponding values.

The syntax or file format of a pattern generator CSV is relatively straight forward. A CSV file is an ASCII file where each row in the file is a comma separated list of values. Individual rows are separated by newline characters. Values separated by commas can have zero or more whitespace characters between each value. Rows can also be blank or contain only whitespace. CSV files have essentially the same format as a file exported using Microsoft Excel using its CSV file format. They are also the same format generated by exporting data using the *Agilent Logic Analyzer* application in CSV format. All the specific details of the CSV syntax are discussed below.

NOTE

There must be a newline after the last line in a pattern generator CSV format file. That is, there must be one empty line at the end of the file. Otherwise, you get an "unexpected eof" error when trying to import the CSV file.

File Naming Convention

Pattern generator CSV files end with the extension *.csv. This is consistent with the *Agilent Logic Analyzer* application and Microsoft Excel.

CSV Example File

The following figure is an example pattern generator CSV file:

```
*Case Sensitive
"Lab6","Lab5","Lab4","Lab3","Lab2","Lab1"
"Pod 6[7:0]","Pod 5[7:0]","Pod 4[7:0]","Pod 3[7:0]","Pod 2[7:0]","Pod 1[
7:0]"
*Macro Start, "Test Macro 1", ("param1", "param2")
*Foregroundcolor "mintcream"
*Backgroundcolor "maroon"
*Comment Test Macro 1
"param1",2d,2e,2f,30,31
2d,"param2",2f,30,31,32
04,05,06,07,08,09
*Start Loop Repeat 5 times
00,01,02,03,04,05
01,02,03,04,05,06
*Start Loop Repeat 2 times
02,03,04,05,06,07
xx,xx,xx,xx,xx,xx
*End Loop
*End Loop
05,06,07,08,09,0a
06,07,08,09,0a,0b
*Macro End
*Init Start
*Comment Init Start
f0,f1,f2,f3,f4,f5
f1,f2,f3,f4,f5,f6
*Start Loop Repeat 5 times
*Comment Loop in Init Start
00,01,03,03,04,05
*Wait for External Event A = (101)
01,02,03,04,05,06
*Start Loop Repeat 2 times
02,03,04,05,06,07
XX,XX,XX,XX,XX,XX
*End Loop
*End Loop
f0,f1,f2,f3,f4,f5
f1,f2,f3,f4,f5,f6
*Init End
*Main Start
```

```
*Comment Main Start
f2,f3,f4,f5,f6,f7
f3,f4,f5,f6,f7,f8
*Wait for External Event B = (110)
f4,f5,f6,f7,f8,f9
*Start Loop Repeat 5 times
*Comment Loop in Main Start
02,03,04,05,06,07
03,04,05,06,07,08
*Start Loop Repeat 2 times
04,05,06,07,08,09
xx,xx,xx,xx,xx,xx
*End Loop
*End Loop
f5,f6,f7,f8,f9,fa
*Break
f6,f7,f8,f9,fa,fb
f7,f8,f9,fa,fb,fc
*User-Defined Macro "Test Macro 1" ("param1" = h0,"param2" = h1)
f8,f9,fa,fb,fc,fd
f9,fa,fb,fc,fd,fe
*Wait for Arm in from "External Trigger"
fa,fb,fc,fd,fe,ff
fb,fc,fd,fe,ff,00
*Send Arm out to
fc,fd,fe,ff,00,01
fd,fe,ff,00,01,02
*Main End
```

The file above demonstrates the main sections of a pattern generator CSV file:

- "*Case Sensitive Instruction" on page 109
- "Label Header" on page 109
- "Macro Definitions" on page 110
- "Initialization Sequence" on page 112
- "Main Sequence" on page 113

Within macro definitions, the initialization sequence, and the main sequence, there are:

- "Instructions" on page 113
- "Vectors" on page 115

Syntax Descriptions

In the following pattern generator CSV format syntax descriptions:

- Tokens (terminals) are shown as underlined like this: FOO
- Non-terminals are shown in brackets like this: <foo>
- A mandatory choice of options is shown in parenthesis with "|" separating each option like this: (A | B | C)
- An optional symbol is marked with a following "?" like this: <foo>?

- A repeat of zero or more times (optional or one or more) is indicated with a "*" like this: (<foo>)*
- A repeat of one or more times is indicated with a "+" like this: (<foo>)+
- String literals are shown in single or double quotes like this: "String literal" or 'String literal'

  Quoted strings can be used for Label names, Macro names and Macro Parameter names. Quoted strings may contain any printable 7-bit ASCII character except single-quote and double-quote. The backslash character is permitted and does not receive special treatment, such as an escape character.

See Also

- Exporting Vector Sequences to CSV Format Files (see page 104)
- Importing Vector Sequences from CSV Format Files (see page 105)

*Case Sensitive Instruction

The instruction, *Case Sensitive, must be the first line of the CSV file if it is present. It tells the pattern generator to keep lower-case or mixed-case label names imported from the CSV file. Without this instruction, all lower-case characters are translated to upper case.

Label Header

At the top of the file is the optional label header section. The label header section begins with a comma-separated list of label names. Label names must be unique, otherwise an error occurs. If a label name contains whitespace within the name, it must be enclosed in double-quotation marks. Otherwise, the quotation marks are optional. If the label header section is missing, the sequence data is applied to the existing labels within the pattern generator.

Underneath the label names comes the label definitions. The label definitions are order specific and respective to the order of the label names above them. Therefore, the first label definition is applied to the first label name and so on. The syntax of the label definitions is consistent with the XML syntax for labels:

```
<label_definition> = <pod_assignment> (, <pod_assignment>)*
<pod_assignment>   = Pod <integer> [ <channels> (, <channels>)* ]
<channels>         = (<integer> | <integer> : <integer>)

where: <integer> = an integer value
```

The first integer following the Pod keyword is the pod number. In this grammar, pods are numbered 1..N where N is the maximum number of pods in a module. In a 5-card 16720A pattern generator setup, this would be 1..30 in Full Channel Mode, 1..15 in Half Channel Mode. The pods are numbered from bottom to top, right to left as in the Bus/Signal Setup dialog. By using what are basically index numbers rather than the names they would have in the frame, the command in independent of how the frame is configured.

Here are some examples for three labels:

```
Example 1: "Pod 1 [7:0], Pod 2 [0:7]", "Pod 3 [1,2,3,4:6], Pod 4 [0:5,7:
6]", "Pod 5 [0]"
Example 2: Pod1[0], Pod1[1], Pod2[7:0]
Example 3: "Pod 1 [3,4,2:0]", "Pod 2 [0:2]", "Pod 3 [6:4], Pod 4 [1:2]"
```

Note that each label's pod definition specifies both position and order. Therefore, reordered channels can be specified. Order is always specified from most significant bit to least significant bit. Example 3 above specifies that the first three labels are assigned the following bits in MSB to LSB order:

- First label (5 bits total): 5 bits from Pod 1 in the order of: 3,4,2,1,0
- Second label (3 bits total): 3 bits from Pod 2 in the order of: 0,1,2
- Third label (5 bits total): 3 bits from Pod 3 in the order of: 6,5,4 and 2 bits from Pod 4 in the order of: 1,2

Bit reordering occurs during the load. That is, the data is reordered as it is imported. Therefore, if a label is reordered, the data appears reordered within the module after the load, but the label does not show up as reordered. Label reordering is a display-only concept, where import reordering affects what order the data is loaded.

Macro Definitions

Macros are defined within *Macro Start and *Macro End instructions. Macro definitions must appear after any label and pod definitions and before *Init Start. (The macro must be defined before it is used.)

```
*Macro Start , <macro_name> , ( <param_list> )

<macro_name> = The name of the macro being defined.  If there are
               embedded spaces in the name, the name must be quoted.

<param_list> = <param_name>* (, <param_name>)*

<param_name> = The name of the macro parameter.  If there are
               embedded spaces in the name, the name must be quoted.

<vectors_or_instructions>*
*Macro End
```

Macro Color Instructions

Within a macro definition, you can use the *Foregroundcolor and *Backgroundcolor instructions to set the macro's color properties. These colors appear in *User-Defined Macro instruction rows in the initialization or main sequence.

The *Foregroundcolor and *Backgroundcolor instructions can appear on any line between *Macro Start and *Macro End; however, for readability purposes, we suggest they be placed on the lines immediately after *Macro Start.

```
*Foregroundcolor <color_name>
*Backgroundcolor <color_name>

where:

<color_name> = A quoted string with the color to use.
```

You can choose from the following available colors:

⬛	Black	🟧	DarkOrange	🟩	PaleGreen	🟦	SlateGray
⬛	DimGray	🟫	BurlyWood	🟩	LightGreen	🟦	LightSteelBlue
⬛	Gray	⬜	AntiqueWhite	🟩	ForestGreen	🟦	CornflowerBlue
⬜	DarkGray	🟫	Tan	🟩	LimeGreen	🟦	RoyalBlue
⬜	Silver	⬜	NavajoWhite	🟩	DarkGreen	⬜	GhostWhite
⬜	LightGrey	⬜	BlanchedAlmond	🟩	Green, Lime	⬜	Lavender
⬜	Gainsboro	⬜	PapayaWhip	🟩	SeaGreen	🟦	MidnightBlue
⬜	WhiteSmoke	⬜	Moccasin	🟩	MediumSeaGreen	🟦	Navy
⬜	White	🟧	Orange	🟩	SpringGreen	🟦	DarkBlue
⬜	Snow	⬜	Wheat	⬜	MintCream	🟦	MediumBlue
🟫	RosyBrown	⬜	OldLace	🟩	MediumSpringGreen	🟦	Blue
🟥	LightCoral	⬜	FloralWhite	🟩	MediumAquamarine	🟪	SlateBlue
🟥	IndianRed	🟫	DarkGoldenrod	🟩	Aquamarine	🟦	DarkSlateBlue
🟥	Brown	🟨	Goldenrod	🟦	Turquoise	🟪	MediumSlateBlue
🟥	FireBrick	⬜	CornSilk	🟦	LightSeaGreen	🟪	MediumPurple
🟥	Maroon	🟨	Gold	🟦	MediumTurquoise	🟪	BlueViolet
🟥	DarkRed	⬜	LemonChiffon	⬜	Azure	🟪	Indigo
🟥	Red	🟨	Khaki	⬜	LightCyan	🟪	DarkOrchid
⬜	MistyRose	🟨	PaleGoldenrod	🟦	PaleTurquoise	🟪	DarkViolet
🟥	Salmon	🟨	DarkKhaki	⬛	DarkSlateGray	🟪	MediumOrchid

🟥	Tomato	⬜	Ivory	🟦	Teal	🟪	Thistle
🟧	DarkSalmon	⬜	Beige	🟦	DarkCyan	🟪	Plum
🟧	Coral	⬜	LightYellow	🟦	Aqua, Cyan	🟪	Violet
🟥	OrangeRed	⬜	LightGoldenrodYellow	🟦	DarkTurquoise	🟪	Purple
🟧	LightSalmon	🟫	Olive	🟦	CadetBlue	🟪	DarkMagenta
🟫	Sienna	🟨	Yellow	🟦	PowderBlue	🟪	Fuchsia, Magenta
⬜	Seashell	🟩	OliveDrab	🟦	LightBlue	🟪	Orchid
🟧	Chocolate	🟩	YellowGreen	🟦	DeepSkyBlue	🟪	MediumVioletRed
🟫	SaddleBrown	🟩	DarkOliveGreen	🟦	SkyBlue	🟪	DeepPink
🟧	SandyBrown	🟩	GreenYellow	🟦	LightSkyBlue	🟪	HotPink
⬜	PeachPuff	🟩	Chartreuse	🟦	SteelBlue	⬜	LavenderBlush
🟧	Peru	🟩	LawnGreen	⬜	AliceBlue	🟥	PaleVioletRed
⬜	Linen	⬜	Honeydew	🟦	DodgerBlue	🟥	Crimson
⬜	Bisque	🟩	DarkSeaGreen	🟦	LightSlateGray	⬜	Pink

Initialization Sequence

Following the Label Header is the Initialization Sequence. The Initialization Sequence must always be present — even if there are no Initialization Sequence vectors. The Initialization Sequence begins with the *Init Start instruction and ends with the *Init End instruction.

```
*Init Start
<vectors_or_instructions>*
*Init End
```

The Initialization Sequence can contain 0 or more vectors or instructions within its sequence. Trailing and leading whitespace is also allowable. Extra trailing commas are treated like whitespace too. This is important for the instruction rows because Microsoft Excel would generate a CSV file with an instruction row like this:

```
*Init Start,,
64,64,64
65,64,65
66,64,66
67,64,67
*Init End,,

*Main Start,,
68,64,68
69,64,69
```

Microsoft Excel tries to make sure the correct number of comma characters are always present based on the number of columns within the application. Because the above example includes three columns, there are 2 comma characters included on every row. Therefore, the parser ignores trailing commas for instruction rows.

Finally, if a value is specified that is larger than the corresponding label's width, then only the least significant bits of the value are applied to the label. Thus, if the value FE is applies to a label only 4 bits wide, then the label will get the value of E for that vector.

The *Init Start and *Init End instructions must be included exactly once.

Main Sequence

Following the Initialization Sequence is the Main Sequence. The Main Sequence begins with the *Main Start instruction and ends with the *Main End instruction.

```
*Main Start
<two_or_more_vectors_or_instructions>
*Main End
```

The Main Sequence can contain 2 or more vectors or instructions within its sequence. Like the Initialization Sequence, trailing and leading whitespace is also allowable, as well as trailing comma characters.

The *Main Start and *Main End instructions must be included exactly once. In addition, the *Main Start instruction must immediately follow the *Init End instruction.

Instructions

Pattern generator instructions begin with the asterisk (*) character. Instructions must be placed within the first column of a row; that is, they must not be preceded by any comma characters.

All the instructions described below are optional. The *Break and *Wait for External Event instructions can be inserted any number of times. The *Wait for Arm in from and *Send Arm out to instructions can only be inserted a maximum of one time.

***Break** To insert a hardware break instruction into a sequence:

```
*Break
```

***Comment** Comment instructions cause comment text to show up in sequences. Comment instructions are allowed anywhere a vector row is allowed. This is as opposed to regular comments, which begin with '//' and do not show up in any sequence.

```
*Comment <text>

<text> = Following the instruction and a space, this is all remaining
         characters on the line.
```

There is no explicit support for multi-line comments. To create a multi-line comment, use *Comment instructions on consecutive lines.

*Loop Start, *Loop End

Loop instructions may appear in macro definitions (*Macro Start ... *Macro End), the initialization sequence (*Init Start ... *Init End), or the main sequence (*Main Start ... *Main End). Loops may be arbitrarily nested, but Loops cannot span sequences; for example, a *Loop Start in the Init sequence cannot match a *Loop End in the Main sequence. Each *Loop Start must have a matching *Loop End.

```
*Start Loop Repeat <integer> ( "Time" | "Times" )

*End Loop
```

*Send Arm out to

The Send Arm out to instruction indicates to insert a hardware Send Arm out to instruction for signaling to generate an intermodule signal to the specified module(s) before continuing.

```
*Send Arm out to <module_list>?

   <module_list> = (<module_name> | 'External Trigger')
                   (, <module_name> | 'External Trigger')*

where:

<module_name> = The name of a module in the local frame.  If a
                module name contains whitespace within the name,
                then it must be enclosed in single or double
                quotation marks.  Otherwise, the quotation marks
                are optional.
```

*User-Defined Macro

To insert a macro into a sequence:

```
*User-Defined Macro , <macro_name> , ( <param_value_list> )

<macro_name> = The name of the macro to insert in the sequence.  If
               there are embedded spaces in the name, the name must
               be quoted.

<param_value_list> = <param_value>* (, <param_value>)*

<param_value> = <param_name> = <hex_value>
```

*Wait for Arm in from

The Wait for Arm in from instruction indicates to insert a hardware Wait for Arm in from instruction for waiting until the pattern generator receives an intermodule signal from the specified module(s) before continuing.

```
*Wait for Arm in from <module_list>?

   <module_list> = (<module_name> | 'External Trigger')
```

```
                    (, <module_name> | 'External Trigger')*

where:

<module_name> = The name of a module in the local frame.  If a
                module name contains whitespace within the name,
                then it must be enclosed in single or double
                quotation marks.  Otherwise, the quotation marks
                are optional.
```

***Wait for External Event**

A Wait for External Event instruction can optionally define the event that it is waiting on with the optional {} syntax at the end. If a Wait for External Event instruction does not define the event, then it will default to the preexisting defined value of the event. At least one Wait for External Event instruction should define the event being waited for. Otherwise, the event will default to the current value stored within the pattern generator model. If multiple Wait for External Event instructions try to define the same event (for example, two "*Wait for External Event A = ..." occur), then the last definition in the file will win, thus overriding any previous definitions.

```
*Wait for External Event (A | B | C | D) (= <event_desc>)?

   <event_desc> = ( (None | Any | <3dbn_list>) )
   <3dbn_list> = <wait_pattern> (+ <wait_pattern>)*

where:

<wait_pattern> = 3-digit binary number (for example, 001).
```

Vectors

Vectors for each defined label are separated by comma characters and zero or more whitespace.

Ditto characters are allowed in the form of "x" or "X" representing a single ditto character.

Importing PattGen Binary (PGB) Format Files

If you have a PattGen Binary (PGB) file from the 16700-series logic analysis system (and you have converted it to the 16900-series format (see page 140)), if you have created a PGB file in the 16900-series format (see page 117), or if you have a PGB file that was created when a pattern generator setup was saved to an XML format configuration file (see page 101), you can import that file to set up the 16720A pattern generator.

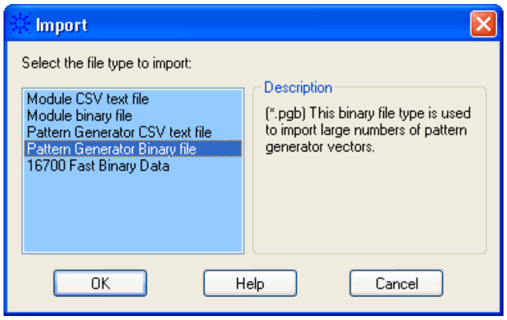
In general, the PattGen Binary file consists of a set of ASCII setup commands followed by a binary representation of the output vectors. PGB files do not contain loops or macros. PGB files can contain up to 8,388,608 vectors in full channel mode or 16,777,216 vectors in half channel mode.

CAUTION Importing a PattGen Binary (PGB) file causes all current setup and sequence information to be overwritten. Be sure to save the pattern generator configuration before you begin the import process.

To import vector sequences from a PattGen Binary (PGB) format file:

1 From the *Agilent Logic Analyzer* application's main menu, choose **File>Import...**.

2 In the Import dialog, select **Pattern Generator Binary**, and click **OK**.



Import

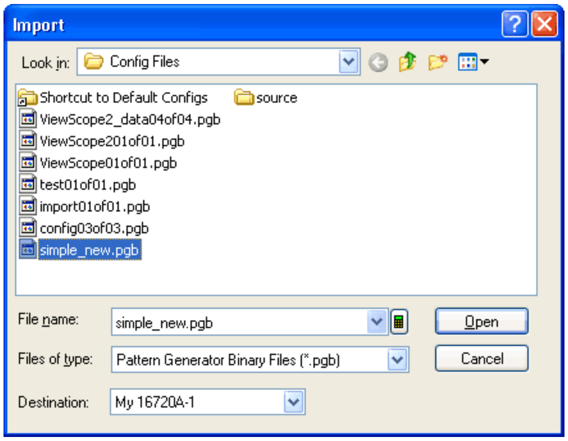
Select the file type to import:

Module CSV text file
Module binary file
Pattern Generator CSV text file
Pattern Generator Binary file
16700 Fast Binary Data

Description

(*.pgb) This binary file type is used to import large numbers of pattern generator vectors.

3 In the next Import dialog, select or enter the **File name** of the CSV file you want to import.



Look in: Config Files

Shortcut to Default Configs
source
ViewScope2_data04of04.pgb
ViewScope201of01.pgb
ViewScope01of01.pgb
test01of01.pgb
import01of01.pgb
config03of03.pgb
simple_new.pgb

File name: simple_new.pgb

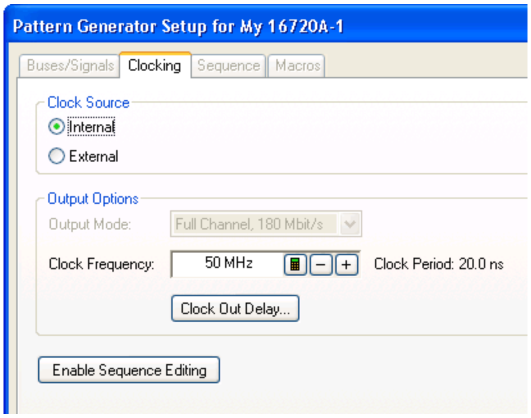
Files of type: Pattern Generator Binary Files (*.pgb)

Destination: My 16720A-1

4 Select the **Destination** pattern generator module.

5 Click **Open**.

6 If you want to edit the imported vector sequence, click **Enable Sequence Editing** in the Clocking tab of the Pattern Generator Setup dialog.



Clock Source: ● Internal ○ External

Output Mode: Full Channel, 180 Mbit/s

Clock Frequency: 50 MHz

Clock Period: 20.0 ns

Once data has been imported from PGB format files, you can save it to ALA format configuration files or export it to comma-separated value (CSV) format data files.

Creating a PattGen Binary File

The PattGen Binary format is primarily designed for deep (>1048576) vector sets; CSV format (which can be created by converting 16522A ASCII files (see page 132)) can be used for shorter sets.

PattGen Binary (PGB) format files can be created in the following ways:

- By converting PGB files from the 16700-series logic analysis system (see page 140).

- By saving pattern generator setups to XML format configuration files (see page 101) (the compiled vector sequence is saved to a PGB file that the XML format file references).
- By using programs for generating PGB format files.

  Deep vector sets can be generated from tools (such as Verilog simulators) and translated to PattGen Binary by third party tools or by translation utilities you have developed. The changes to vectors, bus/signal names, etc., should be made in the tool that originally generated the vectors (for example, Verilog).

Sample Code for Generating PattGen Binary Files

Visual C++ code for generating PattGen Binary files is installed with the *Agilent Logic Analyzer* application and is located in the directory:

```
<Drive letter>:\<Install directory>\Pattern Generator\
Visual C++ Examples\
```

For example:

```
C:\Program Files\Agilent Technologies\Logic Analyzer\Pattern Generator\
Visual C++ Examples\
```

There are two subdirectories, each containing a readme file, the program source code, the compiled executable, and a PattGen Binary file generated by the program:

- SimplePGB — A simple example of how to generate a PattGen Binary file.
- CommandLinePGB — A more extensive example with command line options.

PattGen Binary File Requirements and Precautions

- The file must contain only specified pattern generator commands (see page 119), and in the order and format shown in the example below.
- The file must be created in binary byte-stream format.
- Vector data must be entirely in binary.
- No pattern generator instructions are allowed in the data.
- No pattern generator macros are defined or invoked in the data.
- Bus/signal names (labels) may consist of adjacent bits or arbitrarily ordered bits.
- Each ASCII command must end with a line termination character (line feed "<LF>" or a carriage return and line feed "<CR><LF>")
- Comments can be included after the first command (PGBINARY).
- Comments begin with a slash '/' and terminate at the end of the line.

PattGen Binary Example

NOTE

In this example, only the ASCII pattern generator commands are shown. The binary vector data is not shown.

```
PGBINARY
MODE FULL
CLOCK INTERNAL 5e7   / 50 MHz.
MAIN 0
LABEL "My Bus 1" POD 1 [7:0]
LABEL "My Bus 2" POD 2 [7:0]
LABEL "My Bus 3" POD 3 [7:0]
LABEL "My Bus 4" POD 4 [7:0]
LABEL "My Bus 5" POD 5 [7:0]
LABEL "My Bus 6" POD 6 [7:0]
WIDTH 6
DEPTH 4096
BEGIN
(binary data ...)
```

NOTE The LABEL sequence specified in the 5th through 10th lines results in a specific bit assignment. Different POD assignments in those LABEL commands would give a different ordering to the bits.

Binary File Commands

The following ASCII commands are used in the PattGen Binary (PGB) format file to set up the pattern generator for the binary vector data at the end of the file. Commands cannot be abbreviated or truncated.

Commands

- PattGen Binary File Identifier (see page 120)
- CLOCK (see page 120)
- DELAY (see page 121)
- MODE (see page 121)
- LABEL (see page 122)
- ORDER (see page 124)
- WIDTH (see page 125)
- DEPTH (see page 125)
- BREAK (see page 126)
- EVENT (see page 127)
- WAITARM (see page 128)
- WAITEXT (see page 128)
- SENDARM (see page 129)
- MAIN (see page 129)
- DESTINATION (see page 130)
- BEGIN (see page 130)
- Binary Data (see page 131)

PattGen Binary File Identifier The first line of a PattGen Binary file must contain the text string "PGBINARY". The purpose of this string is to uniquely identify the file as a PattGen Binary file.

Example

```
PGBINARY
MODE FULL
CLOCK INTERNAL 5e7   / 50 MHz.
MAIN 0
LABEL "My Bus 1" POD 1[7:0]
LABEL "My Bus 2" POD 2[7:0]
LABEL "My Bus 3" POD 3[7:0]
LABEL "My Bus 4" POD 4[7:0]
LABEL "My Bus 5" POD 5[7:0]
LABEL "My Bus 6" POD 6[7:0]
WIDTH 6
DEPTH 4096
BEGIN
(binary data ...)
```

CLOCK Command The CLOCK command is optional. The existing clock scheme is used if nothing is specified. The CLOCK command specifies the clock source for the pattern generator.

Command Syntax

```
CLOCK [ EXTERNAL | INTERNAL <floating_point_number> ]
```

EXTERNAL = sets the clock mode to externally clocked.

INTERNAL = sets the clock mode to internally clocked.

<floating_point_number> = specifies the internal clock frequency. If the number is >= 1.0, Hz is assumed. If the number is < 1.0, a clock period is assumed.

NOTE

The maximum clock rate is limited by the channel mode. See MODE command.

Example

```
PGBINARY
MODE FULL
CLOCK INTERNAL 5e7   / 50 MHz.
MAIN 0
LABEL "My Bus 1" POD 1 [7:0]
LABEL "My Bus 2" POD 2 [7:0]
LABEL "My Bus 3" POD 3 [7:0]
LABEL "My Bus 4" POD 4 [7:0]
LABEL "My Bus 5" POD 5 [7:0]
LABEL "My Bus 6" POD 6 [7:0]
WIDTH 6
DEPTH 4096
BEGIN
(binary data ...)
```

DELAY Command The DELAY command is optional. The existing delay scheme is used if nothing is specified. The DELAY command specifies the clock out delay. The clock out delay setting allows positioning of the clock with respect to the data. Delay setting range is 0 - 14 with each increment delaying the clock approximately 500 ps per step.

NOTE

The delay setting that corresponds to zero is uncalibrated. You must measure it to determine the basic clock/data timing.

Command Syntax

```
DELAY <delay_arg>
```

<delay_arg> = an integer from 0 to 14 with each increment delaying the clock by approximately 500 ps.

Example

```
PGBINARY
MODE FULL
CLOCK INTERNAL 5e7   / 50 MHz.
DELAY 6
MAIN 0
LABEL "My Bus 1" POD 1 [7:0]
LABEL "My Bus 2" POD 2 [7:0]
LABEL "My Bus 3" POD 3 [7:0]
LABEL "My Bus 4" POD 4 [7:0]
LABEL "My Bus 5" POD 5 [7:0]
LABEL "My Bus 6" POD 6 [7:0]
WIDTH 6
DEPTH 4096
BEGIN
(binary data ...)
```

MODE Command The MODE command is optional. The existing mode scheme is used if nothing is specified. FULL channel output mode limits the data rate to a maximum of 180 MHz, but allows 48 channels per card. HALF channel output mode allows an output rate of up to 300 MHz, but limits the number of channels to 24 per card.

Command Syntax

```
MODE [ FULL | HALF ]
```

Example

```
PGBINARY
MODE FULL
CLOCK INTERNAL 5e7   / 50 MHz.
MAIN 0
LABEL "My Bus 1" POD 1 [7:0]
LABEL "My Bus 2" POD 2 [7:0]
LABEL "My Bus 3" POD 3 [7:0]
LABEL "My Bus 4" POD 4 [7:0]
LABEL "My Bus 5" POD 5 [7:0]
LABEL "My Bus 6" POD 6 [7:0]
WIDTH 6
DEPTH 4096
BEGIN
(binary data ...)
```

LABEL Command The LABEL command sets up the bus/signal names used by a PattGen Binary file.

The first <integer> following the POD keyword is the pod number. Pods are numbered from 1...N where N is the number pods in a card set. For example, in a 5-card pattern generator module, there would be 30 pods in Full channel mode or 15 pods in Half channel mode. Pods are numbered from the the bottom card to the top card and from the right pod to the left pod.

By using pod index numbers rather than slot and pod names, the LABEL command is independent of which slots a pattern generator module's cards are in.

Bit assignments are from the Most Significant Bit to the Least Significant Bit.

Command Syntax

```
LABEL <name_str> POD <integer> '[' <integer> | <integer> : <integer> {,
...} ']' {, POD ...}
```

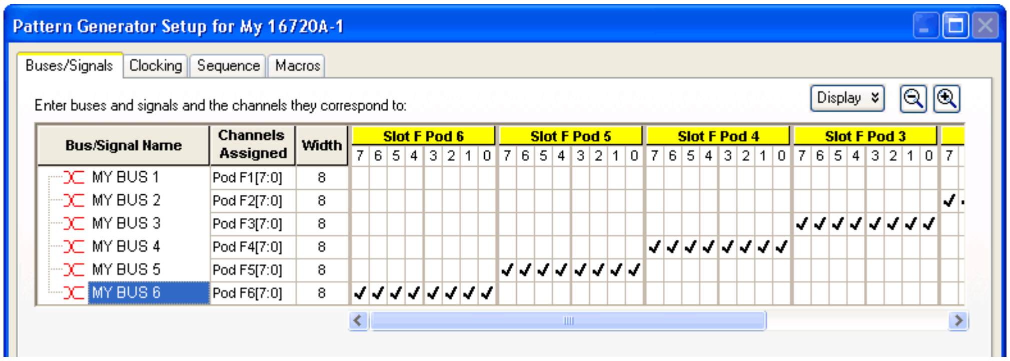
<name_str> = bus/signal name string with a maximum of 20 characters in length. The name string must begin with a letter (A-Z, a-z) or underscore (_) and can be followed by letters, underscores, or digits (0-9). A bus/signal name may not be a keyword (see page 123). If you wish to use a keyword as a bus/signal name, you must enclose it in quotes.

<integer> = pod index or bit number.

The pod and bit order affect the order in which the data is loaded. Pod/channel assignment is specified in MSB to LSB order and affects how the data is *loaded*. It does *not* reorder a label.

Example The graphics below are examples of how the binary files are imported into the pattern generator.

```
PGBINARY
MODE FULL
RMODE SINGLE
CLOCK INTERNAL 5e7   / 50 MHz.
MAIN 0
LABEL "My Bus 1" POD 1 [7:0]
LABEL "My Bus 2" POD 2 [7:0]
LABEL "My Bus 3" POD 3 [7:0]
LABEL "My Bus 4" POD 4 [7:0]
LABEL "My Bus 5" POD 5 [7:0]
LABEL "My Bus 6" POD 6 [7:0]
WIDTH 6
DEPTH 4096
BEGIN
(binary data ...)
```

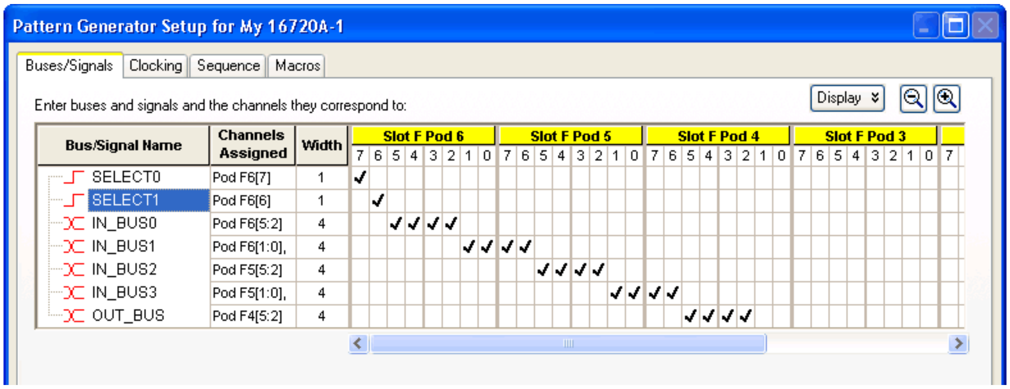


Enter buses and signals and the channels they correspond to:

Bus/Signal Name	Channels Assigned	Width	Slot F Pod 6								Slot F Pod 5								Slot F Pod 4								Slot F Pod 3								
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7
MY BUS 1	Pod F1[7:0]	8																																	
MY BUS 2	Pod F2[7:0]	8																																	✓
MY BUS 3	Pod F3[7:0]	8																									✓	✓	✓	✓	✓	✓	✓	✓	
MY BUS 4	Pod F4[7:0]	8																	✓	✓	✓	✓	✓	✓	✓	✓									
MY BUS 5	Pod F5[7:0]	8									✓	✓	✓	✓	✓	✓	✓	✓																	
MY BUS 6	Pod F6[7:0]	8	✓	✓	✓	✓	✓	✓	✓	✓																									

Example

```
PGBINARY
MODE FULL
RMODE SINGLE
CLOCK INTERNAL 5e7   / 50 MHz.
MAIN 0
LABEL Select0 POD 6 [7]
LABEL Select1 POD 6 [6]
LABEL IN_Bus0 POD 6 [5:2]
LABEL IN_Bus1 POD 6 [1,0], POD 5 [7:6]
LABEL IN_Bus2 POD 5 [5:2]
LABEL IN_Bus3 POD 5 [1,0], POD 4 [7:6]
LABEL OUT_Bus POD 4 [5:2]
WIDTH 7
DEPTH 4096
BEGIN
(binary data ...)
```



Enter buses and signals and the channels they correspond to:

Bus/Signal Name	Channels Assigned	Width	Slot F Pod 6								Slot F Pod 5								Slot F Pod 4								Slot F Pod 3								
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7
SELECT0	Pod F6[7]	1	✓																																
SELECT1	Pod F6[6]	1		✓																															
IN_BUS0	Pod F6[5:2]	4			✓	✓	✓	✓																											
IN_BUS1	Pod F6[1:0],	4							✓	✓	✓	✓																							
IN_BUS2	Pod F5[5:2]	4											✓	✓	✓	✓																			
IN_BUS3	Pod F5[1:0],	4															✓	✓	✓	✓															
OUT_BUS	Pod F4[5:2]	4																			✓	✓	✓	✓											

Keywords The following are keywords in the PattGen Binary file and may not be used as labels.

- A
- B
- BEGIN
- BREAK

- C
- CLOCK
- D
- DELAY
- DEPTH
- EVENT
- EXTERNAL
- FULL
- HALF
- INTERNAL
- LABEL
- MAIN
- MODE
- NONE
- ORDER
- PGBINARY
- POD
- REPETITIVE
- RMODE
- SENDARM
- SINGLE
- WAITARM
- WAITEXT
- WIDTH

ORDER Command The Order command provides an optional means for specifying the order that data for labels will appear. If the order is not specified it is assumed to be the order in which labels were defined. If the ORDER command contains only a subset of the labels that have been defined, then the subsequent BEGIN command refers only to the labels named in the ORDER command.

Command Syntax

```
ORDER { <name_str> ... }
```

<name_str> = label string a maximum of 20 characters in length.

Example

```
PGBINARY
MODE FULL
RMODE SINGLE
CLOCK INTERNAL 5e7   // 50 MHz
MAIN 0
```

```
LABEL "My Bus 1" POD 1 [7:0]
LABEL "My Bus 2" POD 2 [7:0]
LABEL "My Bus 3" POD 3 [7:0]
LABEL "My Bus 4" POD 4 [7:0]
LABEL "My Bus 5" POD 5 [7:0]
LABEL "My Bus 6" POD 6 [7:0]
ORDER "My Bus 6" "My Bus 5" "My Bus 4" "My Bus 3" "My Bus 2" "My Bus 1"
WIDTH 6
DEPTH 4096
BEGIN
```

WIDTH Command The WIDTH command defines how wide, in number of bytes, the binary data will be. The width is determined by summing the size, rounded up to full bytes, of every label that is defined or named in an ORDER command. The total number of data bytes in the file must be (DEPTH * WIDTH). WIDTH must be defined before the binary data part begins. The binary data part also contains the width. This redundant information is used as one verification of the binary data.

Command Syntax

```
WIDTH <integer>
```

<integer> = number of bytes wide the binary data will be.

Example

```
PGBINARY
MODE FULL
RMODE SINGLE
CLOCK INTERNAL 5e7   / 50 MHz.
MAIN 0
LABEL "My Bus 1" POD 1 [7:0]
LABEL "My Bus 2" POD 2 [7:0]
LABEL "My Bus 3" POD 3 [7:0]
LABEL "My Bus 4" POD 4 [7:0]
LABEL "My Bus 5" POD 5 [7:0]
LABEL "My Bus 6" POD 6 [7:0]
WIDTH 6
DEPTH 4096
BEGIN
(binary data ...)
```

DEPTH Command The DEPTH command defines how may vectors will appear in the binary data. The total number of bytes in the binary data part of the file must be (DEPTH * WIDTH). DEPTH must be defined before the binary data part begins. The binary data part also contains the depth. This redundant information is used as one verification of the binary data.

The DEPTH has the following restrictions based on the MODE (FULL or HALF) that the instrument will be in. These restrictions only apply if the DESTINATION (see page 130) is HARDWARE_ONLY or HARDWARE_AND_SETUP; otherwise, normal sequence rules from the user interface apply.

FULL Channel Mode

- INIT part must be an even number of vectors.

- MAIN part must be an even number of vectors.
- Total depth can be no more than 8,388,608.
- Total depth can be no less than 4, 096.

HALF Channel Mode

- INIT part must be a multiple of 4 vectors.
- MAIN part must be a multiple of 4 vectors.
- Total depth can be no more than 16,777,216.
- Total depth can be no less than 4,096.

Command Syntax

```
DEPTH <integer>
```

<integer> = number of vectors in the binary data.

Example

```
PGBINARY
MODE FULL
RMODE SINGLE
CLOCK INTERNAL 5e7   / 50 MHz.
MAIN 0
LABEL "My Bus 1" POD 1 [7:0]
LABEL "My Bus 2" POD 2 [7:0]
LABEL "My Bus 3" POD 3 [7:0]
LABEL "My Bus 4" POD 4 [7:0]
LABEL "My Bus 5" POD 5 [7:0]
LABEL "My Bus 6" POD 6 [7:0]
WIDTH 6
DEPTH 4096
BEGIN
(binary data ...)
```

BREAK Command The BREAK command places a hardware BREAK instruction on a vector number <integer>. Vectors are numbered from 0 to DEPTH- 1.

Command Syntax

```
BREAK <integer>
```

<integer> = the vector where a hardware BREAK instruction is to be placed.

Example

```
PGBINARY
MODE FULL
RMODE SINGLE
CLOCK INTERNAL 5e7   / 50 MHz.
MAIN 0
LABEL "My Bus 1" POD 1 [7:0]
LABEL "My Bus 2" POD 2 [7:0]
LABEL "My Bus 3" POD 3 [7:0]
LABEL "My Bus 4" POD 4 [7:0]
LABEL "My Bus 5" POD 5 [7:0]
LABEL "My Bus 6" POD 6 [7:0]
BREAK 100
WIDTH 6
DEPTH 4096
```

```
BEGIN
(binary data ...)
```

NOTE A BREAK instruction can go on any vector except the first vector or the last vector. The last vector instruction is used to implement single and repetitive run modes and is not available for other instructions. Attempting to place an instruction on or beyond the last vector is an error.

EVENT Command The EVENT command sets a pattern into one of the four external event registers. These event registers can be specified in WAITEXT instructions. If the pattern is NONE then a wait on this event will always wait. If the pattern is an integer, its value is interpreted according to the following table.

integer	External Event Input Values		
(hex)	WAITEXT_2	WAITEXT_1	WAITEXT_0
#H01	0	0	0
#H02	0	0	1
#H04	0	1	0
#H08	0	1	1
#H10	1	0	0
#H20	1	0	1
#H40	1	1	0
#H80	1	1	1

Values can be ORed together; for example, a value of #H22 would indicate patterns of (101 + 001).

Command Syntax

```
EVENT [ A | B | C | D ] [ NONE | <integer> ]
```

A, B, C, D = four external wait event registers.

NONE - value that indicates "always wait".

<integer> = the event pattern from the table above.

Example

```
PGBINARY
MODE FULL
RMODE SINGLE
MAIN 0
LABEL "My Bus 1" POD 1 [7:0]
LABEL "My Bus 2" POD 2 [7:0]
LABEL "My Bus 3" POD 3 [7:0]
LABEL "My Bus 4" POD 4 [7:0]
LABEL "My Bus 5" POD 5 [7:0]
LABEL "My Bus 6" POD 6 [7:0]
EVENT A #H01
WAITEXT A 100
WIDTH 6
DEPTH 4096
BEGIN
(binary data ...)
```

WAITARM Command The WAITARM command places a hardware instruction on vector number <integer>. Vectors are numbered from 0 to DEPTH-1. There can only be one WAITARM instruction in a configuration.

Command Syntax

```
WAITARM { [ <module_name> | "External Trigger" ] {, ... } } <integer>
```

<integer> = the vector where a hardware WAITARM instruction is to be placed.

Example

```
PGBINARY
MODE FULL
RMODE SINGLE
CLOCK INTERNAL 5e7   / 50 MHz.
MAIN 0
LABEL "My Bus 1" POD 1 [7:0]
LABEL "My Bus 2" POD 2 [7:0]
LABEL "My Bus 3" POD 3 [7:0]
LABEL "My Bus 4" POD 4 [7:0]
LABEL "My Bus 5" POD 5 [7:0]
LABEL "My Bus 6" POD 6 [7:0]
WAITARM "EXTERNAL TRIGGER" 200
WIDTH 6
DEPTH 4096
BEGIN
(binary data ...)
```

NOTE

A WAITARM instruction can go on any vector except the first vector or the last vector. The last vector instruction is used to implement single and repetitive run modes and is not available for other instructions. Attempting to place an instruction on or beyond the last vector is an error.

WAITEXT Command The WAITEXT command places a hardware instruction on vector number <integer>. Vectors are numbered from 0 to DEPTH-1.

Command Syntax

```
WAITEXT [ A | B | C | D ] <integer>
```

A, B, C, D = external wait events.

<integer> = the vector where a hardware WAITEXT instruction is to be placed.

Example

```
PGBINARY
MODE FULL
RMODE SINGLE
CLOCK INTERNAL 5e7   / 50 MHz.
MAIN 0
LABEL "My Bus 1" POD 1 [7:0]
LABEL "My Bus 2" POD 2 [7:0]
LABEL "My Bus 3" POD 3 [7:0]
LABEL "My Bus 4" POD 4 [7:0]
LABEL "My Bus 5" POD 5 [7:0]
LABEL "My Bus 6" POD 6 [7:0]
EVENT A #H01
WAITEXT A 100
```

```
WIDTH 6
DEPTH 4096
BEGIN
(binary data ...)
```

NOTE

A WAITEXT instruction can go on any vector except the first vector or the last vector. The last vector instruction is used to implement single and repetitive run modes and is not available for other instructions. Attempting to place an instruction on or beyond the last vector is an error.

SENDARM Command The SENDARM command places a hardware SENDARM instruction on vector number <integer>. Vectors are numbered from 0 to DEPTH-1. There can only be one SENDARM instruction in a configuration.

Command Syntax

```
SENDARM <integer>
```

Example

```
PGBINARY
MODE FULL
RMODE SINGLE
CLOCK INTERNAL 5e7   / 50 MHz.
MAIN 0
LABEL "My Bus 1" POD 1 [7:0]
LABEL "My Bus 2" POD 2 [7:0]
LABEL "My Bus 3" POD 3 [7:0]
LABEL "My Bus 4" POD 4 [7:0]
LABEL "My Bus 5" POD 5 [7:0]
LABEL "My Bus 6" POD 6 [7:0]
SENDARM 100
WIDTH 6
DEPTH 4096
BEGIN
(binary data ...)
```

NOTE

A SENDARM instruction can go on any vector except the first vector or the last vector. The last vector instruction is used to implement single and repetitive run modes and is not available for other instructions. Attempting to place an instruction on or beyond the last vector is an error.

MAIN Command The MAIN command gives the vector number where the MAIN part of the vector set begins. The vectors prior to the MAIN vector comprise the INIT part. This is defaulted to zero (no INIT part). Vectors are numbered from 0 to DEPTH-1.

Command Syntax

```
MAIN <integer>
```

<integer> = the vector number where the MAIN part of the vector set begins.

Example

```
PGBINARY
MODE FULL
```

```
RMODE SINGLE
CLOCK INTERNAL 5e7   / 50 MHz.
MAIN 0
LABEL "My Bus 1" POD 1 [7:0]
LABEL "My Bus 2" POD 2 [7:0]
LABEL "My Bus 3" POD 3 [7:0]
LABEL "My Bus 4" POD 4 [7:0]
LABEL "My Bus 5" POD 5 [7:0]
LABEL "My Bus 6" POD 6 [7:0]
WIDTH 6
DEPTH 4096
BEGIN
(binary data ...)
```

DESTINATION Command The DESTINATION command specifies the destination of the PGB import process.

Command Syntax

```
DESTINATION [ HARDWARE_ONLY | HARDWARE_AND_SETUP | SETUP_ONLY ]
```

HARDWARE_ONLY = the PGB file is loaded into the hardware. In this case, the pattern generator is set up to run without doing a compile and load beforehand.

HARDWARE_AND_SETUP = the PGB file is loaded into both the hardware and the Sequence tab of the Pattern Generator Setup dialog.

SETUP_ONLY = the PGB file is only loaded into the Sequence tab of the Pattern Generator Setup dialog.

Default is HARDWARE_ONLY.

Example

```
PGBINARY
MODE FULL
RMODE SINGLE
CLOCK INTERNAL 5e7   / 50 MHz.
MAIN 0
LABEL "My Bus 1" POD 1 [7:0]
LABEL "My Bus 2" POD 2 [7:0]
LABEL "My Bus 3" POD 3 [7:0]
LABEL "My Bus 4" POD 4 [7:0]
LABEL "My Bus 5" POD 5 [7:0]
LABEL "My Bus 6" POD 6 [7:0]
EVENT A #H01
WAITEXT A 100
WIDTH 6
DEPTH 4096
DESTINATION HARDWARE_AND_SETUP
BEGIN
(binary data ...)
```

BEGIN Command The BEGIN command is the last command in the ASCII part of the file. It is terminated with a newline or a comment like all other commands. The binary data immediately follows the newline.

Command Syntax

```
BEGIN <newline> <binary data>
```

Example

```
PGBINARY
MODE FULL
RMODE SINGLE
CLOCK INTERNAL 5e7   / 50 MHz.
MAIN 0
LABEL "My Bus 1" POD 1 [7:0]
LABEL "My Bus 2" POD 2 [7:0]
LABEL "My Bus 3" POD 3 [7:0]
LABEL "My Bus 4" POD 4 [7:0]
LABEL "My Bus 5" POD 5 [7:0]
LABEL "My Bus 6" POD 6 [7:0]
WIDTH 6
DEPTH 4096
BEGIN
(binary data ...)
```

Binary Data The Binary Data begins with a simple eight byte header that gives the DEPTH and WIDTH in binary. These values must match the DEPTH and WIDTH given as ASCII commands earlier in the file. These values must be written as 32-bit integers in big-endian order, in other words, the first byte is the most significant byte. This is followed by the binary data itself.

- 4 bytes = depth (number of vectors).
- 4 bytes = width (number of bytes per vector).
- (WIDTH * DEPTH) bytes = the binary data.

Each vector is comprised of one value for each LABEL that has been defined or named in an ORDER command. The number of bytes for each label is the lowest possible integer number of bytes given the bit width of the label. For example, a 17-bit label requires 3 bytes (24 bits), and a 16-bit label requires 2 bytes (16 bits).

The bytes are taken to be in big-endian order. If a label is comprised of a smaller number of bits than are written for that label (for example, a 17-bit label that receives 24-bit data), the least significant bits are used and the excess most significant bits are discarded. This number must match the number of bytes required for the labels that have been defined or named in an ORDER command.

The remaining length of the file after the newline following "BEGIN" must be (8 + DEPTH * WIDTH).

If a LABEL is defined for pods that are not in the current hardware (for example, POD 7 for a 1-card module), the configuration is truncated to what fits and a warning is emitted.

Converting Agilent 16522A ASCII Files

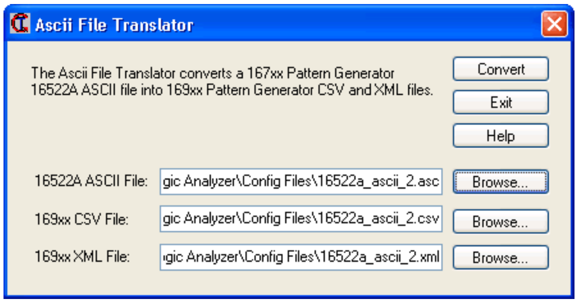
If you have an old 16522A pattern generator ASCII format file (or if you create an ASCII file (see page 133) in that format), you can convert it into two files that can be used with the 16720A pattern generator:

- A CSV format file containing the vector sequence data.
- An XML format file containing pattern generator clock source and output mode/frequency setup information.

You can create an ASCII text file and import it as a complete pattern generator program. In general, the ASCII file consists of a block of setup information, a block of label and channel information, and a block of pattern generator vector data. The file must be saved in ASCII format and organized as shown in step 1 of the procedure below.

1 From the Windows Start menu, choose **Start>All Programs>Agilent Logic Analyzer>Utilities>Pattern Generator>167xx Pattern Generator Ascii Translator**.

2 In the Ascii File Translator dialog, enter the name of the **16522A ASCII File** you want to translate.



The Ascii File Translator converts a 167xx Pattern Generator 16522A ASCII file into 169xx Pattern Generator CSV and XML files.

16522A ASCII File: gic Analyzer\Config Files\16522a_ascii_2.asc

169xx CSV File: gic Analyzer\Config Files\16522a_ascii_2.csv

169xx XML File: ıgic Analyzer\Config Files\16522a_ascii_2.xml

3 Enter the name for the **169xx CSV File** part of the translation output.

4 Enter the name for the **169xx XML File** part of the translation output.

5 Click **Convert**.

6 After the conversion has been performed, click **Exit** to close the Ascii File Translator dialog.

7 Open the XML format configuration file as described in "To open a configuration file" (in the online help).

8 Import the CSV format vector sequence file as described in Importing Vector Sequences from CSV Format Files (see page 105).

Creating an ASCII File

You can create an ASCII file using any Windows, MS-DOS, or UNIX text editor. An ASCII file consists of a file identifier and three blocks of information. Each block must follow the specified order.

- **ASCII 000000** - required file identifier ("ASCII" followed by 5 spaces and 6 zeros). **ASCDown** - optional. Retained for backwards compatibility.
- 1st block (optional)
  - **FORMat** - clock, channel mode, and delay information.
- 2nd block
  - **LABel** - names and number of channels.
- 3rd block
  - **VECTor** - vector data and Repeat indicators.

File Requirements and Precautions

- The file must contain only specified pattern generator commands (see page 134), and in the order and format shown in the example below.
- The file must be saved in "ASCII" or "text only" format.
- Vector data is assumed to be entirely hexadecimal base.
- No pattern generator instructions are allowed in the data.
- No pattern generator macros are defined or invoked in the data.
- All labels consist of adjacent bits.
- The file must end with a line termination character (line feed "<lf>" or a carriage return and line feed "<CR><lf>").
- Comments can be included after the first line (ASCII 000000). Comments begin with a slash '/' and terminate at the end of the line.

ASCII File Example

NOTE

In this example, the underlined links are added for documentation purposes only, and would NOT be part of an actual disk file's text. Line feeds "<lf>" are shown for example purposes only, and depending on the computer and text editor used, could actually be a carriage return followed by a line feed "<CR><lf>".

```
ASCII     000000 (see page 134) <lf>
/
/ This is a test of the ascii file
/
ASCDown (see page 135)<lf>
FORMat:MODE FULL (see page 135)<lf>
FORMat:CLOCk INTernal, 10E-9 (see page 136)<lf>
LABel LAB1,8 (see page 137)<lf>
LABel DATA,8<lf>
LABel TEST,9<lf>
LABel CLK,3<lf>
LABel BIG,12<lf>
/*
/* This is the beginning of the vector part
/*
VECTor (see page 138)<lf>
12 34 056 7 89A<lf>         / Some vectors
0 22 007 0 FFF<lf>
A0 33 000 1 111<lf>         /* Some more vectors */
*M<lf>
92 6F 000 1 FF0<lf>
CA CA 000 1 00F<lf>         // Even more vectors
*R 3<lf>
00 10 011 0 ABC<lf>
```

NOTE

The *LABel* sequence specified in the 8th through 12th lines results in a specific bit assignment. A different ordering of the LABel commands would give a different ordering to the bits.

ASCII File Commands

The following commands are used in the 16522A ASCII format file to set up the pattern generator and define the vector sequence data. Commands are not case sensitive. Lowercase letters in an illustrated command simply show the long and short form difference.

The uppercase letters of the command show the mandatory portions of each command.

Commands

- ASCII Disk File Identifier (see page 134)
- ASCDown Command (see page 135)
- FORMat Commands
  - MODe (see page 135)
  - CLOCk (see page 136)
  - DELay (see page 136)
- LABel Command (see page 137)
- VECTor Command (see page 138)

ASCII Disk File Identifier The first line of a disk file must contain the text string "**ASCII 000000**". It consists of the text "**ASCII**" followed by 5 blanks, then 6 zeros. The purpose of this string is to uniquely identify the file as an ASCII disk file.

Example

```
ASCII     000000
ASCDOWN
FORMAT: MODE FULL
LABEL LAB,8
LABEL DATA,8
LABEL "TEST",9
LABEL CLK,3
LABEL BIG,12
VECTOR
12 34 56 7 89A
0 22 7 0 FFF
A0 33 00 1 111
*M
92 6F 00 1 FF0
CA CA 00 1 00F
00 10 11 0 ABC
```

ASCDown Command The ASCDown command was formerly used to signal the start of an ASCII file load. It causes the current pattern generator label and sequence structures to be cleared and reset to a default state. The Agilent 16720A pattern generator now does this automatically.

Example

```
ASCII     000000
ASCDOWN
FORMAT: MODE FULL
LABEL LAB,8
LABEL DATA,8
LABEL "TEST",9
LABEL CLK,3
LABEL BIG,12
VECTOR
12 34 56 7 89A
0 22 7 0 FFF
A0 33 00 1 111
*M
92 6F 00 1 FF0
CA CA 00 1 00F
00 10 11 0 ABC
```

FORMat:MODe Command The FORMat:MODe command is optional. The existing mode scheme is used if nothing is specified. FULL channel output mode limits the data rate to a maximum of 180 MHz, but allows 48 channels per card. HALF channel output mode allows an output rate of greater than 180 MHz, but limits the number of channels to 24 per card.

Command Syntax

```
FORMat:MODe [FULL|HALF]
```

Example

```
ASCII     000000
ASCDOWN
FORMAT: MODE FULL
LABEL LAB,8
LABEL DATA,8
LABEL "TEST",9
LABEL CLK,3
LABEL BIG,12
VECTOR
12 34 56 7 89A
0 22 7 0 FFF
A0 33 00 1 111
*M
92 6F 00 1 FF0
```

```
CA CA 00 1 00F
00 10 11 0 ABC
```

FORMat:CLOCk Command The FORMat:CLOCk command is optional. The existing clock scheme is used if nothing is specified. The CLOCk command specifies the clock source for the pattern generator.

Command Syntax

```
FORMat:CLOCk INTernal, <clk_period>
```

<clk_period> = a real number value that corresponds to the interface selectable clock period values (example = 5E-9).

```
FORMat:CLOCk EXTernal, [LEFifty|GTFifty|GTONe]
```

[LEFifty] = Less than or equal to 50 MHz.

[GTFifty] = Greater than 50 MHz and less than or equal to 180 MHz.

[GTONe] = Greater than 180 MHz.

NOTE

The maximum clock rate is limited by the channel mode. See FORMat:MODe (see page 135) command.

Example

```
ASCII     000000
ASCDOWN
FORMAT: MODE FULL
FORMAT: CLOCK INTERNAL, 10E-9
LABEL LAB,8
LABEL DATA,8
LABEL "TEST",9
LABEL CLK,3
LABEL BIG,12
VECTOR
12 34 56 7 89A
0 22 7 0 FFF
A0 33 00 1 111
*M
92 6F 00 1 FF0
CA CA 00 1 00F
00 10 11 0 ABC
```

FORMat:DELay Command The FORMat:DELay command is optional. The existing delay scheme is used if nothing is specified. The DELay command specifies the clock out delay. The clock out delay setting allows positioning of the clock with respect to the data. Delay setting range is 0 - 14 with each increment delaying the clock approximately 500 ps per step.

NOTE

The delay setting that corresponds to zero is uncalibrated. You must measure it to determine the basic clock/data timing.

Command Syntax

```
FORMat:DELay <delay_arg>
```

<delay_arg> = an integer from 0 to 14 with each increment delaying the clock by approximately 500 ps.

Example

```
ASCII     000000
ASCDOWN
FORMAT: MODE FULL
FORMAT: CLOCK INTERNAL, 10E-9
FORMAT: DELAY 2
LABEL LAB,8
LABEL DATA,8
LABEL "TEST",9
LABEL CLK,3
LABEL BIG,12
VECTOR
12 34 56 7 89A
0 22 7 0 FFF
A0 33 00 1 111
*M
92 6F 00 1 FF0
CA CA 00 1 00F
00 10 11 0 ABC
```

LABel Command The LABel command is a special means of specifying labels for use by an ASCII file. The label bits are assigned from most to least significant bits across the output pods. You must specify the label string (quotation marks on the string are optional) and the width of the field. The label base defaults to hexadecimal. There are a maximum of 126 labels. No label may be more than 32 bits wide. If a label is too wide (too many bits) for the remaining unused pattern generator bits, it will be discarded.

Command Syntax

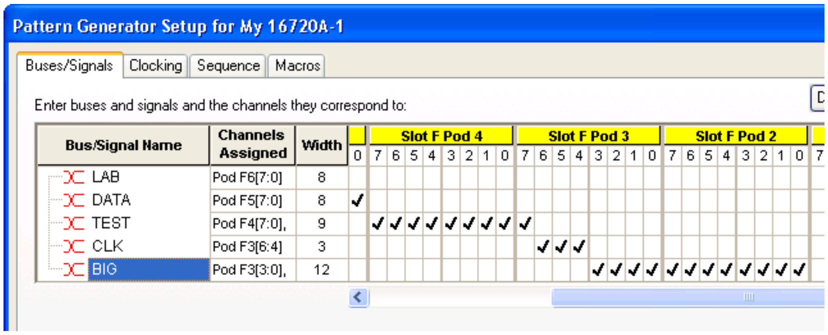
```
LABel <name_str>,<width>
```

<name_str> = label string a maximum of 20 characters in length.

<width> = integer number of bits in the label (1 through 32).

Example

```
ASCII     000000
ASCDOWN
FORMAT: MODE FULL
LABEL LAB,8
LABEL DATA,8
LABEL "TEST",9
LABEL CLK,3
LABEL BIG,12
VECTOR
12 34 56 7 89A
0 22 7 0 FFF
A0 33 00 1 111
*M
92 6F 00 1 FF0
CA CA 00 1 00F
00 10 11 0 ABC
```



Pattern Generator Setup for My 16720A-1

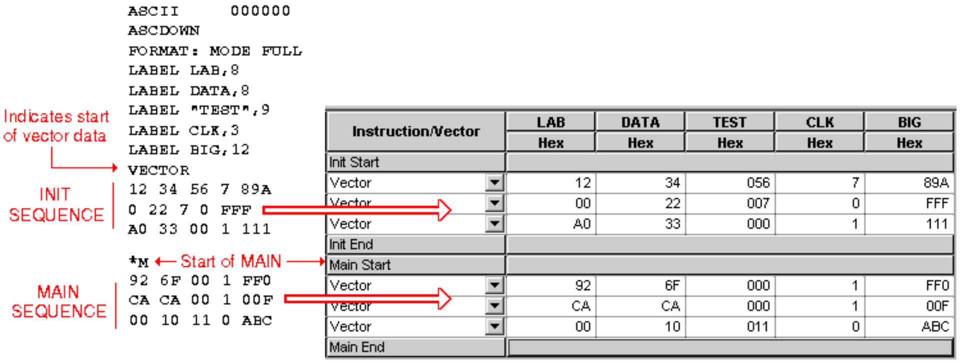
Buses/Signals | Clocking | Sequence | Macros

Enter buses and signals and the channels they correspond to:

Bus/Signal Name	Channels Assigned	Width		Slot F Pod 4								Slot F Pod 3								Slot F Pod 2								
			0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7
LAB	Pod F6[7:0]	8																										
DATA	Pod F5[7:0]	8	✓																									
TEST	Pod F4[7:0],	9		✓	✓	✓	✓	✓	✓	✓	✓	✓																
CLK	Pod F3[6:4]	3											✓	✓	✓													
BIG	Pod F3[3:0],	12														✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	

VECTor Command The VECTor command is used after the end of the header/setup commands to signal the start of the actual pattern generator data in an ASCII file. No data is allowed in the same line as the VECTor command.

Example



```
ASCII     000000
ASCDOWN
FORMAT: MODE FULL
LABEL LAB,8
LABEL DATA,8
LABEL "TEST",9
LABEL CLK,3
LABEL BIG,12
VECTOR
12 34 56 7 89A
0 22 7 0 FFF
A0 33 00 1 111

*M
92 6F 00 1 FF0
CA CA 00 1 00F
00 10 11 0 ABC
```

Instruction/Vector	LAB	DATA	TEST	CLK	BIG
	Hex	Hex	Hex	Hex	Hex
Init Start					
Vector	12	34	056	7	89A
Vector	00	22	007	0	FFF
Vector	A0	33	000	1	111
Init End					
Main Start					
Vector	92	6F	000	1	FF0
Vector	CA	CA	000	1	00F
Vector	00	10	011	0	ABC
Main End					

Vector Data The data portion of the ASCII file is an array of hexadecimal data fields. Each row of the array corresponds to a single line of the main program. Each column of the array corresponds to a single label as defined under the pattern generator Buses/Signals tab.

Data fields are separated by one or more blank characters. A line termination (line feed or carriage return + line feed) signals the end of a line and the start of a new line. If a data field has more data than the label width would indicate, only the least significant bits of the data field are used. If there are more data fields in a row than there are labels, the extra data fields (last data fields in the row) are ignored. If there are fewer data fields in a row than there are labels, the data for the extra (right-most) labels will be zero.

The MAIN sequence must have at least two data lines. In the ASCII data file, a row consisting of only "*M" signals the start of the MAIN sequence. If there is to be no data in the INIT sequence, the first row of the file after the VECTor command must be "*M". Note that the quotation marks in "*M" are not really in the file.

CAUTION Any character that is not a valid hexadecimal digit (that is, 0 through 9, or upper/lower case A through F) are ignored and treated as field separators. This could cause problems if a mistyped character appears in the middle of a data value. For example, "12R4" will be assigned to two labels as "12" and "4".

Either of the INIT or MAIN sequences can include multiple Repeat indicators specified by "*R <count>". Note that the quotation marks around the Repeat indicator are not part of the indicator, and like the "*M", the "*R <count>" must be located on a separate line.

The Repeat indicator specifies that the previous data vector is repeated <count> more times, where <count> is a positive integer. The Repeat indicator must follow a sequence line containing a data vector or another Repeat indicator. Placing the Repeat indicator after the VECTor command or the "*M" will cause an error message to appear.

The last data row of the file must end with a line termination character. The line termination character is the flag to load the data row into the data structure. Failure to do this will result in the last data row not being loaded.

The ASCII file import mechanism assumes correctness in the data file and any header commands. Error handling is basic, and treating unexpected characters as field separators could create bizarre results when parsing the file. Error messages point to the line number where the parser finds the error.

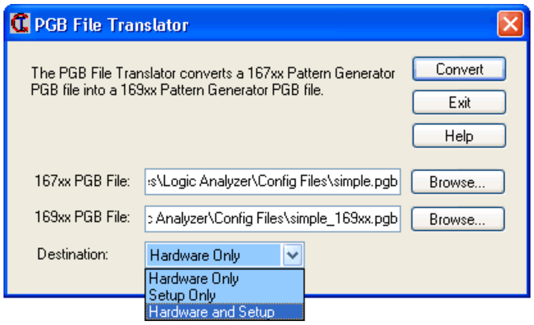
Serious problems will cause the default main program to be loaded in an effort to avoid locking up the logic analysis system.

Converting 16700-Series PGB Files

If you have a PattGen Binary (PGB) file from the 16700-series logic analysis system, you can convert it into a PGB file that can be used in the 16900-series logic analysis system.

1 From the Windows Start menu, choose **Start>All Programs>Agilent Logic Analyzer>Utilities>Pattern Generator>167xx Pattern Generator PGB Translator**.

2 In the PGB File Translator dialog, enter the name of the **167xx PGB File** you want to translate.



The PGB File Translator converts a 167xx Pattern Generator PGB file into a 169xx Pattern Generator PGB file.

167xx PGB File: s\Logic Analyzer\Config Files\simple.pgb

169xx PGB File: Analyzer\Config Files\simple_169xx.pgb

Destination: Hardware Only

3 Enter the name for the **169xx PGB File** translation output.

4 Select the desired **Destination** option. A DESTINATION command (see page 130) is inserted into the output 169xx PGB file, specifying where the vectors are placed when the PGB file is imported:

- Hardware Only — vectors are imported into the pattern generator module's hardware (memory). In this case, the pattern generator is set up to run without doing a compile and load beforehand.
- Setup Only — vectors are imported into the Sequence tab of the Pattern Generator Setup dialog.
- Hardware and Setup — vectors are imported into both the pattern generator module's hardware (memory) and the Sequence tab of the Pattern Generator Setup dialog.

5 Click **Convert**.

6 After the conversion has been performed, click **Exit** to close the PGB File Translator dialog.

7 Import the PGB file as described in Importing PattGen Binary (PGB) Format Files (see page 116).

Differences Between 16700- and 16900-Series PGB Format

16700-Series PGB	16900-Series PGB
Pods ordered from top card to bottom card and from left pod to right pod.	Pods ordered from bottom card to top card and from right pod to left pod.
Pod numbers start from 0.	Pod numbers start from 1.
In half-channel mode, pods are numbered consecutively starting from 0 (that is, 0, 1, 2, ...).	In half-channel mode, pod numbers don't change (that is, 5, 3, 1).
WAIT [A|B|C|D] <integer>	WAITEXT [A|B|C|D] <integer>
WAIT IMB <integer>	WAITARM "EXTERNAL TRIGGER" <integer>
SIGNAL <integer>	SENDARM <integer>

9
Pattern Generator Reference

- Key Characteristics (see page 144)

Key Characteristics

Characteristic	16720A Pattern Generator
Output Channels:	24 channels per card at 300 MHz clock (half channel mode); 48 channels per card at 180 MHz clock (full channel mode).
Memory Depth:	16,777,216 vectors in half channel mode; 8,388,608 vectors in full channel mode.
Logic Levels (data pods):	TTL, 3-state TTL/3.3v, 3-state TTL/CMOS, ECL/PECL/LVPECL terminated, ECL unterminated, and differential ECL (without pod).
Data Inputs:	3-bit pattern level sensing (clock pod).
Clock Output:	Synchronized to output data, delay of 7 ns in 14 steps (clock pod).
Clock Input:	DC to 300 MHz (clock pod).
Internal Clock Period:	Programmable from 1 MHz to 300 MHz in 1 MHz steps.
External Clock Period:	DC to 300 MHz.
External Clock Duty Cycle:	1.3 ns minimum high time.
Maximum Number of "Wait" Event Patterns:	4

10
Pattern Generator Control, COM Automation

The *Agilent Logic Analyzer* application includes the COM Automation Server. This software lets you write programs that control the *Agilent Logic Analyzer* application from remote computers on the Local Area Network (LAN).

In a COM automation program, you can configure a pattern generator module by:

- Loading a configuration file (which configures the complete logic analyzer setup).
- Using the "Module" (in the online help) object's "DoCommands" (in the online help) method with an XML-format string parameter (see Pattern Generator Setup, XML Format (see page 147)).

You can get information about a module's configuration using the Module object's "QueryCommand" (in the online help) method. Queries supported by the pattern generator module are listed below.

For more information about logic analyzer COM automation and tool objects in general, see "COM Automation" (in the online help).

XML-Based Queries Supported

The pattern generator module supports the following XML-based queries (made with the "Module" (in the online help) object's "QueryCommand" (in the online help) method).

Query	Description
GetAllSetup	Returns the current setup, using the full tag set, used for writing generic configuration files (see the XML format <Module> element (see page 154)).
GetBusSignalSetup	Returns only the BusSignalSetup settings (see the XML format <BusSignalSetup> element (see page 150)).

GetClockingSetup	Returns only the ClockingSetup settings (see the XML format <ClockingSetup> element (see page 153)).
GetModule	Returns the current setup, using the full tag set, equivalent to "GetAllSetup" (see the XML format <Module> element (see page 154)).

See Also

- "COM Automation" (in the online help)
- Pattern Generator Setup, XML Format (see page 147)

11
Pattern Generator Setup, XML Format

When you save logic analyzer configurations to XML format files, setup information for the pattern generator module is included.

This XML format setup information is also used when writing COM automation programs to control the pattern generator from a remote computer.

XML elements for the pattern generator module have the following hierarchy:

```
<Module> (see page 154)
   <BusSignalSetup> (see page 150)
      <BusSignals> (see page 149)
         <BusSignal> (see page 148)
            <Channels> (see page 151)
   <ClockingSetup> (see page 153)
      <Clocking> (see page 152)
   <SequenceSetup> (see page 156)
      <Sequence> (see page 155)
```

See Also

- "XML Format" (in the online help)
- Pattern Generator Control, COM Automation (see page 145)

<BusSignal> Element

The <BusSignal> element contains a pattern generator bus/signal definition.

Attributes

Name	Description
Comment	'string'
DefaultBase	'Binary', 'Hex', 'Octal', 'Decimal', or 'Signed Decimal'
Name	'string'
Polarity	'Positive' or 'Negative'

Children This element can have the following children: <Channels> (see page 151).

Parents This element can have the following parents: <BusSignals> (see page 149).

Example

```
<BusSignals>
   <Clear />
   <BusSignal Name='My Bus 1' Polarity='Positive'
         DefaultBase='Hex' Comment=''>
      <Channels>Pod 1[7:0]</Channels>
   </BusSignal>
</BusSignals>
```

<BusSignals> Element

The <BusSignals> element contains pattern generator bus/signal definitions.

Children This element can have the following children: "<Clear/>" (in the online help), <BusSignal> (see page 148).

Parents This element can have the following parents: <BusSignalSetup> (see page 150).

Example

```
<BusSignalSetup>
   <BusSignals>
      <Clear />
      <BusSignal Name='My Bus 1' Polarity='Positive'
            DefaultBase='Hex' Comment=''>
         <Channels>Pod 1[7:0]</Channels>
      </BusSignal>
   </BusSignals>
</BusSignalSetup>
```

<BusSignalSetup> Element

The <BusSignalSetup> element contains a <BusSignals> element.

Children This element can have the following children: <BusSignals> (see page 149).

Parents This element can have the following parents: <Module> (see page 154).

Example

```
<Module Name='My 16720A-1'>
   <BusSignalSetup>
      <BusSignals>
         <Clear />
         <BusSignal Name='My Bus 1' Polarity='Positive'
               DefaultBase='Hex' Comment=''>
            <Channels>Pod 1[7:0]</Channels>
         </BusSignal>
      </BusSignals>
   </BusSignalSetup>
   <ClockingSetup>
      <Clocking OutputMode='Full' ClockSource='Internal'
            ClockFrequency='180' ClockOutDelay='0 s' />
   </ClockingSetup>
   <SequenceSetup>
      <Sequence FileName='s2p01of01.pgb' />
   </SequenceSetup>
</Module>
```

<Channels> Element

The <Channels> element specifies the pattern generator channels assigned to the bus/signal.

Data This element's data is a string representing the selected channels for a bus/signal (as shown in the user interface — see To reorder bits in assigned channels (see page 46)).

Parents This element can have the following parents: <BusSignal> (see page 148).

Example

```
<BusSignal Name='My Bus 1' Polarity='Positive' DefaultBase='Hex'
      Comment=''>
   <Channels>Pod 1[7:0]</Channels>
</BusSignal>
```

<Clocking> Element

The <Clocking> element describes the pattern generator clocking setup configuration options.

Attributes

Name	Description
ClockFrequency	'integer' (in MHz representing the vector output rate in the currently selected mode, only valid when ClockSource='Internal')
ClockOutDelay	'number "time_unit" (in the online help)'
ClockSource	'External' or 'Internal'
OutputMode	'Full' or 'Half'

Parents This element can have the following parents: <ClockingSetup> (see page 153).

Example

```
<ClockingSetup>
   <Clocking OutputMode='Full' ClockSource='Internal'
         ClockFrequency='180' ClockOutDelay='0 s' />
</ClockingSetup>
```

<ClockingSetup> Element

The <ClockingSetup> element contains elements that describe the pattern generator clocking setup configuration options.

Children This element can have the following children: <Clocking> (see page 152).

Parents This element can have the following parents: <Module> (see page 154).

Example

```
<Module Name='My 16720A-1'>
   <BusSignalSetup>
      <BusSignals>
         <Clear />
         <BusSignal Name='My Bus 1' Polarity='Positive'
               DefaultBase='Hex' Comment=''>
            <Channels>Pod 1[7:0]</Channels>
         </BusSignal>
      </BusSignals>
   </BusSignalSetup>
   <ClockingSetup>
      <Clocking OutputMode='Full' ClockSource='Internal'
            ClockFrequency='180' ClockOutDelay='0 s' />
   </ClockingSetup>
   <SequenceSetup>
      <Sequence FileName='s2p01of01.pgb' />
   </SequenceSetup>
</Module>
```

<Module> Element

The <Module> element describes the pattern generator configuration options.

Attributes

Name	Description
Name	'string'

Children This element can have the following children: <BusSignalSetup> (see page 150), <ClockingSetup> (see page 153).

Parents This element can have the following parents: "<Setup> (under Configuration)" (in the online help).

Example

```
<Module Name='My 16720A-1'>
   <BusSignalSetup>
      <BusSignals>
         <Clear />
         <BusSignal Name='My Bus 1' Polarity='Positive'
               DefaultBase='Hex' Comment=''>
            <Channels>Pod 1[7:0]</Channels>
         </BusSignal>
      </BusSignals>
   </BusSignalSetup>
   <ClockingSetup>
      <Clocking OutputMode='Full' ClockSource='Internal'
            ClockFrequency='180' ClockOutDelay='0 s' />
   </ClockingSetup>
   <SequenceSetup>
      <Sequence FileName='s2p01of01.pgb' />
   </SequenceSetup>
</Module>
```

<Sequence> Element

The <Sequence> element references the compiled vector sequence PattGen Binary (PGB) file.

Attributes

Name	Description
FileName	'string' (name of PGB file)

Parents This element can have the following parents: <SequenceSetup> (see page 156).

Example

```
<SequenceSetup>
   <Sequence FileName='s2p01of01.pgb' />
</SequenceSetup>
```

<SequenceSetup> Element

The <SequenceSetup> element contains elements that reference the compiled vector sequence PattGen Binary (PGB) file.

Children This element can have the following children: <Sequence> (see page 155).

Parents This element can have the following parents: <Module> (see page 154).

Example

```
<Module Name='My 16720A-1'>
   <BusSignalSetup>
      <BusSignals>
         <Clear />
         <BusSignal Name='My Bus 1' Polarity='Positive'
               DefaultBase='Hex' Comment=''>
            <Channels>Pod 1[7:0]</Channels>
         </BusSignal>
      </BusSignals>
   </BusSignalSetup>
   <ClockingSetup>
      <Clocking OutputMode='Full' ClockSource='Internal'
            ClockFrequency='180' ClockOutDelay='0 s' />
   </ClockingSetup>
   <SequenceSetup>
      <Sequence FileName='s2p01of01.pgb' />
   </SequenceSetup>
</Module>
```

Index