Help Volume

© 2000-2002 Agilent Technologies Inc. All rights reserved.

Instrument: Agilent Technologies 16720A 300 M Vectors/s Pattern Generator

Using the Agilent Technologies 16720A Pattern Generator

The Agilent 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.

Getting Started

- "Overview of the Agilent Technologies 16720A Pattern Generator" on page 10
- "A Beginner's Exercise" on page 17

Creating the Program

- "Building an Initialization Sequence" on page 21
- "Building a Main Sequence" on page 23
- "Building a User Macro" on page 25
- "Importing Agilent 16522A ASCII Files" on page 28
- "Importing Agilent 16720A PattGen Binary Files" on page 39
- "Importing System Data Files" on page 58
- "Loading and Saving Pattern Generator Configurations" on page 61

See Also

"Selecting the Correct Probe Pod" on page 62

"Connecting the Probe Pods" on page 74

"Editing Sequences" on page 76

"Working with Instruction Types" on page 81

"Working with Labels and Pods" on page 88

"Working with Macro Parameters" on page 101

"Working with Automatic Pattern Fills" on page 104

"Printing the Pattern Generator Window" on page 111

"Printing Vector Sequences to a File" on page 112

"Viewing a Compiled Sequence" on page 119

Using the Intermodule Window (see the *Agilent Technologies 16700A/B-Series Logic Analysis System* help volume)

"Key Characteristics" on page 114

Main System Help (see the *Agilent Technologies 16700A/B-Series Logic Analysis System* help volume)

Glossary of Terms (see page 121)

Contents

Using the Agilent Technologies 16720A Pattern Generator

1 Using the Agilent Technologies 16720A Pattern Generator

Overview of the Agilent Technologies 16720A Pattern Generator 10
Mapping Probe Pods to the Interface 11
Vector Output Mode 12
Clock Source 13
Building a Sequence of Test Vectors 15
Running the Pattern Generator 16

A Beginner's Exercise 17
Configure Format 17
Configure Sequence 18
Set up the Workspace 19
View the Results 19

Building an Initialization Sequence 21

Building a Main Sequence 23

Building a User Macro 25

Importing Agilent 16522A ASCII Files 28
Creating an ASCII File 30
ASCII Disk File Identifier 32
ASCII File Commands 32

Importing Agilent 16720A PattGen Binary Files 39
Creating a PattGen Binary File 40
Binary File Commands 43

Importing System Data Files 58
Data Sets 59
Data Set Labels 59
Data Set Range 60

Loading and Saving Pattern Generator Configurations 61

Selecting the Correct Probe Pod 62
Data Pod Descriptions 64
Clock Pod Descriptions 69

Connecting the Probe Pods 74

Editing Sequences 76
Cutting, Copying, Pasting, and Deleting Sequence Lines 76
Deleting Sequence Lines 77
Inserting Blank Sequence Lines 78
Go to a Line Number 79
Positioning the Sequence 79
Using Ditto " values 80

Working with Instruction Types 81
The Break Instruction 82
The Signal IMB Instruction 82
The Wait IMB Event Instruction 83
The Wait External Event Instruction 83
The User Macro Instruction 84
The Repeat Loop Instruction 85

Contents

Working with Labels and Pods 88
Creating and Inserting New Labels 89
Deleting Labels 90
Inserting Pre-assigned Labels 91
Renaming Existing Labels 91
Reordering a Label's Pod Bits 92
Turning Labels On/Off 92
Clearing Format Labels 93
Searching for Labels 93
Swap Pods 93
Clear Pods 94
Assigning Bits to a Label 94
Label Polarity 95
Finding a label 95
Replace Labels 96
Appending Labels 97
Insert All Labels 98
Delete All Labels 98
Setting the Label Font Size 98
Adjusting Column Width 99
Setting Column Color 99
Setting the Numeric Base 100
Rearranging the Label Order 100

Working with Macro Parameters 101
Turning Parameters On 101
Inserting Parameters into a Macro 102
Assigning Parameter Values 102
Removing Parameters from a Macro 103

Working with Automatic Pattern Fills 104
Generating a Fixed Pattern Fill 104
Generating a Count Pattern Fill 105
Generating a Rotate Pattern Fill 106
Generating a Toggle Pattern Fill 108
Generating a Random Pattern Fill 109

Printing the Pattern Generator Window 111

Printing Vector Sequences to a File 112

Key Characteristics 114

Automatic Cursor Wrap 116

Recalling Macros 117

Copying Macros 118

Viewing a Compiled Sequence 119

Glossary

Index

1

Using the Agilent Technologies 16720A Pattern Generator

Overview of the Agilent Technologies 16720A Pattern Generator

Description of the Agilent 16720A Pattern Generator

The Agilent 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.

A Conceptual Measurement Example

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 run the pattern generator.

1. Select the probing (see page 62) that is compatible with your target circuit.
2. Set the Vector Output mode (see page 12) and the Clock Source (see page 13) parameters.
3. Connect the probes (see page 74) to your circuit and map the probe channels (see page 11) into the interface of the pattern generator.
4. Build a sequence of test vectors (see page 15) to generate the desired output signals.
5. Run (see page 16) the pattern generator and measure the active target circuit or prototype 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 61) pattern generator configurations.
- You can import an Agilent 16522A ASCII file (see page 28) (can only be used for vector sets <= 1048576 vectors)
- You can import a Agilent 16720A PattGen Binary file (see page 39) (must be used for vector sets >1048576 vectors)

See Also

"A Beginner's Exercise" on page 17

"Key Characteristics" on page 114

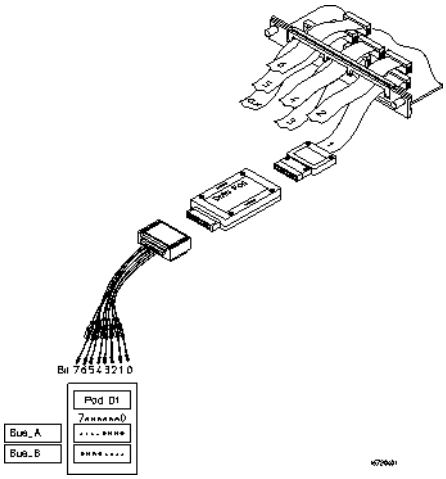
Mapping Probe Pods to the Interface

While the probes make the physical connection to your target circuit, a software connection is also made within the interface which routes generated signals to the proper probe output lines. This software connection is done within the Format tab and is called mapping.

The mapping process consists of logically grouping output signals that have a similar purpose to a label (see page 88) with a unique name. To add to or delete signals from a group, you simply turn On/Off the bits (see page 94) beside the label.

Example

This example shows eight channels (or bits) on probe pod 1 mapped to two labels in the interface. Bits 0-3 are assigned to label Bus_A, and bits 4-7 are assigned to label Bus_B. When a bit is assigned, an asterisk "*" appears in the bit assignment field, verifying the software connection.



NOTE: After you set up your first measurement, use the ribbon cable ID clips to mark the data cable numbers for future reference.

Vector Output Mode

Output Mode Full Channel 180Mbit/s

The Vector 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.

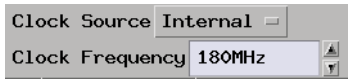
	Full Channel 180 Mbits/s	Half Channel 300 Mbits/s
Pods Available	Pods 1, 2, 3, 4, 5, 6	Pods 1, 3, 5
Maximum Channels	48; 8 per pod	24; 8 per pod
Maximum External Clock Frequency	180 MHz	300 MHz
Maximum Internal Clock Frequency	180 MHz	300 MHz
Minimum External Clock Frequency	DC	DC
Minimum Internal Clock Frequency	1 MHz	1 MHz

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



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 steps of 1. If you use the keypad to select a value between the step intervals, the value is rounded to the nearest interval.

NOTE: If you use the keypad to change the the clock frequency value, you must press the *Enter key* to register the new value. You are not required to type "Hz", you may use only a metric prefix or you may enter a floating point number, i.e. "50M" and "5e7" will both be displayed as "50MHz" after you select enter. You may also enter the value as a period, i.e. "20n" and "2e-8" will both be displayed as "50MHz".

The minimum clock period available with Vector Output Mode at Full Channel 180Mbit/s is 1MHz. Maximum clock frequency for Full Channel Mode is 180MHz. The minimum clock frequency available with Vector Output Mode at Half Channel 300Mbit/s is 1MHz. Maximum clock frequency for Half Channel Mode is 300MHz.

External Clock Source

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.

When using an external clock source the maximum clock period for the Vector Output Mode at Full Channel 180Mbit/s is 180 MHz. The maximum clock period for the Vector Output Mode at Half Channel 300Mbit/s is 300 MHz.

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

Clock Out Delay

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.

Each numerical change of one on the counter results in an approximate change of 500 ps.

Building a Sequence of Test Vectors

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 16), 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 run once, while the MAIN sequence loops if you select a repetitive run.

Using Hardware and Software Instructions

In addition to test vectors, both INIT and MAIN sequences can include predefined instruction (see page 81) elements. Instructions can create Breaks, Loops, and Wait, and can even signal the Intermodule Bus. The most useful instruction is "*User Macro*". With a User Macro instruction, you can 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.

“Building an Initialization Sequence” on page 21

“Building a Main Sequence” on page 23

“Building a User Macro” on page 25

“Working with Instruction Types” on page 81

Running the Pattern Generator

If you are not changing the run options, simply select the *Run* icon to run a measurement. Select the *Stop* icon to stop a measurement.

See Also

"What Happens when Run is Selected" on page 16

"What Happens when Stop is Selected" on page 16

What Happens when Run is Selected

In single run mode, the vectors are output from the first vector in the initialization sequence to the last vector of the main sequence. The last vector of the main sequence will be held at the outputs until you run again.

In repetitive run mode, the vectors in the initialization sequence will be output from first to last, one time, then the main sequence will repetitively output the vectors in that sequence until you select the *Stop* icon.

What Happens when Stop is Selected

When the pattern generator acknowledges stop, the vector currently being output will be held at the outputs until you run again.

A Beginner's Exercise

This exercise begins with the pattern generator Format tab active. If it is not active, select *Format* in the pattern generator window now.

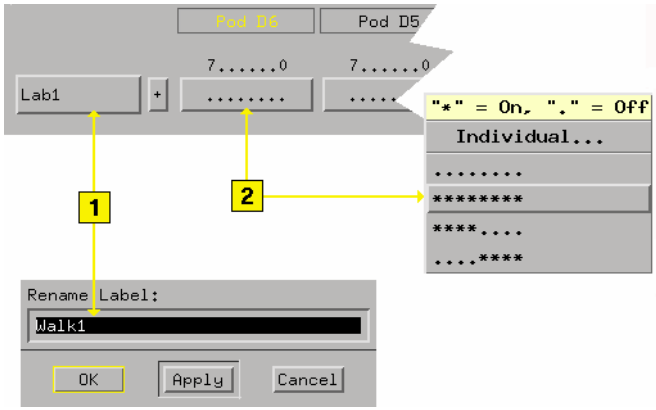
In this exercise, you will create a label with eight output channels assigned to it. You will then create a "Walking Ones" output pattern using one of the automatic pattern fill functions.

NOTE: This exercise does NOT require you to connect the probes or view the output. The intent of this exercise is to give you practice configuring the pattern generator interface. A timing analyzer display of the results is furnished for you.

1. Configure Format (see page 17) with a label called "Walk1" and all eight bits of Pod 6 assigned.
2. Select *Sequence*.
3. Configure Sequence (see page 18) with a Walking Ones sequence.
4. Set up the Workspace. (see page 19)
5. View the results. (see page 19)

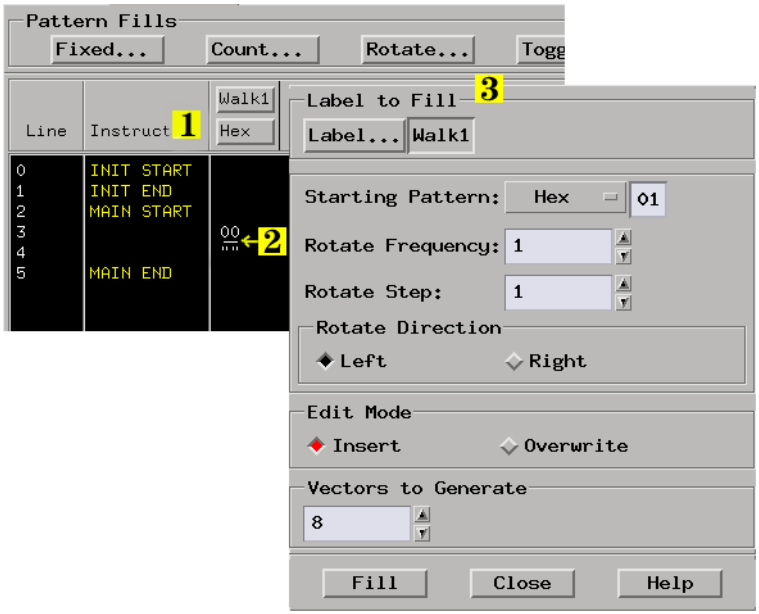
Configure Format

1. In the pattern generator's Format area, select the label *Label1*, then select *Rename*. In the Rename dialog, enter "Walk1", then select *OK*.
2. Select the bit assignment field under Pod 6 and select the menu item with all eight bits set to "*" (on).



Configure Sequence

1. In the pattern generator's Sequence window, select *Hex* and set the numeric base to Binary.
2. Select the first sequence line. This positions the cursor.
3. Select *Rotate*, and configure the Rotate Pattern Fill dialog as shown. Select *Fill*.
4. Select *Close*.

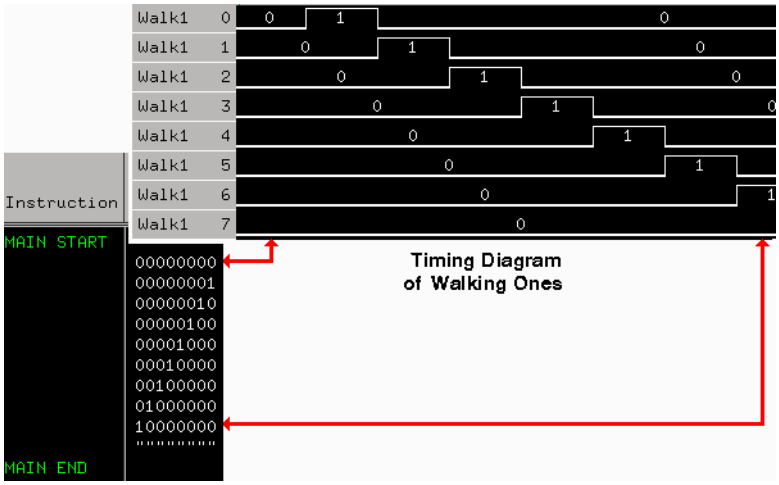


Set up the Workspace

1. Open the system Workspace window.
2. Drag and drop a waveform icon onto the pattern generator icon.
3. Select the waveform icon, then select *Display*.
4. Select the *Run* icon.

View the Results

1. Select *Walk 1 all* in the waveform display, then select *Expand*.
2. Set the *Samples/div* to 1.


Walk1 0
Walk1 1
Walk1 2
Walk1 3
Walk1 4
Walk1 5
Walk1 6
Walk1 7
Instruction
MAIN START
00000000
00000001
00000010
00000100
00001000
00010000
00100000
01000000
10000000
MAIN END
Timing Diagram
of Walking Ones

Building an 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 Signal IMB instruction to begin a measurement when the MAIN sequence begins. If you leave the INIT sequence empty, it will be ignored.

What Happens when Run is Selected

In single run mode, the vectors are output from the first vector in the initialization sequence to the last vector of the main sequence. The last vector of the main sequence will be held at the outputs until you execute run again.

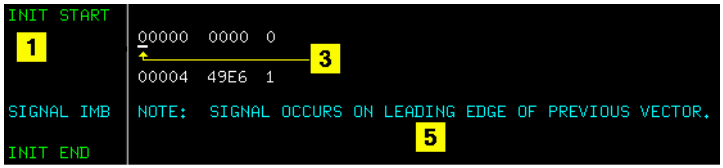
In repetitive run mode, the vectors in the initialization sequence will be output from first to last, one time, then the main sequence will repetitively output the vectors in that sequence until you select the *Stop* icon.

Building the INIT Sequence

The INIT sequence can contain hardware and software instructions (see page 81) 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 After*, then select *Vector*.
3. Repeat for each new vector line you want to insert.
4. Select the left-most character in the new vector line.
5. Select *Edit*.
6. Enter in the desired vector data. As you enter the information, the default cursor wrap (see page 116) setting will roll the cursor left-to-right and top line to bottom line.

7. Optional - If applicable, insert an instruction (see page 81) instead of entering vector data.



See Also

"Working with Labels and Pods" on page 88

"Editing Sequences" on page 76

"Working with Instruction Types" on page 81

"Building a User Macro" on page 25

"Automatic Cursor Wrap" on page 116

Building a 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.

What Happens when Run is Selected

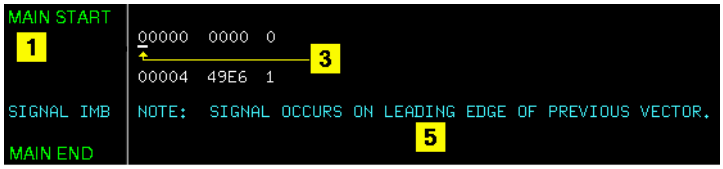
In single run mode, the vectors are output from the first vector in the initialization sequence to the last vector of the main sequence. The last vector of the main sequence will be held at the outputs until you select run again.

In repetitive run mode, the vectors in the initialization sequence will be output from first to last, one time, then the main sequence will repetitively output the vectors in that sequence until you select the *Stop* icon.

Building the Main Sequence

The MAIN sequence can contain hardware and software instructions (see page 81) 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 After*, then select *Vector*.
3. Repeat for each new vector line you want to insert.
4. Select the left-most character in the new vector line.
5. Select *Edit*.
6. Enter in the desired vector data. As you enter the information, the default cursor wrap (see page 116) setting will roll the cursor left-to-right and top line to bottom line.
7. Optional - If applicable, insert an instruction (see page 81) instead of entering vector data.



See Also

"Working with Labels and Pods" on page 88

"Editing Sequences" on page 76

"Working with Instruction Types" on page 81

"Building a User Macro" on page 25

"Automatic Cursor Wrap" on page 116

Building a User Macro

A User Macro is a vector sequence defined by a custom name, then inserted by name into a sequence wherever the macro is needed. Macros may be inserted into the INIT or MAIN sequences of the vectors in Sequence, or into other macros. Using macros gives you the benefit of keeping INIT or MAIN sequences generic. By simply interchanging macros, you change the pattern generator output.

NOTE: Care should be taken to avoid infinite loops. For example, if macro 0 calls macro 1, and macro 1 calls macro 0, this will cause an infinite loop.

Macros can also accept parameters (see page 101). A major benefit in using parameters is that you keep a macro's functionality generic and still direct specific action identified by parameters. Think of a parameter as the only part of a macro that changes as the macro is reused. Each macro can accept a maximum of 10 unique parameters.

Typically, you create a macro first under the *Macro* tab, then insert it into sequences under the *Sequence* tab. You can create 100 different macros for use in one or more pattern generator sequences.

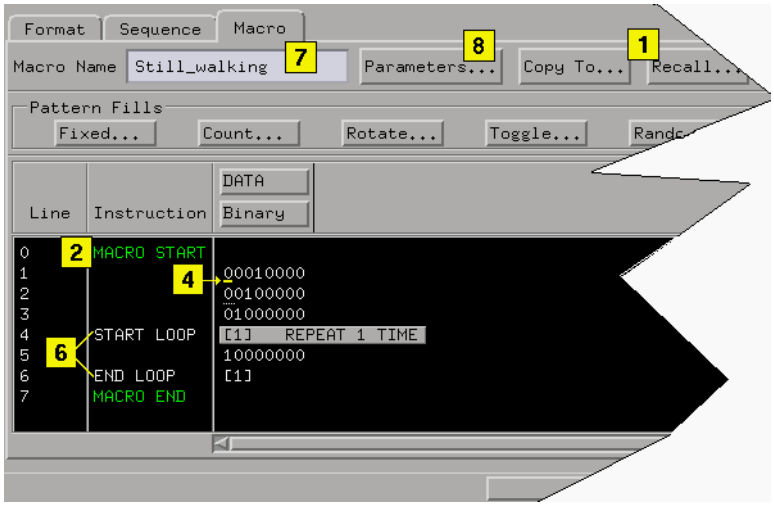
Differences between User Macros and the INIT and MAIN sequences are that macros cannot use any instruction that interacts with the intermodule bus (IMB). The reason is that these instructions can only be included once into the sequence. Since macros may be called as many times as desired, allowing these instructions within macros would violate this restriction. You remove macros from sequences by using the *Delete Line(s)* function.

Creating the Macro

A macro sequence can contain hardware and software instructions (see page 81) as well as vector data. However, instructions are not allowed on the first vector line.

1. In Macro, recall (see page 117) the macro that you want to create.
2. Select MACRO START.
3. Select *Insert After*, then select *Vector*.

4. Repeat for each new vector line you want to insert.
5. Select left-most character in the new vector line.
6. Enter in the desired vector data. As you enter the information, the default cursor wrap (see page 116) setting will roll the cursor left-to-right and top line to bottom line.
7. Optional - Insert an instruction (see page 81) instead of entering vector data.
8. Enter in a name for the new macro.
9. Optional - Select *Parameters* and turn on any parameters you plan to use.



Inserting the Macro

1. In *Sequence* or *Macro*, select the vector line directly above where you want to insert the User Macro instruction.
2. Select *Insert After*, then select *User Macro*.
3. Select the User Macro instruction.
4. From the Macro Selection dialog that appears, select the desired macro name you want to insert.
5. Select *OK*.

NOTE: See The User Macro Instruction (see page 84) for restrictions on User Macro instruction usage.

See Also

"Recalling Macros" on page 117

"Copying Macros" on page 118

"Working with Macro Parameters" on page 101

"Working with Instruction Types" on page 81

"Editing Sequences" on page 76

"Working with Labels and Pods" on page 88

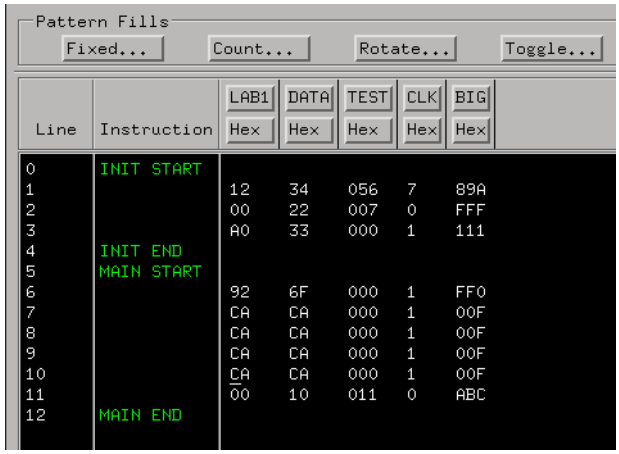
Importing Agilent 16522A ASCII Files

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.

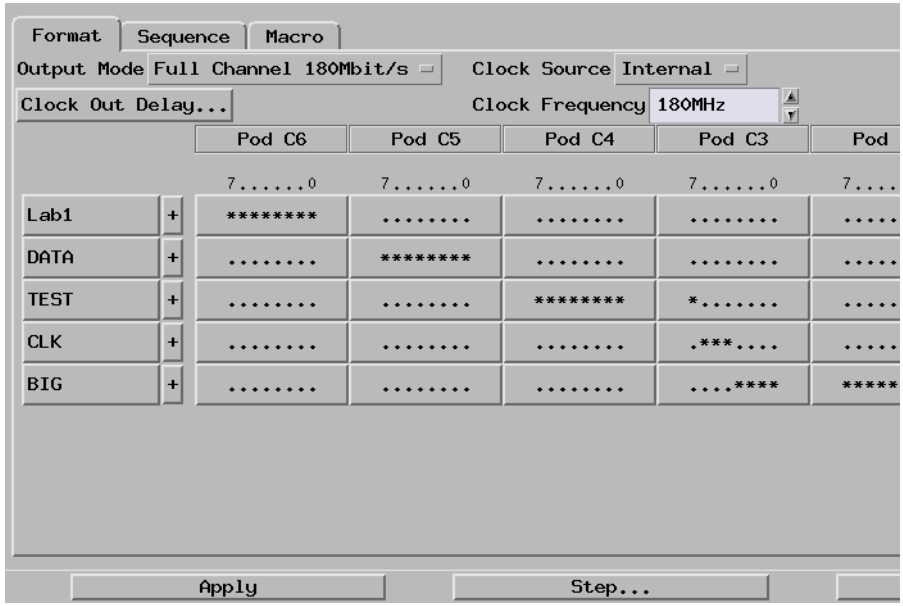
1. Create the ASCII file (see page 30) in a text editor.
2. Save the file as *Text Only* or *ASCII Format* in a directory on your analyzer's hard drive.
3. In the Sequence menu bar, select *File*, then *Import 16522A ASCII File*. See the caution below.
4. From the file selection dialog that appears, select the desired path and ASCII file name.
5. Select *Import*.

CAUTION: Importing an Agilent *16522A ASCII* file causes all current Format and Sequence information to be overwritten. Be sure to save the pattern generator configuration before you begin the import process.

NOTE: Importing a 1048576 line 16522A ASCII file may take approximately two minutes. If a 1048576 sequence is in memory when you begin an import, it may take up to a minute to clear the current data and up to two minutes to import the new data.



This figure shows Format after the ASCII file example shown in step 1 was imported.



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 32), 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 32)<lf>
/
/ This is a test of the ascii file
/
ASCDown (see page 33)<lf>
FORMat:MODE FULL (see page 36)<lf>
FORMat:CLOCk INTernal, 10E-9 (see page 36)<lf>
LABel LAB1,8 (see page 33)<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 34)<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 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

```
ASC11      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
```

ASCII File Commands

The following commands are used in the ASCII file to configure Sequence and Format. 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

"ASCDown Command" on page 33

FORMat Commands

MODe (see page 36)

CLOCk (see page 36)

DELay (see page 37)

ASCII Disk File Indentifier (see page 32)

"LABel Command" on page 33

"VECTor Command" on page 34

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

```
ASC11      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
```

LABel Command

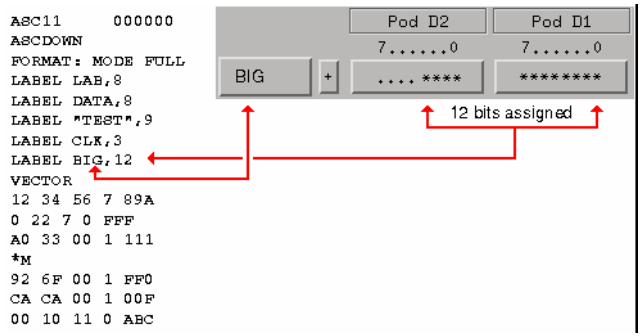
The LABel command is a special means of specifying labels for use by an ASCII file. The label bits are assigned from most to least significant bits across the output pods. You must specify the label string (quotation marks on the string are optional) and the width of the field. The label base defaults 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).

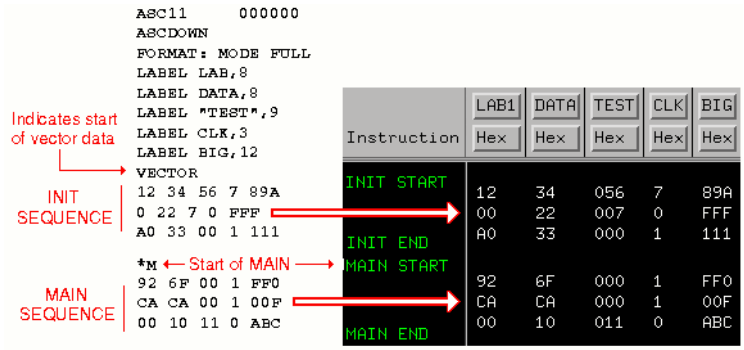
Example



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



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 Format 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.

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

```
ASC11     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>
```

= 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 36) command.

Example

```
ASC11      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

```
ASC11      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
```

Importing Agilent 16720A PattGen Binary Files

You can create a PattGen Binary file and import it as a complete pattern generator program. In general, the PattGen Binary file consists of a set of ASCII commands followed by a binary representation of the vectors. The file must be organized as shown in step 1 of the procedure below.

1. Create the PattGen Binary file (see page 40).
2. Make the file available in a directory on your logic analysis system hard drive using the various connectivity methods available from the logic analysis system. These include FTP, NFS, or PC file sharing.
3. Under the *Format* tab, select *File*, then select *Import PattGen Binary File*. See the Caution Below
4. From the file selection dialog, select the desired path and PattGen Binary file name.
5. Select *Import*.

CAUTION: Importing an Agilent *16720 PattGen Binary* file causes all current Format and Sequence information to be overwritten. Be sure to save the pattern generator configuration before you begin the import process.

When a PattGen Binary file is loaded, the GUI changes in the following ways:

- The Sequence and Macro tabs are disabled.
- The *Output Mode* (full channel/half channel) button is disabled.
- Channel assignments can not be edited.

The reason for each of these changes is the same. When a PattGen Binary file is loaded, it is loaded directly into the 16720A hardware. Because PattGen Binary files can be very large there is no internal representation that can be edited and/or displayed as there is for the 16522A ASCII or System Date files.

Changing among Full and Half channel modes requires memory to be reloaded. Changing channel assignments or adding labels with channel

assignments also requires memory to be reloaded.

If you save a configuration after loading a PattGen file, the configuration will not contain the binary data. You will have to reload the original PattGen Binary file.

Vectors that are loaded from a PattGen Binary file can be viewed in a Listing Window attached to the output of the 16720A icon on the Workspace. The Listing Window displays vectors by reading them from memory on 16720A card.

You can exit the *Binary Mode* (where editing is disabled) either by loading a non-binary file (i.e. 16522A ASCII) or by selecting *Enable Sequence Tab* in the *File* menu. This causes the vectors that were loaded from the binary file to be replaced by the default vectors, which are two vectors of all zero values.

Creating a PattGen Binary File

The PattGen Binary is primarily designed for deep (>1048576) vector sets and the various other formats (i.e. 16522A ASCII) will be used for shorter sets. The deep vector sets can be generated from tools, such as Verilog simulators, and translated to PattGen Binary by third party tools or customer-developed translation utilities. The changes to vectors, labels, etc. should be made in the tool that originally generated the vectors (i.e. Verilog).

Example

C/C++ code for generating PattGen Binary is available on the logic analysis system in the directory */logic/demo/pattgen* which contains these files.

- simple_pg.c- A very simple example of how to generate PattGen Binary.
- simple pgb- The PattGen Binary file created by simple_pg.c
- pattgen.c- A more extensive example with command line options. (For Unix based systems.)
- pattgen.exe - A more extensive example with command line options. (For

PC based systems.)

- binary_yacc.y- A simple utility that will parse and print PattGen Binary.
- binary_lex.1- A simple utility that will parse and print PattGen Binary.
- makefile- Rules for building the examples with typical C/C++ compilers.
- ToolDevKit - A directory containing instructions for installing a Workspace Tool that will generate a PattGen Binary file from vectors captured with a logic analyzer. See the README file in that directory for details.
- fastReader - A utility that will translate Fast Binary files to PattGen Binary files. Source code and a .exe file are included. See the README file in that directory for details.

File Requirements and Precautions

- The file must contain only specified pattern generator commands (see page 43), 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.
- Labels may consist of adjacent bits or arbitrarily ordered bits.
- Each ACSII 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 illustrated. This example is available on the logic analysis system in the file /logic/pattgen/demo/simpe.pgb.

```
PGBINARY
MODE FULL
CLOCK INTERNAL 5e7
```

```
MAIN 0
LABEL Label0 POD 0 [7:0]
LABEL Label1 POD 1 [7:0]
LABEL Label2 POD 2 [7:0]
LABEL Label3 POD 3 [7:0]
LABEL Label4 POD 4 [7:0]
LABEL Label5 POD 5 [7:0]
WIDTH 6
DEPTH 4096
BEGIN
```

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.

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 Label0 POD 0 [7:0]
LABEL Label1 POD 1 [7:0}
LABEL Label2 POD 2 [7:0]
LABEL Label3 POD 3 [7:0]
LABEL Label4 POD 4 [7:0]
LABEL Label5 POD 5 [7:0]
WIDTH 6
DEPTH 4096
BEGIN
```

Binary File Commands

The following ASCII commands are used in the PattGen Binary file to configure Sequence and Format. Commands may be all uppercase or all lowercase, but not mixed. Commands may not be abbreviated or truncated.

Commands

PattGen Binary File Identifier (see page 42)

CLOCK (see page 43)

DELAY (see page 44)

MODE (see page 45)

RMODE (see page 46)

LABEL (see page 46)

ORDER (see page 48)

WIDTH (see page 49)

DEPTH (see page 50)

BREAK (see page 51)

EVENT (see page 52)

WAIT (see page 53)

SIGNAL (see page 54)

MAIN (see page 55)

BEGIN (see page 56)

Binary Data (see page 56)

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

Set the clock mode to externally clocked.

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 Label0 POD 0 [7:0]
LABEL Label1 POD 1 [7:0]
LABEL Label2 POD 2 [7:0]
LABEL Label3 POD 3 [7:0]
LABEL Label4 POD 4 [7:0]
LABEL Label5 POD 5 [7:0]
WIDTH 6
DEPTH 4096
BEGIN
```

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 Label0 POD 0 [7:0]
LABEL Label1 POD 1 [7:0]
LABEL Label2 POD 2 [7:0]
LABEL Label3 POD 3 [7:0]
LABEL Label4 POD 4 [7:0]
LABEL Label5 POD 5 [7:0]
WIDTH 6
DEPTH 4096
BEGIN
```

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 Label0 POD 0 [7:0]
LABEL Label1 POD 1 [7:0]
LABEL Label2 POD 2 [7:0]
LABEL Label3 POD 3 [7:0]
LABEL Label4 POD 4 [7:0]
LABEL Label5 POD 5 [7:0]
```

```
WIDTH 6
DEPTH 4096
BEGIN
```

RMODE Command

The RMODE command is optional. The existing run mode scheme is used if nothing is specified. SINGLE indicates that a run will output the INIT part followed by the MAIN part one time each. REPETITIVE indicates that a run will output the INIT part once followed by the MAIN part repeating until the *STOP* icon is selected.

Command Syntax

RMODE [SINGLE][REPETITIVE]

Example

```
PGBINARY
MODE FULL
RMODE SINGLE
CLOCK INTERNAL 5e7 // 50 MHz
MAIN 0
LABEL Label0 POD 0 [7:0]
LABEL Label1 POD 1 [7:0]
LABEL Label2 POD 2 [7:0]
LABEL Label3 POD 3 [7:0]
LABEL Label4 POD 4 [7:0]
LABEL Label5 POD 5 [7:0]
WIDTH 6
DEPTH 4096
BEGIN
```

LABEL Command

The LABEL command is a special means of specifying labels for use by a PattGen Binary file. The first <integer> following the POD keyword is the pod number. Pods are numbered from 0...N where N is the maximum number pods in a logic analysis system minus 1. In the 16720A the number of pods would be 0...29 in Full channel mode or 0...14 in Half channel mode when five cards are connected as a master

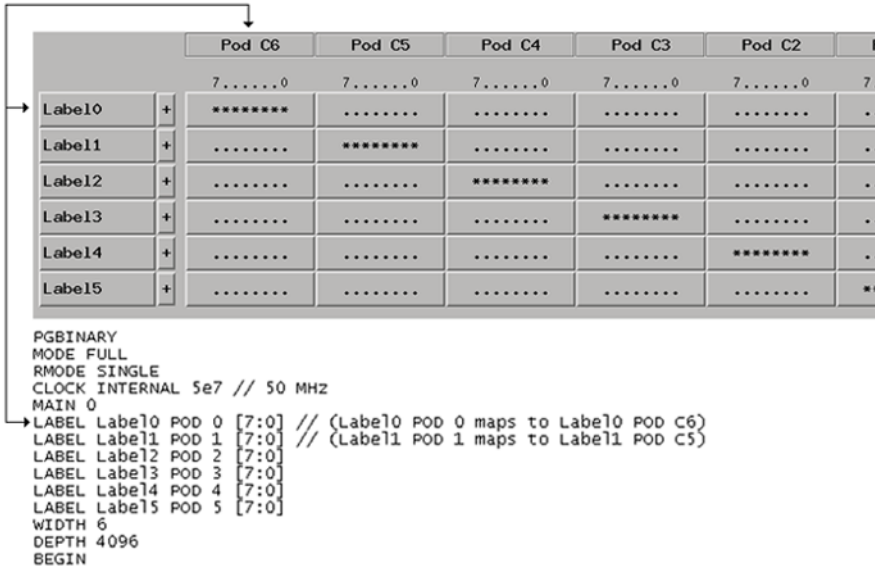
card and four expander cards. The pods are numbered from left to right. By using index numbers rather than the names they would have in the logic analysis system (i.e. B6 for Pod 6 in Slot B) the command is independent of how the logic analysis system is configured. Bit assignment are from Most Significant Bit to Least Significant Bit.

Command Syntax

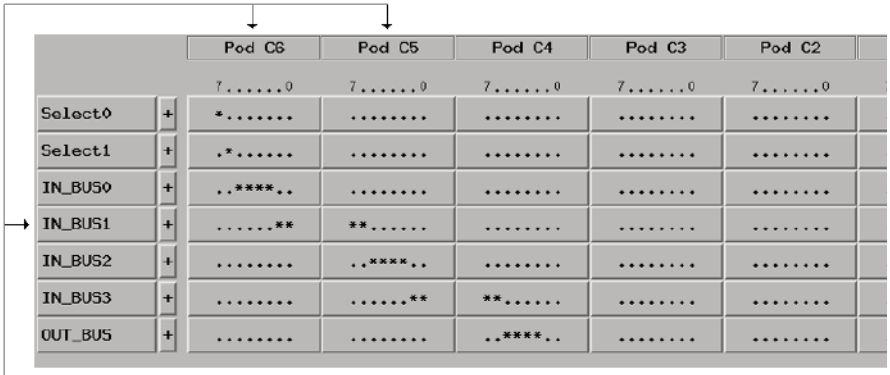
LABEL <name_str> { POD <integer> { <integer> | [<integer> : <integer>] }}
<name_str> = label string a maximum of 20 characters in length. The name string must begin with a letter (A-Z, a-z) or underscore (_) and can be followed by letters, underscores, or digits (0-9). A label may not be a keyword (see page 48). If you wish to use a keyword as a label, you may do so by enclosing it in quotes.
<integer> = POD index or bit number.

Example

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



Example



		Pod C6	Pod C5	Pod C4	Pod C3	Pod C2
		7.....0	7.....0	7.....0	7.....0	7.....0
Select0	+	*.....	.....	.....	.....	.....
Select1	+	.*.....	.....	.....	.....	.....
IN_BUS0	+	..****..	.....	.....	.....	.....
IN_BUS1	+	.....**	**.....	.....	.....	.....
IN_BUS2	+	.....	..****..	.....	.....	.....
IN_BUS3	+	.....	.....**	**.....	.....	.....
OUT_BUS	+	.....	.....	..****..	.....	.....

```
PGBINARY
MODE FULL
RMODE SINGLE
CLOCK INTERNAL 5e7 // 50 MHz
MAIN 0
LABEL Select0 POD 0 7
LABEL Select1 POD 0 6
LABEL IN_BUS0 POD 0 [5:2]
LABEL IN_BUS1 POD 0 [1:0] POD 1 [7:6] // (IN_BUS1 POD 0 [1:0] POD 1 [7:6] maps
LABEL IN_BUS2 POD 1 [5:2]
LABEL IN_BUS3 POD 1 [1:0] POD 2 [7:6]
LABEL OUT_BUS POD 2 [5:2]
WIDTH 7
DEPTH 4096
BEGIN
```

Keywords

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

A	DELAY	INTERNAL	POD
B	DEPTH	LABEL	REPETITIVE
BEGIN	EVENT	MAIN	RMODE
BREAK	EXTERNAL	MODE	SIGNAL
C	FULL	NONE	SINGLE
CLOCK	HALF	ORDER	WAIT
D	IMB	PGBINARY	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 Label0 POD 0 [7:0]
LABEL Label1 POD 1 [7:0]
LABEL Label2 POD 2 [7:0]
LABEL Label3 POD 3 [7:0]
LABEL Label4 POD 4 [7:0]
LABEL Label5 POD 5 [7:0]
ORDER Label 0 Label1 Label2 Label3 Label4 Label5
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 Label0 POD 0 [7:0]
LABEL Label1 POD 1 [7:0]
LABEL Label2 POD 2 [7:0]
LABEL Label3 POD 3 [7:0]
LABEL Label4 POD 4 [7:0]
LABEL Label5 POD 5 [7:0]
WIDTH 6
DEPTH 4096
BEGIN
```

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.

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 Label0 POD 0 [7:0]
LABEL Label1 POD 1 [7:0]
LABEL Label2 POD 2 [7:0]
LABEL Label3 POD 3 [7:0]
LABEL Label4 POD 4 [7:0]
LABEL Label5 POD 5 [7:0]
WIDTH 6
DEPTH 4096
BEGIN
```

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 Label0 POD 0 [7:0]
LABEL Label1 POD 1 [7:0]
LABEL Label2 POD 2 [7:0]
LABEL Label3 POD 3 [7:0]
LABEL Label4 POD 4 [7:0]
LABEL Label5 POD 5 [7:0]
BREAK 100
WIDTH 6
DEPTH 4096
BEGIN
```

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 WAIT 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 (hex)	WAIT_2	External Event Input Values WAIT_1	WAIT_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, i.e. 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 Label0 POD 0 [7:0]
LABEL Label1 POD 1 [7:0]
LABEL Label2 POD 2 [7:0]
LABEL Label3 POD 3 [7:0]
LABEL Label4 POD 4 [7:0]
LABEL Label5 POD 5 [7:0]
EVENT A #H01
WAIT A 100
WIDTH 6
DEPTH 4096
BEGIN
```

WAIT Command

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

Command Syntax

WAIT [IMB | A | B | C | D] <integer>
IMB = Inter-Module Bus.
A, B, C, D = external wait events.
<integer> = the vector where a hardware WAIT instruction is to be placed.

Example

```
PGBINARY
```

```
MODE FULL
RMODE SINGLE
CLOCK INTERNAL 5e7 // 50 MHz
MAIN 0
LABEL Label0 POD 0 [7:0]
LABEL Label1 POD 1 [7:0]
LABEL Label2 POD 2 [7:0]
LABEL Label3 POD 3 [7:0]
LABEL Label4 POD 4 [7:0]
LABEL Label5 POD 5 [7:0]
EVENT A #H01
WAIT A 100
WAIT IMB 200
WIDTH 6
DEPTH 4096
BEGIN
```

NOTE: A WAIT 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.

SIGNAL Command

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

PGBINARY

```
MODE FULL
RMODE SINGLE
CLOCK INTERNAL 5e7 // 50 MHz
MAIN 0
LABEL Label0 POD 0 [7:0]
LABEL Label1 POD 1 [7:0]
LABEL Label2 POD 2 [7:0]
LABEL Label3 POD 3 [7:0]
LABEL Label4 POD 4 [7:0]
```

```
LABEL Label5 POD 5 [7:0]
SIGNAL 100
WIDTH 6
DEPTH 4096
BEGIN
```

NOTE: A SIGNAL 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 Label0 POD 0 [7:0]
LABEL Label1 POD 1 [7:0]
LABEL Label2 POD 2 [7:0]
LABEL Label3 POD 3 [7:0]
LABEL Label4 POD 4 [7:0]
LABEL Label5 POD 5 [7:0]
WIDTH 6
DEPTH 4096
BEGIN
```

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
<binary data>

Example

```
PGBINARY
MODE FULL
RMODE SINGLE
CLOCK INTERNAL 5e7 // 50 MHz
MAIN 0
LABEL Label0 POD 0 [7:0]
LABEL Label1 POD 1 [7:0]
LABEL Label2 POD 2 [7:0]
LABEL Label3 POD 3 [7:0]
LABEL Label4 POD 4 [7:0]
LABEL Label5 POD 5 [7:0]
WIDTH 6
DEPTH 4096
BEGIN
```

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, i.e. the first byte is the most significant byte. This is followed by the binary data itself.

- 4 bytes = number of vectors.
- 4 bytes = 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 will require 3 bytes (24bits), a 16 bit label will require 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 (i.e. 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 (i.e. POD 7 for a 1 board system) the configuration is truncated to what fits and a warning is emitted.

Importing System Data Files

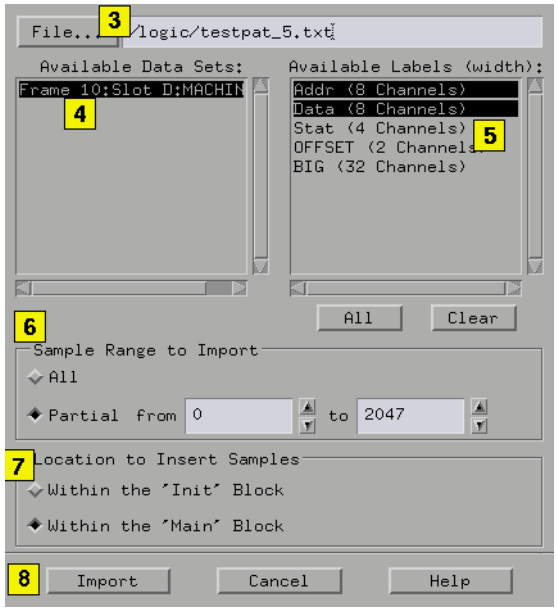
If you store logic analyzer data using the File Out tool, in either Internal, ASCII, or Fast Binary format, you can import these files into the pattern generator's INIT or MAIN sequences.

The "Import System Data File" dialog will only load files saved using the File Out tool. For ASCII files created in a PC or UNIX text editor, use Import 16522A ASCII File (see page 28). For PattGen Binary files created in a PC or UNIX program, use Import 16720A PattGen Binary File (see page 39).

CAUTION: Importing a *File Out* tool file causes all current Format and Sequence information to be overwritten. Be sure to save the pattern generator configuration before you begin the import process.

NOTE: Importing a 1048576 line System Data file may take approximately two minutes. If a 1048576 sequence is in memory when you begin an import, it may take up to a minute to clear the current data and up to two minutes to import the new data.

1. Select the Sequence tab.
2. Select *File* from the menu bar, then *Import System Data File*.
3. From the Import System Data File dialog, load the desired File Out tool file. See the note below.
4. Select the desired data set (see page 59).
5. Select the desired labels (see page 59).
6. Select a data set range (see page 60) to import. If you select "Partial", the range integers you specify will correspond to state numbers found in a state listing of the same data set.
7. Select the location to insert the samples. Samples are inserted at the beginning of the selected sequence. See the caution below.
8. Select Import.



NOTE: If in step 3 you entered the file name into the text entry field using a keyboard, you must press the Return key on your keyboard to start the file import process instead of selecting the *Import* field.

Data Sets

A data set is defined as data captured from a single source. For example, the data captured from analyzer one of a logic analyzer is a single source. Because the File Out tool allows multiple analyzers worth of data to be stored within the same file, you pick only one data set to import into the pattern generator.

Data Set Labels

All labels in a single data set are listed along with each label's corresponding channel width. If a label is wider than the pattern

generator's maximum allowable channel width (32 channels), the label is marked "unavailable". Labels wider than 32 channels cannot be selected.

Data Set Range

When you select a data set, the "from" and "to" fields update to the minimum and maximum values of the state listing. Both positive and negative integers are valid.

CAUTION: If the range of data samples is too large for the pattern generator sequence (1048576 vectors), a message will appear giving you the choice to continue or not. If you continue, data will be imported starting from the beginning of the file. At the point where you run out of vectors, data will be missing. To solve this problem, reduce the range of samples you are importing.

Loading and Saving Pattern Generator Configurations

You can save pattern generator settings and data to a configuration file. You can also save any tools connected to the pattern generator. Later, you can restore your data and settings by loading the configuration file.

- Loading Configuration Files (see the *Agilent Technologies 16700A/B-Series Logic Analysis System* help volume)
- Saving Configuration Files (see the *Agilent Technologies 16700A/B-Series Logic Analysis System* help volume)

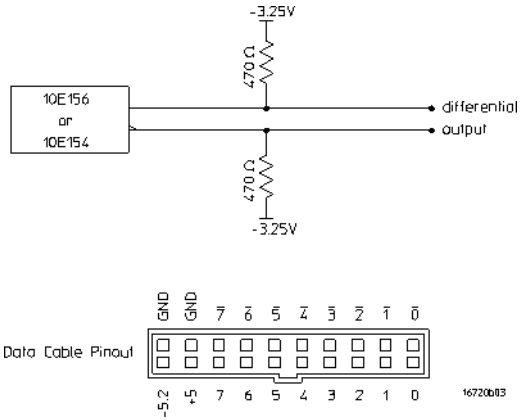
Selecting the Correct Probe Pod

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

"Data Pod Descriptions" on page 64

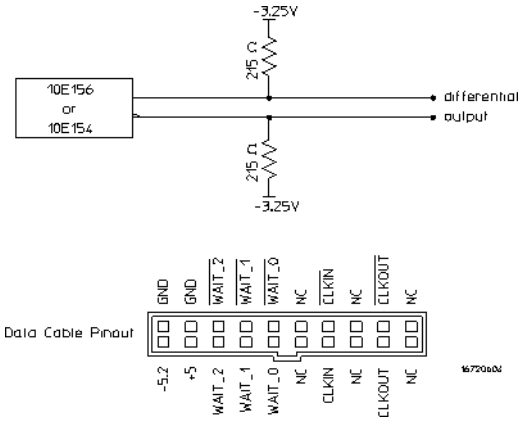
"Clock Pod Descriptions" on page 69

Data Cable Characteristics without a Data Pod

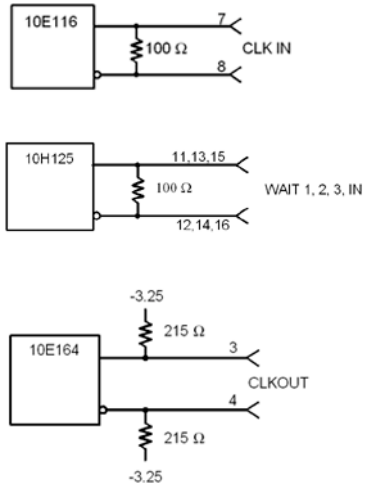


The data cables without a data pod provide an ECL-terminated (470 ohm; to -3.25V) differential signal. These signals are usable when received by a differential receiver, preferably with a 100 ohm termination across the lines. These signals should not be used single-ended due to the slow fall time and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

Clock Cable Characteristics without a Clock Pod



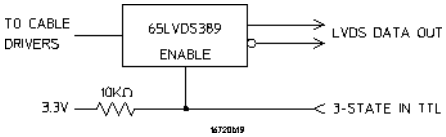
The clock out signals (CLKOUT and not-CLKOUT) without a clock pod provide an ECL-terminated (215 ohm to -3.25V) differential signal. These signals are usable when received by a differential receiver, preferably with a 100 ohm termination across the lines. These signals should not be used single-ended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



See Also "Connecting the Probe Pods" on page 74

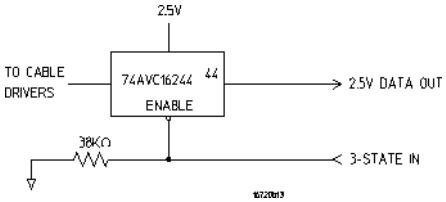
Data Pod Descriptions

E8141A LVDS Data Pod



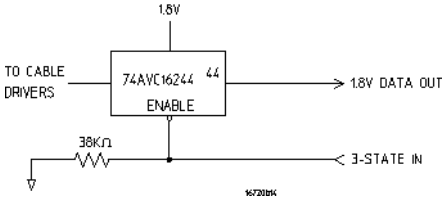
Output type	65LVDS389 (LVDS data lines) 10H125 (TTL non-3-state channel 7)
3-state enable	positive true TTL; no connect=enabled
Maximum clock	300 MHz
Skew	Typical less than 1 ns, worst case 2 ns
Recommended lead set	E8142A

10473A 3-State 2.5 V Data Pod



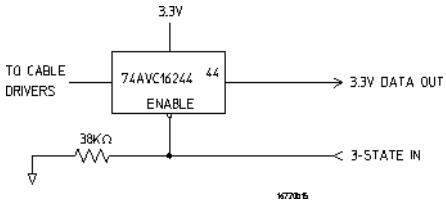
Output type	74AVC16244
3-state enable	negative true, no connect=enabled
Maximum clock	300 MHz
Skew	Typical less than 1 ns, worst case 2 ns
Recommended lead set	10498A

10476A 3-State 1.8 V Data Pod



Output type	74AVC16244
3-state enable	negative true, no connect=enabled
Maximum clock	300 MHz
Skew	Typical less than 1.5 ns, worst case 2.5 ns
Recommended lead set	10498A

10483 3-State 3.3V Data Pod

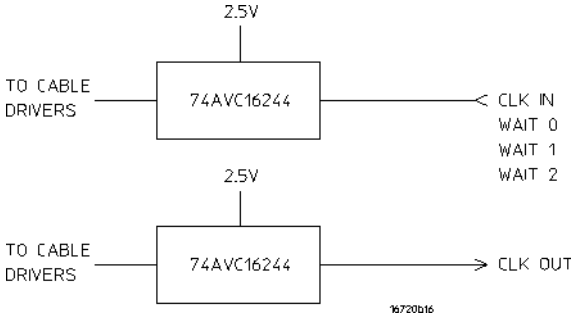



```
Output type              74AVC16244
3-state enable           negative true, no connect=enabled
Maximum clock            300 MHz
Skew                     Typical less than 1 ns, worst case 2 ns
Recommended lead set     10498A
```

E8140A LVDS Clock Pod

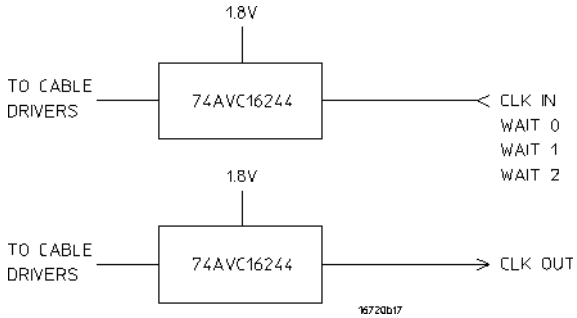
```
Clock output type        65LVDS179 (LVDS) and 10H125 (TTL)
Clock output rate        200 MHz maximum (LVDS and TTL)
Clock out delay          approximately 8 ns total in 14 steps
Clock input type         65LVDS179 (LVDS with 100 ohm)
Clock input rate         DC to 150 MHz (LVDS)
Pattern input rate       10H124 (TTL) (no connect=logic 1)
Clock in to clock out    approximately 30 ns
Patt in to recognition approximately 15 ns + 1 clock period
Recommended lead set     10498A
```

10472A 2.5 V Clock Pod



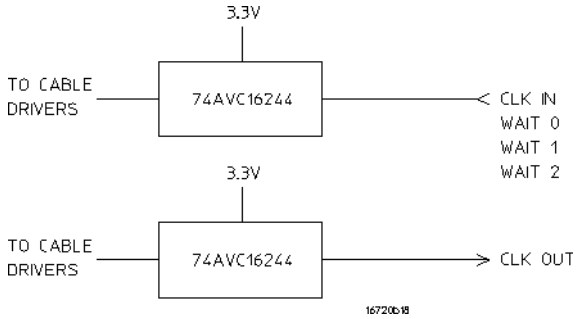

```
Clock output type        74AVC16244
Clock output rate        200 MHz maximum
Clock out delay          approximately 8 ns total in 14 steps
Clock input type         74AVC16244 (3.6 V maximum)
Clock input rate         DC to 200 MHz
Pattern input rate       74AVC16244 (3.6 V maximum; no connect=logic 0)
Clock in to clock out    approximately 30 ns
Patt in to recognition approximately 15 ns + 1 clock period
Recommended lead set     10498A
```

10475A 1.8 V Clock Pod



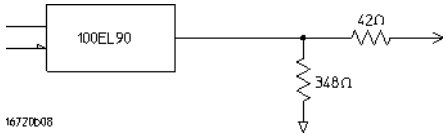

```
Clock output type       74AVC16244
Clock output rate       200 MHz maximum
Clock out delay         approximately 8 ns total in 14 steps
Clock input type        74AVC16244 (3.6 V maximum)
Clock input rate        DC to 200 MHz
Pattern input rate      74AVC16244 (3.6 V maximum; no connect=logic 0)
Clock in to clock out   approximately 30 ns
Patt in to recognition approximately 15 ns + 1 clock period
Recommended lead set    10498A
```

10477A 3.3 V Clock Pod



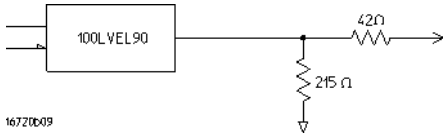

```
Clock output type       74AVC16244
Clock output rate       200 MHz maximum
Clock out delay         approximately 8 ns total in 14 steps
Clock input type        74AVC16244 (3.6 V maximum)
Clock input rate        DC to 200 MHz
Pattern input rate      74AVC16244 (3.6 V maximum; no connect=logic 0)
Clock in to clock out   approximately 30 ns
Patt in to recognition approximately 15 ns + 1 clock period
Recommended lead set    10498A
```

10469A PECL Data Pod



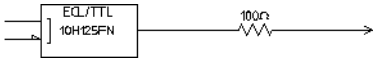
Output type	100EL09 (5V) with 348 pulldown to ground and 42 ohm in series
Maximum clock	300 MHz
Skew	Typical less than 500 ps, worst case 1 ns
Recommended lead set	10498A

10471A LVPECL Data Pod



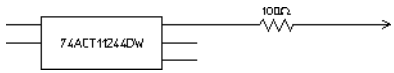
Output type	100LVEL90 (3.3V) with 215 ohm pulldown to ground and 42 ohm in series
Maximum clock	300 MHz
Skew	Typical less than 500 ps, worst case 1 ns.
Recommended lead set	10498A

10461A TTL Data Pod



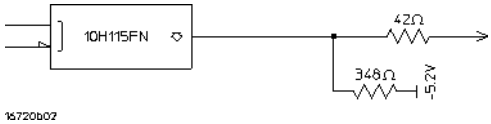
Output type	10H125 with 100 ohm in series
Maximum clock	200 MHz
Skew	Typical less than 2 ns; worst case 4 ns (note 1)
Recommended lead set	10498A

10462A 3-State TTL/CMOS Data Pod



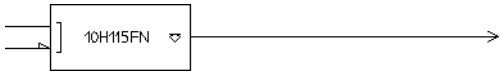
Output type	74ACT11244 with 100 ohm in series 10H125 on non 3-state channel 7 (note 2),
3-state enable	negative true, 100K ohm to GND, enabled on no connect.
Maximum clock	100 MHz
Skew	Typical less than 4 ns; worst case 12 ns (note 1)
Recommended lead set	10498A

10464A ECL Data Pod (terminated)



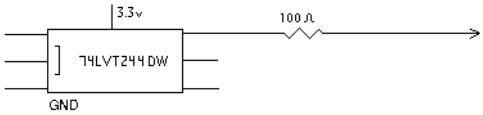
Output type	10H115 with 348 ohm pulldown, 42 ohm in series
Maximum clock	300 MHz
Skew	Typical less than 1 ns; worst case 2 ns (note 1)
Recommended lead set	10498A

10465A ECL Data Pod (unterminated)



Output type	10H115 (no termination)
Maximum clock	300 MHz
Skew	Typical less than 1 ns; worst case 2 ns (note 1)
Recommended lead set	10347A

10466A 3-State TTL/3.3 volt Data Pod



Output type	74LVT244 with 100 ohm in series
	10H125 on non 3-state channel 7 (see note 2)
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 (note 1)
Recommended lead set	10498A

Note 1

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.

Note 2

The pattern generator always holds the last set of state vectors when it is stopped. If you want to insure that the outputs are switched to a "no connect" condition, you must make use of the 3-state capability and turn outputs off via the "no connect" state. That is done by setting the "3-state in" enable pin to the appropriate logic level (depending on the pod you are using) for the duration of time you wish the 3-state outputs

to be enabled or disabled.

There are two pin7's; one is a normal 3-statable data pin, the other is an non 3-statable data pin (a straight-through pin7).

The pattern generator has a feature that allows you to control the 3-state condition via wiring the second straight-through pin7 to the "3-state in" enable pin, and then programmatically setting the level to control the 3-state status.

An alternative 3-state control can be implemented via an external voltage level input to the data output pod "3-state in" enable pin. In this mode, the straight-through bit7 can no longer be used as part of the data pattern, since it is being used to provide a voltage level to control the "3-state in" enable pin.

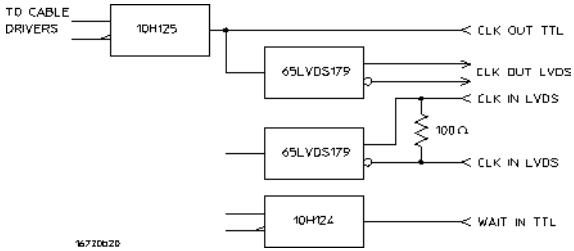
See Also

"Clock Pod Descriptions" on page 69

"Connecting the Probe Pods" on page 74

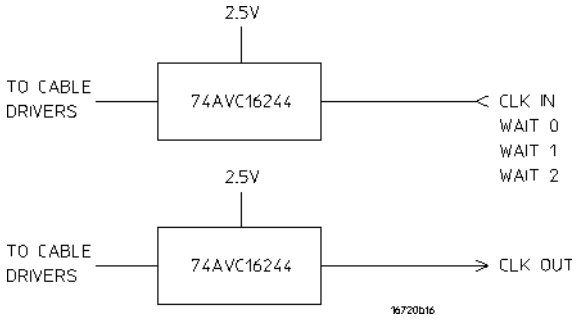
Clock Pod Descriptions

E8140A LVDS Clock Pod



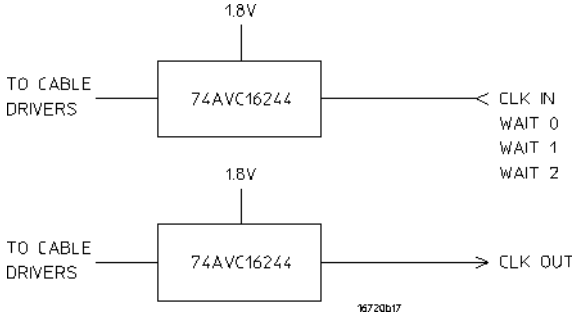
Clock output type	65LVDS179 (LVDS) and 10H125 (TTL)
Clock output rate	200 MHz maximum (LVDS and TTL)
Clock out delay	approximately 8 ns total in 14 steps
Clock input type	65LVDS179 (LVDS with 100 ohm)
Clock input rate	DC to 150 MHz (LVDS)
Pattern input rate	10H124 (TTL) (no connect=logic 1)
Clock in to clock out	approximately 30 ns
Patt in to recognition	approximately 15 ns + 1 clock period
Recommended lead set	10498A

10472A 2.5 V Clock Pod



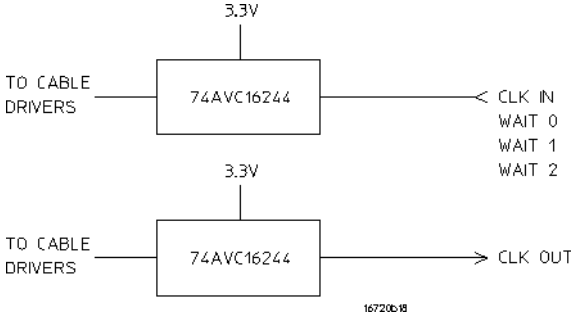

```
Clock output type       74AVC16244
Clock output rate       200 MHz maximum
Clock out delay         approximately 8 ns total in 14 steps
Clock input type        74AVC16244 (3.6 V maximum)
Clock input rate        DC to 200 MHz
Pattern input rate      74AVC16244 (3.6 V maximum; no connect=logic 0)
Clock in to clock out   approximately 30 ns
Patt in to recognition approximately 15 ns + 1 clock period
Recommended lead set    10498A
```

10475A 1.8 V Clock Pod



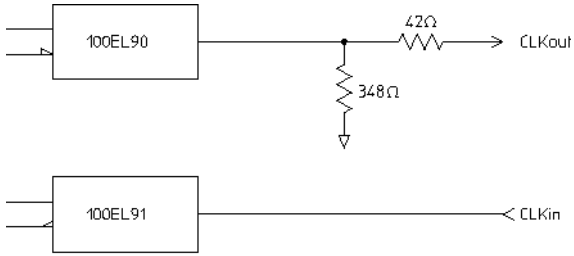

```
Clock output type       74AVC16244
Clock output rate       200 MHz maximum
Clock out delay         approximately 8 ns total in 14 steps
Clock input type        74AVC16244 (3.6 V maximum)
Clock input rate        DC to 200 MHz
Pattern input rate      74AVC16244 (3.6 V maximum; no connect=logic 0)
Clock in to clock out   approximately 30 ns
Patt in to recognition approximately 15 ns + 1 clock period
Recommended lead set    10498A
```

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)
Clock input rate         DC to 200 MHz
Pattern input rate       74AVC16244 (3.6 V maximum; no connect=logic 0)
Clock in to clock out    approximately 30 ns
Patt in to recognition approximately 15 ns + 1 clock period
Recommended lead set     10498A
```

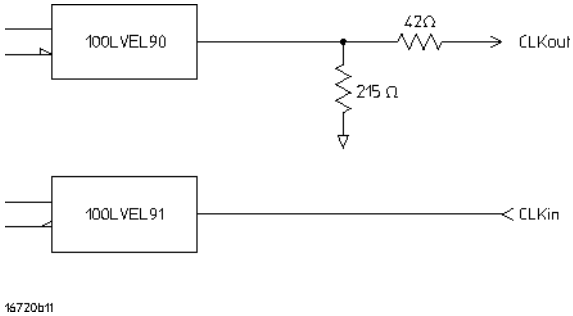
10468A PECL Clock Pod



16720b10

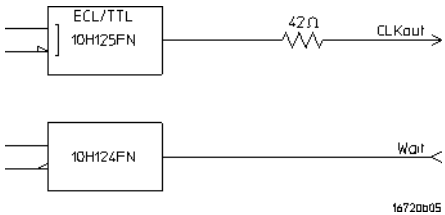
```
Clock output type        100EL90 (5V0 with 348 ohm pulldown to ground
                         and 42 ohm in series
Clock output rate        300 MHz maximum
Clock input delay        approximately 8 ns total in 14 steps
Clock input type         100EL91 PECL (5V), no termination (no connect
                         is logic 0)
Clock input rate         DC to 300 MHz
Pattern input type       100EL91 PECL (5V), no termination (no connect
                         is logic 0)
Recommended lead set     10498A
```

10470A LVPECL Clock Pod



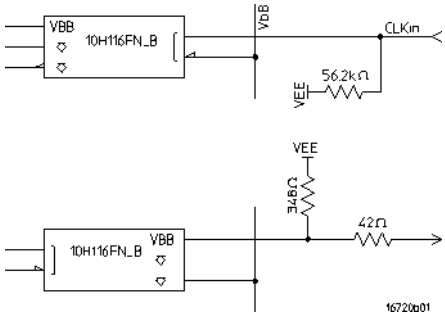
Clock output type	100LVEL90 (3.3V) with 215 ohm pulldown to ground and 42 ohm in series
Clock output rate	300 MHz maximum
Clock out delay	approximately 8 ns in 14 steps
Clock input type	100LVEL91 PECL (3.3V), no termination
Clock input rate	DC to 300 MHz
Pattern input type	100LVEL91 PECL (3.3V), no termination
Clock in to clock out	approximately 30 ns
Patt in to recognition	approximately 15 ns + up to 1 clock period
Recommended lead set	10498A

10460A TTL Clock Pod



Clock output type	10H125 with 42 ohm series; true & inverted
Clock output rate	100 MHz maximum
Clock out delay	8 ns maximum 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)
Clk-in to clk-out	Approx. 30 ns
Patt-in to recognition	Approx. 15 ns + up to 1 clk period
Recommended lead set	10498A

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	11 ns maximum in 9 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)
Clk-in to clk-out	Approx. 30 ns
Patt-in to recognition	Approx. 15 ns + up to 1 clk period
Recommended lead set	10498A

See Also

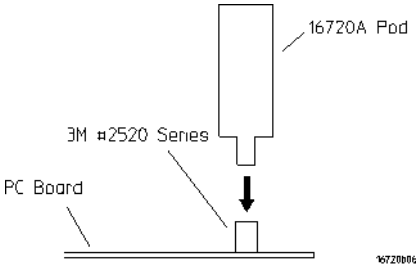
"Data Pod Descriptions" on page 64

"Connecting the Probe Pods" on page 74

Connecting the Probe Pods

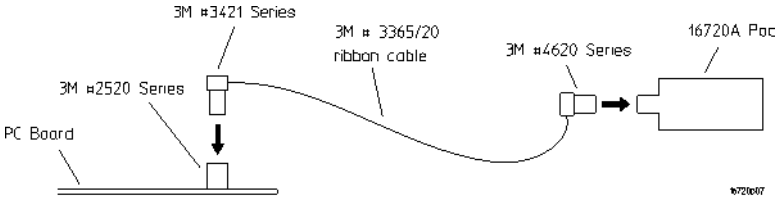
NOTE: Clock and Data pods are required for a proper signal interface. There are different types (see page 62) available, and they should match your target circuit characteristics. In addition, depending on the Vector Output Mode (see page 12) selected, some pods may not be available for use.

Direct Pod-to-Board Connection



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

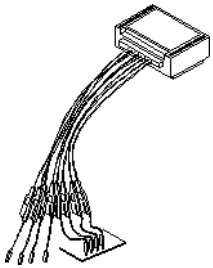
Jumper Cable-to-Pod Connection



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

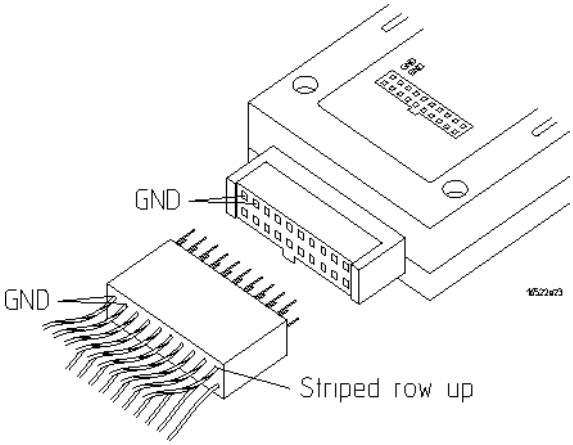


Two probe lead assemblies are available for connecting to PC board pins.

- 10498A 8-channel probe lead set.
- 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.



Editing Sequences

- “Cutting, Copying, Pasting, and Deleting Sequence Lines” on page 76
- “Deleting Sequence Lines” on page 77
- “Inserting Blank Sequence Lines” on page 78
- “Go to a Line Number” on page 79
- “Positioning the Sequence” on page 79
- “Using Ditto " values” on page 80

Cutting, Copying, Pasting, and Deleting Sequence Lines

NOTE: When you use the *Cut* and *Copy* operations from the menu bar, 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 from the menu bar, the sequence lines are not placed in the temporary buffer. They are just deleted.

1. Select the sequence lines to cut, copy, or delete by highlighting the first line.
2. To select the entire sequence, select *Edit* from the menu bar, then *Select All Lines*.
3. From the menu bar, select *Edit*, then *Cut Line(s)*, *Copy Line(s)*, or *Delete Lines(s)*.
4. Select the sequence line just above where you want to paste the sequence lines. This positions the cursor. When pasting sequence lines, the lines are placed *after* the cursor line.
5. If you are pasting cut or copied lines, select *Edit* from the menu bar, then *Paste Line(s)*.

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

Restrictions on Use

The above operations will not be allowed if the result of the operation places an instruction on one of the following vectors:

- 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.

In addition, the above operations will not be allowed if the result of the operation places a hardware instruction immediately after another instruction. Hardware instructions must follow data vectors.

Deleting Sequence Lines

CAUTION: When you delete sequence lines, they are permanently removed. If you want to place them in a temporary storage buffer, use the Cut (see page 76) operation from the menu bar.

NOTE: To highlight a line, drag from the bottom of the desired line to the top of the desired line.

Deleting Single Lines

- To delete single lines, highlight the line that is to be deleted.
- Select *Delete Line(s)* from the *Edit* menu.

Deleting Multiple Lines

- To delete multiple lines, highlight the lines that are to be deleted.
- Select *Delete Line(s)* from the *Edit* menu.

Delete All Lines

To delete all lines in a sequence, go to the menu bar and select *Edit -> Select All Lines*, then *Edit -> Delete Lines(s)*. Deleting all lines causes the sequence to be reset to the power-up state.

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 an 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.

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

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

Inserting Blank Sequence Lines

Inserting blank lines (see page 80) is a very useful operation when starting a new sequence. After you insert a new line, you replace the ditto values with the desired normal data values.

Inserting Blank Lines

1. To position the cursor, select the vector line directly above where you want to insert lines.
2. Select *Insert After*, then select *Vector* to insert a new vector line.
3. Repeat for each new vector line you want to insert.

NOTE: You can also insert blank sequence lines by using the keyboard as follows: * Select the vector line directly above where you want to insert the new data vector. * Press the Insert key on your keyboard one time for each new vector line you want to insert.

NOTE: The new blank lines are inserted with ditto values.

See Also

"Using Ditto " values" on page 80

Go to a Line Number

1. From within the *Sequence* or *Macro* tab, select any vector line.
2. Select *Goto Line*.
3. In the Goto Line dialog that appears, enter the desired line number.
4. Select *OK*.

See Also

"Positioning the Sequence" on page 79

Positioning the Sequence

Using the Keyboard

Among the keyboard keys that insert and delete data vectors, the pattern generator defines four other keys used to position the

sequence when either the *Sequence* or *Macro* tab is active. Specifically, these additional keys are defined:

1. Home - move to the first line of the sequence.
2. End - move to the last line of the sequence.
3. Page Up - scroll upward by one screen worth of data.
4. Page Down - scroll downward by one screen worth of data.

To use the positioning keys, perform the following steps:

1. From within Sequence or Macro, select any vector line.
2. Press one of the above keys on the keyboard.

See Also

"Go to a Line Number" on page 79

Using Ditto " values

In any existing data vector, with the exception of the first, you can replace normal data values with ditto values (designated by the double quotation marks).

Ditto values indicate that the previous data value should be repeated for the current data vector. Ditto values save you effort by requiring you to only enter in the data values that change.

NOTE: When you insert blank lines, they are inserted as ditto values.

1. From either the *Sequence* or *Macro* tab, select the *data field* that you want to edit. This positions the cursor.
2. Select *Insert After*.
3. Select *Vector*.

See Also

"Inserting Blank Sequence Lines" on page 78

Working with Instruction Types

Instructions are like programming commands. They are inserted into a sequence for the purpose of executing their designated control over the flow of the sequence. There are two types of instructions: hardware and software. The differences between these types are described below.

Inserting Instructions into Sequences

1. Select the vector line directly above where you want to insert the new instruction.
2. Select *Insert After*.
3. Select the desired instruction to insert.

Hardware Instruction Types

The following hardware instruction types are available. Each of these instructions can affect external hardware or the pattern generator hardware.

- "The Break Instruction" on page 82
- "The Signal IMB Instruction" on page 82
- "The Wait IMB Event Instruction" on page 83
- "The Wait External Event Instruction" on page 83

Software Instruction Types

The following software instruction types are available. Each of these instructions only affects the execution flow of the currently running sequence. However, a User Macro software instruction can include hardware instructions.

- "The User Macro Instruction" on page 84

See Also

- "The Repeat Loop Instruction" on page 85

"Building a Main Sequence" on page 23

"Building an Initialization Sequence" on page 21

"Building a User Macro" on page 25

The Break Instruction

The Break instruction causes a break at the current vector. In single run mode, this instruction halts the sequence and holds the outputs at the value of the previously outputted data value. In repetitive run mode, this instruction pauses the sequence at the current vector momentarily, then continues. The duration of the pause depends on the activity of other modules in the frame.

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 Signal IMB Instruction

The Signal IMB instruction creates an arming signal on the *intermodule bus* when the instruction is executed. This *arming* signal allows the pattern generator to *cross trigger* other modules in the frame.

Restrictions on Use

The Signal IMB instruction, as with all hardware instructions, must follow data vectors. Also, the Signal IMB 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 Signal IMB 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

Using the Intermodule Window (see the *Agilent Technologies 16700A/B-Series Logic Analysis System* help volume)

The Wait IMB Event Instruction

The Wait IMB Event instruction halts the execution of the program sequence until an IMB signal is received by the pattern generator.

Restrictions on Use

The Wait IMB Event instruction, as with all hardware instructions, must follow data vectors. Also, the Wait IMB Event 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 IMB Event 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 Wait External Event Instruction

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.

Because the Wait External Event is an OR'ing function of the toggled

"On" wait patterns, as soon as one of the selected wait patterns is satisfied, the sequence continues. The wait patterns are specified on the three input lines of the clock pod: Wait0, Wait1, and Wait2.

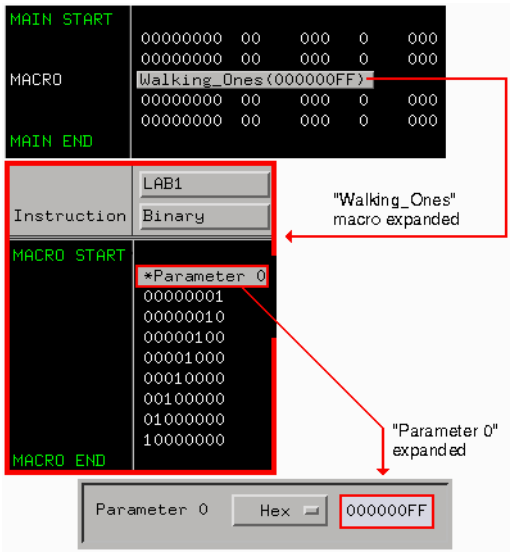
To choose and configure a particular event, select the Wait External Event instruction, then from the External Wait Pattern dialog, toggle the appropriate fields.

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 User Macro Instruction



The User Macro instruction lets you insert a group of vectors that have

a specific function. At run time, the user macro is expanded into its corresponding vectors. A typical macro application would be a generic test stimulus that is used by multiple circuits. By containing the device or circuit specific test vectors within a macro, you can simply interchange the macro and reuse the same INIT and MAIN SEQUENCE.

Macros can contain parameters. Using parameters in a macro gives you the same benefit as using macros in a sequence. By passing parameters, you can reuse the same macro for other purposes.

Restrictions on Use

The User 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.

See Also

"Building a User Macro" on page 25

"Working with Macro Parameters" on page 101

The Repeat Loop Instruction

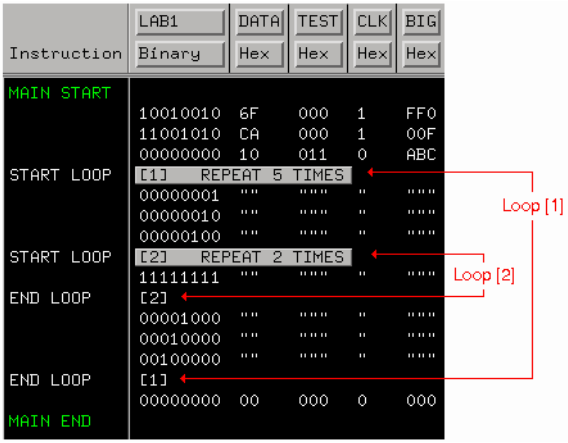
```
MAIN START
            10011110  6F    000   1    FF0
            11001010  CA    000   1    00F
START LOOP  [7]   REPEAT 5 TIMES
            00000001  ""    """   "    """
            00000010  ""    """   "    """
            00000100  ""    """   "    """
END LOOP    [7] _
            """""""""  ""    """   "    """
MAIN END
```

The Repeat Loop instruction inserts the start and end vectors of a repeat loop, along with one blank data vector row, below the selected vector row. Once the loop has been created, you can insert or copy vectors and instructions into the loop.

Set the number of loop repetitions (maximum 20,000) by selecting the "Start Loop" vector line and editing the *Loop Count* dialog. At run time, the loop count determines the number of times to expand the loop's body into individual vectors. 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 Repeat 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].



Restrictions on Use

The following instructions can NOT be used in a repeat loop:

- The Signal IMB instruction.
- The Wait IMB Event instruction.

The Repeat 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.

Working with Labels and Pods

Access label and pod operations from either the *Edit* pick in the menu bar, or under the label name field itself. When you select operations from the menu bar, they are global to all labels or pods in the window. When you select operations from under the label name, they are directed at the individual label.

In addition, label and pod operations in the *Sequence* and *Macro* displays are centered around adding or deleting. Operations in the *Format* display are centered around initial creation and configuration.

NOTE: Label operations are performed on labels assigned in Format. If labels are not created, assigned bits, and turned on in Format, they do not appear in Sequence or Macro. Also, depending on the Vector Output Mode (see page 12), some pods may not be available.

Operations in Format

- “Creating and Inserting New Labels” on page 89
- “Deleting Labels” on page 90
- “Renaming Existing Labels” on page 91
- “Reordering a Label's Pod Bits” on page 92
- “Turning Labels On/Off” on page 92
- “Clearing Format Labels” on page 93
- “Finding a label” on page 95
- “Swap Pods” on page 93
- “Clear Pods” on page 94
- “Assigning Bits to a Label” on page 94
- “Label Polarity” on page 95

Operations in Sequence and Macro

- “Inserting Pre-assigned Labels” on page 91

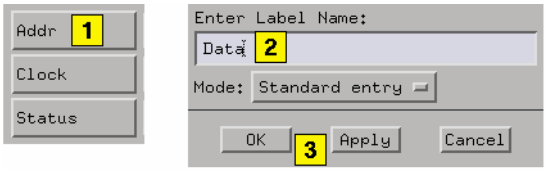
- “Deleting Labels” on page 90
- “Replace Labels” on page 96
- “Finding a label” on page 95
- “Appending Labels” on page 97
- “Insert All Labels” on page 98
- “Delete All Labels” on page 98
- “Searching for Labels” on page 93
- “Setting Column Color” on page 99
- “Adjusting Column Width” on page 99
- “Rearranging the Label Order” on page 100
- “Setting the Numeric Base” on page 100
- “Setting the Label Font Size” on page 98

Creating and Inserting New Labels

Creating New Labels in Format

You create labels in Format to uniquely identify groups of assigned pod bits (see page 94) that have a common purpose or identity. Inserting and assigning labels are part of the mapping process (see page 11) for your application.

1. Select the label where you want to insert the new label, then select *Insert before* or *Insert after*.
2. In the Enter Label Name dialog that appears, enter in the new label name. If you do not provide a label name, a default name is assigned.
3. Select *OK* if you are done, or *Apply* if you have more labels to insert.



NOTE: Duplicate label names are not allowed.

Mode

If you are creating a series of new labels, you have the option to automatically assign bits in a "walking ones" pattern across the label series. You do this by toggling the *Mode* field to *Walking ones*.

Select the Pod, the starting Bit, and the walk direction. As you enter new label names and select *Apply*, a single bit is automatically assigned under each new label in a "walking ones" pattern.

Deleting Labels

1. Select the label to be deleted.
2. Select *Delete*.

See Also

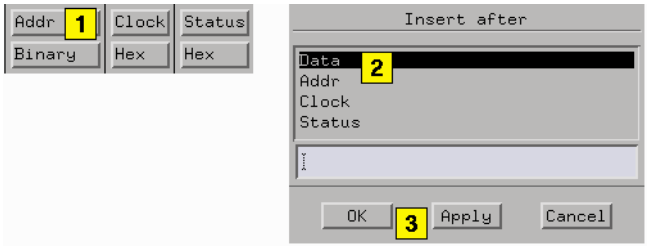
"Clearing Format Labels" on page 93

"Delete All Labels" on page 98

Inserting Pre-assigned Labels

NOTE: If labels are not created, assigned bits, and turned on in *Format*, they do not appear in subsequent label operation dialogs in *Sequence* and *Macro*.

1. From the *Sequence* or *Macro* tab, select the label where you want to insert the new label, then select *Insert Before* or *Insert After*.
2. In the Select Label dialog that appears, select the pre-assigned label name to insert.
3. Select *OK* if you are done, or *Apply* if you have more labels to insert.



See Also "Searching for Labels" on page 93

Renaming Existing Labels

1. Under *Format*, select the label you want to rename, then select *Rename*.
2. In the Rename dialog, enter in the new label name.
3. Select *OK* if you are done, or *Apply* if you have more labels to rename.

NOTE: When you select *Apply*, the next label is automatically highlighted. This lets you rename multiple labels very quickly.

Reordering a Label's Pod Bits

Use the *Reorder bits* feature to remap the physical probe connections to the interface without changing the actual probe connections. This feature allows the probe tips for each channel to be physically connected where convenient.

1. Under the *Format* tab, select the label you want to reorder bits on, then select *Reorder bits*.
2. Set the bit order by one of the following options:
   - To reorder bits individually, for each channel, enter the number of the bits you want to map the channel to.
   - To arrange the bits sequentially, select the field at the top of the dialog, then select *Default Order*.
   - To reverse the order of the bytes, select the field at the top of the dialog, then select *Big Endian - Little Endian* mapping. This option is only available for labels that have either 24 or 48 channels.
3. Select *OK*.

NOTE: Reordering the bits of a label does not reorder the physical output signals of the channels assigned to that label. The bit order of a label is a display-orientation concept, useful only when entering data values.

Turning Labels On/Off

Turning labels off prevents output signals from appearing on the associated probe channels. When a label is turned off, it is removed from the *Sequence* and *Macro* areas. However, the label's name and bit assignments are preserved.

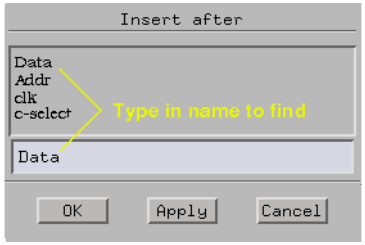
To toggle labels On/Off in Format, select the label that you want to modify and toggle the Label On/Off selection.

Clearing Format Labels

The *Clear Format Labels* command is located under *Edit* in the Format menu bar. When you select *Clear Format Labels*, all user-assigned labels are permanently removed, and the default label *Lab1* is reset.

Searching for Labels

In all label selection dialogs, where a list of label names appears, a text entry field is available for a quick search of label names. Simply enter in a search string, and all labels that begin with that string appear in the list.



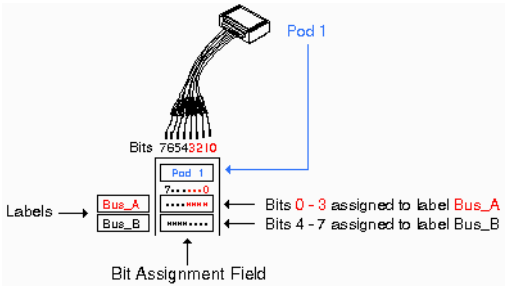
Swap Pods

The *Swap Pods* command is located under *Edit* in the Format menu bar. When you select *Swap Pods*, the bit assignments for the two designated pods are swapped across all labels.

Clear Pods

The *Clear Pods* command is located under *Edit* in the Format menu bar. When you select *Clear Pods*, all bit assignments for all labels under the designated pod are cleared.

Assigning Bits to a Label



The bits in a label correspond to the physical pattern generator probe channels. When you run the pattern generator, data is output on all bits (channels) that are assigned to labels. Unassigned bits are inactive.

- An asterisk "*" indicates an assigned bit.
- A period "." indicates an unassigned bit.

To Assign Bits

1. Select the bit assignment field to the right of the label name you want to define. Each bit assignment field corresponds to the data pod listed above it.
2. Select the bits you want to change, toggling them between an asterisk and a period.
3. When the bits are assigned as desired, close the dialog box.

Select the bit assignment field or dialog to see a shortcut menu for assigning groups of bits.

NOTE: Labels can have a maximum of 32 channels assigned to them, however, you cannot assign any single output channel to more than one label.

Bits assigned to a label are numbered from right to left. The least significant assigned bit on the far right is numbered 0. The next assigned bit to the left is numbered 1, and so on. Labels can contain bits that are not consecutive; however, bits are always numbered consecutively within a label.

Label Polarity

The pattern generator can display each label's data in both positive and negative logic. The default polarity for all labels is positive. To toggle the polarity, select the label's polarity field under the *Format* tab.

NOTE: Toggling the polarity of a label does not toggle the physical output signal. The polarity of a label is a display orientation concept, useful only when entering data values.

Finding a label

Access the *Find Label* feature under *Edit* in the menu bar of all tabs. The *Find Label* feature locates specified labels, then displays them.

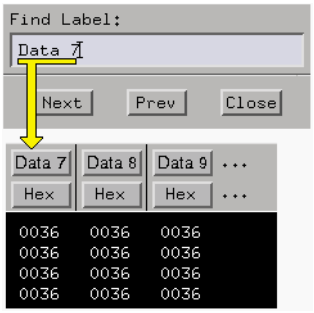
The search functionality is similar to a text based keyword search. After you enter the label name and select Next or Prev, the list of labels is rolled, exposing the label whose text characters, left to right, match the label name entered.

You can search for a complete label name as shown below in example 1, or enter just the base name and use the *Next* or *Prev* fields to roll through the list as in example 2.

Example 1

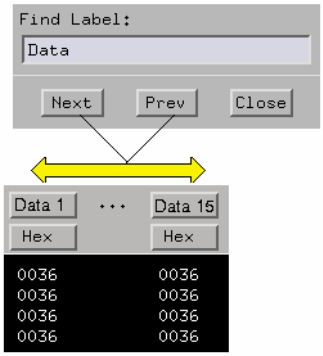
You have a list of labels named *Data1* through *Data15*. Enter the complete name *Data7* in the text entry field, then select *Next*. The

label named *Data7* appears on the left-most side of the displayed labels.



Example 2

You have a list of labels named *Data1* through *Data15*. Enter only the base name *Data* in the text entry field, then use the *Next* or *Prev* fields to roll the list of labels.

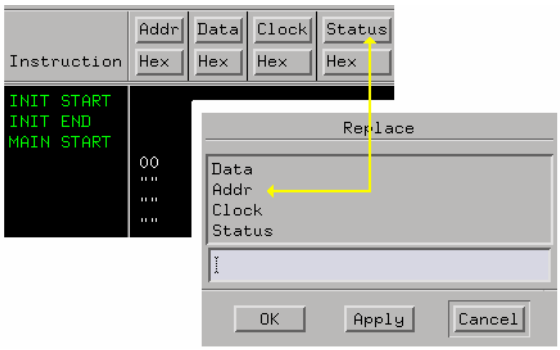


Replace Labels

The *Replace Label* command is available in the *Sequence* and *Macro* tabs.

NOTE: To highlight a line, drag from the bottom of the desired line to the top of the desired line.

1. Highlight the label to replace.
2. Select *Replace*.
3. From the Select Label dialog that appears, select one or more new labels to replace the existing label.
4. Select *OK* if you are done, or *Apply* if you have more labels to replace.



See Also "Searching for Labels" on page 93

Appending Labels

The *Append Label* command is located under *Edit* in the Sequence and Macro menu bar. The *Append Label* command adds labels to the right of the list of labels in Sequence and Macro.

1. In Sequence or Macro, select *Edit*, then select *Append Label*.
2. From the Select Label dialog that appears, select the new labels to append.
3. Select *OK* if you are done, or *Apply* if you have more labels to append.

See Also "Searching for Labels" on page 93

Insert All Labels

The *Insert All Labels* command is located under *Edit* in the Sequence and Macro menu bar. Select this command to insert all valid labels under the Format tab into the sequence.

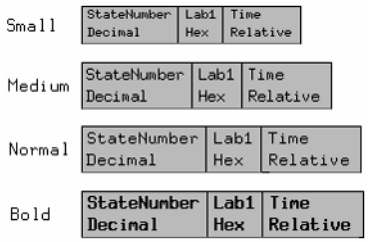
Delete All Labels

The *Delete All Labels* command is located under *Edit* in the Sequence and Macro menu bar. Select this command to delete all assigned labels in the sequence.

Setting the Label Font Size

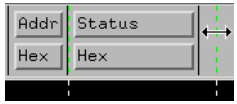
Font settings are global. The font size you select is applied to all text in the display area.

1. From the menu bar select *Options*, then select *Font*.
2. Select a font size from the list.



Example of Font Sizes

Adjusting Column Width

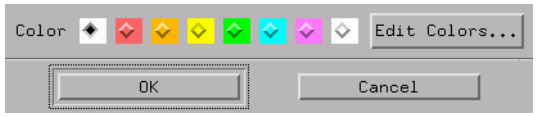


1. Select the right edge of the label box.
2. Drag the box edge to the desired width.

Setting Column Color

NOTE: To highlight a line, drag from the bottom of the desired line to the top of the desired line.

1. Select the *Sequence* or *Macro* tab.
2. Highlight a label, then select *Change Color*.
3. Select the desired color.
4. Select *OK*.

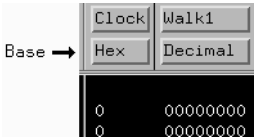


Setting Default Label Color

The *Default Label Color* command lets you set the label color for all new labels inserted into either Sequence or Macro.

1. Select the *Sequence* or *Macro* tab.
2. From the menu bar, select *Options*, then select *Default Label Color*.
3. From the Default Color dialog that appears, select the desired color.
4. Select *OK*.

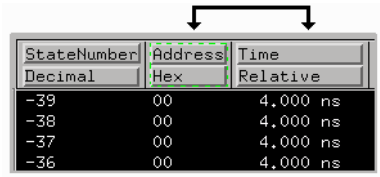
Setting the Numeric Base



1. Select the label's base field.
2. Select the desired base type.

Rearranging the Label Order

1. Select the desired label.
2. Drag the label to its new location.



Working with Macro Parameters

Parameters are used to pass values into macros. A major benefit in passing parameters is that you keep a macro's functionality generic and still direct specific action identified by parameters. Think of a parameter as the only part of a macro that changes as the macro is reused.

Use this procedure after creating your macro or recalling it in the *Macro* display:

1. From *Macro*, turn the desired parameters on. (see page 101)
2. Insert the parameters. (see page 102)
3. Assign values to the parameters. (see page 102)

NOTE: Macro parameters are passed as 32-bit values. If fewer than 32 bits are assigned to a label in *Format*, the most significant bits of the parameter value are truncated when the parameter is used as data for the given label.

See Also

"Building a User Macro" on page 25

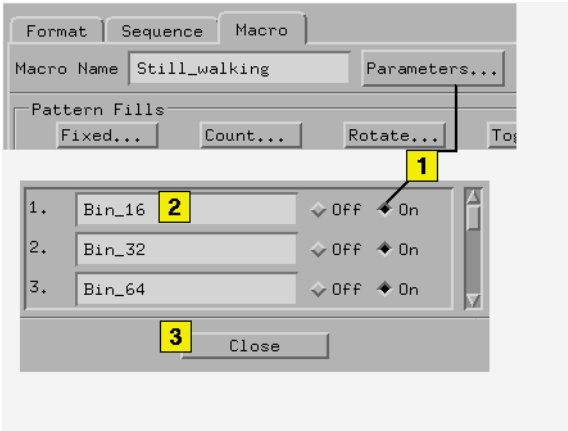
"Recalling Macros" on page 117

"Removing Parameters from a Macro" on page 103

Turning Parameters On

You have 10 available parameters per macro. You cannot use parameters until you turn them on.

1. From *Macro*, select *Parameters...*, then select *On*.
2. Optional - If you want a custom parameter name, select anywhere in the name field and enter a name.
3. Select *Close*.



Inserting Parameters into a Macro

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.

The following procedure is performed after your macro is created, recalled into the Macro display, and the parameter is turned on. (see page 101)

1. From *Macro*, select the data field you want to replace. This positions the cursor.
2. Select over any vector line and select *Set Parameter*.
3. From the Set Parameter dialog, select the parameter you want to set.
4. Select *OK*.

See Also

"Working with Macro Parameters" on page 101

Assigning Parameter Values

Parameter values are easily set or changed from the tab in which the

macro instruction was inserted. Assigned parameter values are displayed in parentheses next to their corresponding macro instruction. Parameter values are always displayed as hexadecimal values.

1. From the *Sequence* tab (or the *Macro* tab, if the macro is nested), select the macro instruction.
2. From the Set Parameters dialog that appears, enter in the new parameter values.
3. Select *OK*.

NOTE: Macro parameters are passed as 32-bit values. If fewer than 32 bits are assigned to a label in Format, the most significant bits of the parameter value are truncated when the parameter is used as data for the given label.

Removing Parameters from a Macro

1. From *Macro*, select the *data field* that contains the parameter to be removed.
2. Highlight over any vector line and select *Clear Parameter*.

Working with Automatic Pattern Fills

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

Pattern fills can either be inserted below the selected sequence line (flashing cursor), or configured to overwrite existing vectors starting at the selected sequence line.

- "Generating a Fixed Pattern Fill" on page 104
- "Generating a Count Pattern Fill" on page 105
- "Generating a Rotate Pattern Fill" on page 106
- "Generating a Toggle Pattern Fill" on page 108
- "Generating a Random Pattern Fill" on page 109

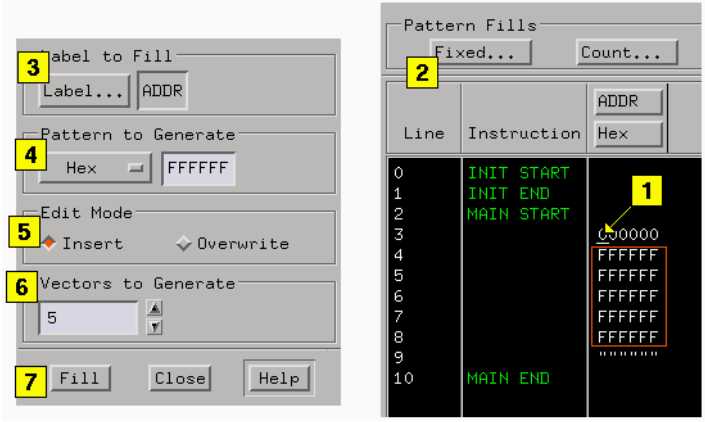
Generating a Fixed Pattern Fill

1. Position the cursor for one of the following operations.
   - To *insert* lines, select the vector line directly above where you want to insert the new lines.
   - To *overwrite* lines, select the vector line where you want the overwrite to begin.
2. From either *Sequence* or *Macro*, select *Fixed...*.
3. From the Fixed Pattern Fill dialog, select *Label...*, then select the desired label to fill.
4. Assign a pattern by filling in the pattern entry field and selecting a numeric base.
5. Select the Edit Mode.
   - Insert - new vector lines are inserted after the flashing cursor.

- Overwrite - existing vector lines are overwritten, beginning with the line containing the flashing cursor.

6. Select the number of pattern fill vectors to generate.
7. Select *Fill*.

NOTE: Select the pattern entry field to see a shortcut menu for assigning common patterns.

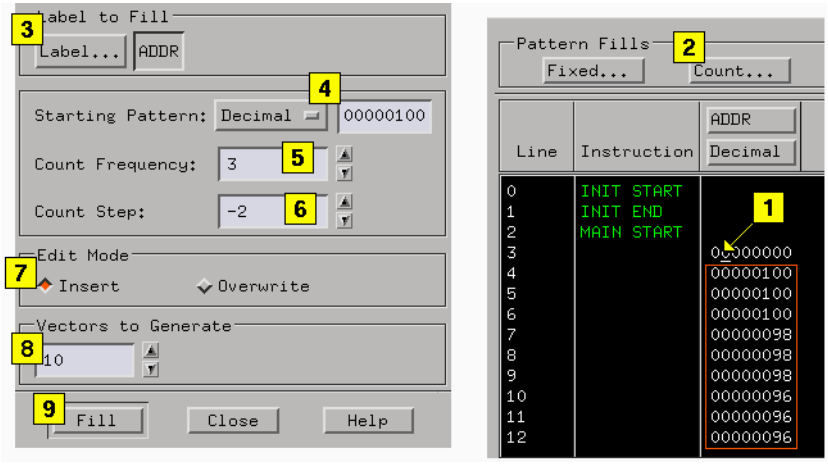


Generating a Count Pattern Fill

1. Position the cursor for one of the following operations.
   - To *insert* lines, select the vector line directly above where you want to insert the new lines.
   - To *overwrite* lines, select the vector line where you want the overwrite to begin.
2. From either *Sequence* or *Macro*, select *Count...*.
3. From the Count Pattern Fill dialog, select *Label...*, then select the desired label to fill.
4. Assign a starting pattern by filling in the pattern entry field and selecting a numeric base.

5. Select the Count Frequency. The count frequency specifies how often the counting pattern changes. For example, if count frequency is set to 2, the counting pattern increments after every 2 vectors (that is, 1,1,2,2,3,3,...).
6. Select the Count Step. The count step specifies whether the pattern counts up or down (by using a positive or negative value) and the unit of step. For example, with count step set to -2, the pattern counts downward, skipping every other value (that is, 100, 98,96,...).
7. Select the Edit Mode.
   - Insert - new vector lines are inserted after the flashing cursor.
   - Overwrite - existing vector lines are overwritten, beginning with the line containing the flashing cursor.
8. Select the number of pattern fill vectors to generate.
9. Select *Fill*.

NOTE: Select the pattern entry field to see a shortcut menu for assigning common patterns.



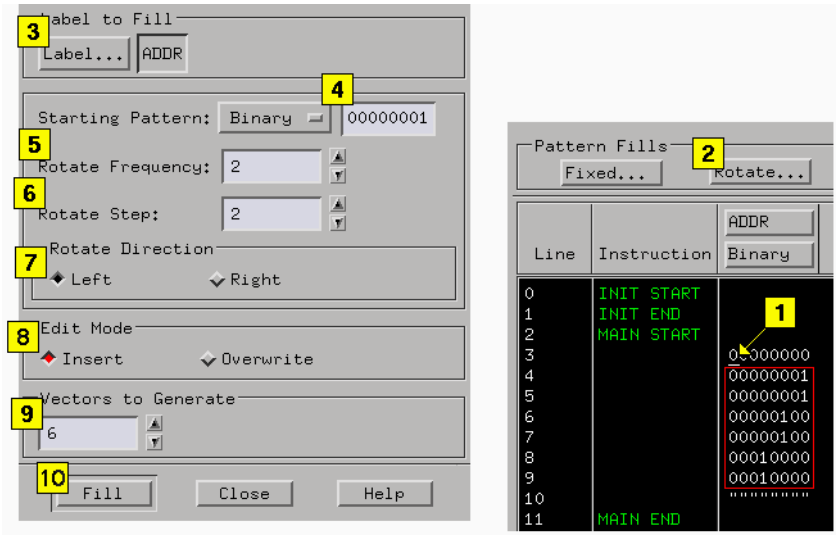
Generating a Rotate Pattern Fill

1. Position the cursor for one of the following operations.

- To *insert* lines, select the vector line directly above where you want to insert the new lines.
- To *overwrite* lines, select the vector line where you want the overwrite to begin.

2. From either *Sequence* or *Macro*, select *Rotate...*.
3. From the Rotate Pattern Fill dialog, select *Label...*, then select the desired label to fill.
4. Assign a starting pattern by filling in the pattern entry field and selecting a numeric base.
5. Select the Rotate Frequency. The rotate frequency specifies how often the rotating pattern changes. For example, if rotate frequency is set to 2, the pattern rotates after every 2 vectors (that is, 0001, 0001, 0010, 0010, 0100, 0100,...).
6. Select the Rotate Step. The rotate step specifies how many units the pattern rotates. For example, with rotate step set to 2, the pattern rotates 2 units (that is, 00001, 00100, 10000,...).
7. Select the Rotate Direction.
   - Left - rotates pattern to the left.
   - Right - rotates pattern to the right.
8. Select the Edit Mode.
   - Insert - new vector lines are inserted after the flashing cursor.
   - Overwrite - existing vector lines are overwritten, beginning with the line containing the flashing cursor.
9. Select the number of pattern fill vectors to generate.
10. Select *Fill*.

NOTE: Select the pattern entry field to see a shortcut menu for assigning common patterns.



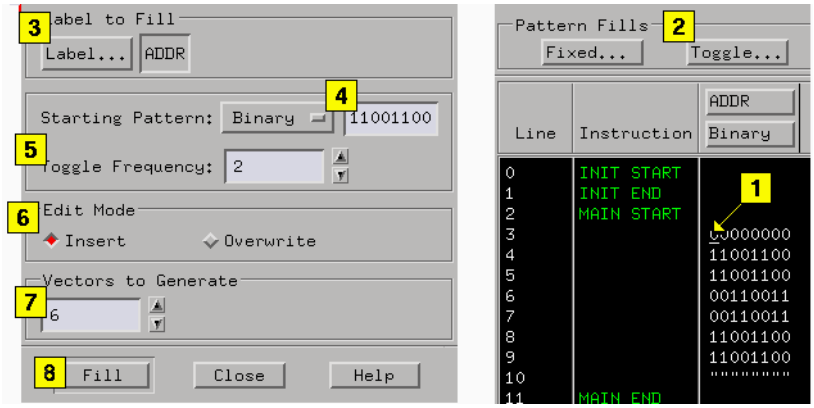
Generating a Toggle Pattern Fill

1. Position the cursor for one of the following operations.
   - To *insert* lines, select the vector line directly above where you want to insert the new lines.
   - To *overwrite* lines, select the vector line where you want the overwrite to begin.
2. From either *Sequence* or *Macro*, select *Toggle...*.
3. From the Toggle Pattern Fill dialog, select *Label...*, then select the desired label to fill.
4. Assign a Starting Pattern by filling in the pattern entry field and selecting a numeric base.
5. Select the Toggle Frequency. The toggle frequency specifies how often the pattern changes. For example, if toggle frequency is set to 2, the pattern toggles after every 2 vectors (that is, 0011, 0011, 1100, 1100, 0011, 0011,...).
6. Select the Edit Mode.

- Insert - new vector lines are inserted after the flashing cursor.
- Overwrite - existing vector lines are overwritten, beginning with the line containing the flashing cursor.

7. Select the number of pattern fill vectors to generate.
8. Select *Fill*.

NOTE: Select the pattern entry field to see a shortcut menu for assigning common patterns.



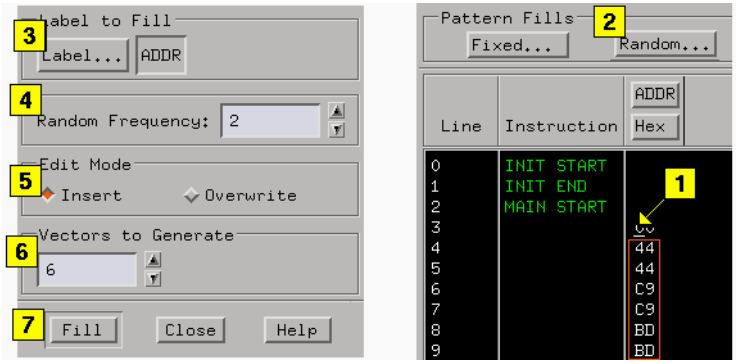
Generating a Random Pattern Fill

1. Position the cursor for one of the following operations.
   - To *insert* lines, select the vector line directly above where you want to insert the new lines.
   - To *overwrite* lines, select the vector line where you want the overwrite to begin.
2. From either *Sequence* or *Macro*, select *Random...*.
3. From the Random Pattern Fill dialog, select *Label...*, then select the desired label to fill.
4. Select the Random Frequency. The Random frequency specifies how often

the random pattern changes. For example, if random frequency is set to 2, the pattern changes after every 2 vectors (that is, 86, 86, 74, 74, F3, F3,...).

5. Select the Edit Mode.
   - Insert - new vector lines are inserted after the flashing cursor.
   - Overwrite - existing vector lines are overwritten, beginning with the line containing the flashing cursor.
6. Select the number of pattern fill vectors to generate.
7. Select *Fill*.

NOTE: Select the pattern entry field to see a shortcut menu for assigning common patterns.



Printing the Pattern Generator Window

The *Print This Window* operation lets you print just the pattern generator window. Use this operation if you want a hardcopy or electronic record of configurations and data currently displayed in the viewing area of the pattern generator window.

NOTE: Only the currently displayed viewing area of the pattern generator window is printed. If any data or configuration fields appear offscreen, scroll the desired data or configuration fields into the viewing area before printing.

1. Optional - configure the Print Options (see the *Agilent Technologies 16700A/B-Series Logic Analysis System* help volume) if desired. Print Options include print destination, file format type, filename autoincrement, and color/b&w; pixel mapping.
2. In the Pattern Generator tool menu bar, select *File*, then select *Print This Window*. The print output will be as configured in the Print Options in step 1.

See Also

Setting Print Options (see the *Agilent Technologies 16700A/B-Series Logic Analysis System* help volume)

Set Up the Printer (see the *Agilent Technologies 16700A/B-Series Logic Analysis System* help volume)

Printing Vector Sequences to a File

Printing sequences to a file is useful when you want a filecopy or hardcopy of your programs for documentation purposes. The procedure is the same for sequences and macros.

1. From the menu bar, select *File*.
2. Select *Print Sequence To File* or *Print Macro To File*. The Print to File dialog appears.
3. Select which label to print. If you want to print all of the labels, select *All*. If you want to print selected labels, choose *Selected*, then select the desired labels.
4. Select either *All* or *Partial* as the range of sequence lines to print. If you select Partial, specify the range of line numbers.
5. Select the path and filename to print the sequence to. Use either the *File...* dialog to build a path and filename, or enter the path and filename into the text entry field.
6. Optional - enter a file description in the text entry area. File descriptions are printed at the top of the output file.
7. Select *Save*.

Viewing Saved Vector Sequence Files

The following procedure shows you how to connect to the logic analysis system through *File Transfer Protocol* (FTP), then execute the FTP "Get" command to place files into your home directory for viewing.

NOTE: The logic analysis system stores the vector sequence files in binary format. Use the FTP binary mode to preserve the data stored in these files.

1. From the command line of the remote machine, enter *ftp "machine_name"*, where *"machine_name"* can be either the IP address, or the hostname with an assigned alias for the logic analysis system.
2. For the User Name: enter *"anonymous"*.

3. For the Password: enter "*anonymous*".
4. From the FTP prompt, enter "*get filename*".

At this point, the sequence file should be saved in your home directory, and be ready for you to open and view it.

Example of Saved Sequence

```
                                DATA        ADDR
Line        Instruction         Binary      Hex
______      ___________         ________    ____

0           INIT START
1           INIT END
2           MAIN START
3                               00000000    00
4                               00000001    01
5                               00000010    02
6                               00000100    03
7                               00001000    04
8                               00010000    05
9                               00100000    06
10                              01000000    07
11                              10000000    08
12                              00000000    00
13          MAIN END
```

Key Characteristics

Output Channnels:

24 channels at 300 MHz clock; 48 channels at 180 MHz clock.

Memory Depth:

16,777,216 of vectors.

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 Different Macros:

100

Maximum Number of Lines in a Macro:

4096

Maximum Number of "Wait" Event Patterns:

4

Automatic Cursor Wrap

The Cursor Wrap mode specifies how the cursor should line wrap when the cursor comes to the end of a *data field* during data entry.

The automatic cursor wrap has two modes:

- Line (default) - enters all values for all labels within a single line before proceeding to the next line below.
- Column - enters all values for a single label only.

Setting the Cursor Wrap

1. In *Sequence* or *Macro*, select *Options*, then select Cursor Wrap.
2. Select either Line or Column.

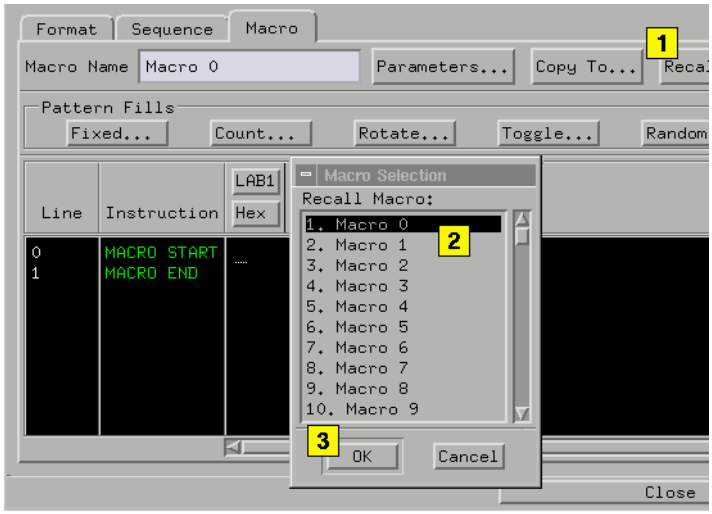
NOTE: If you are entering many data vectors, it is quicker to insert all blank data vectors first, and then enter the data while letting the cursor wrap feature guide you.

Recalling Macros

The pattern generator contains 100 macros. Each macro is initially empty at power-up. To create a new macro or modify a particular macro, you must first recall that macro in the *Macro* display for editing.

To Recall a User Macro

1. Select the *Macro* tab.
2. Select *Recall*.
3. From the Macro Selection dialog that appears, select the desired macro name.
4. Select *OK*.



See Also

"Working with Macro Parameters" on page 101

"Working with Instruction Types" on page 81

"Editing Sequences" on page 76

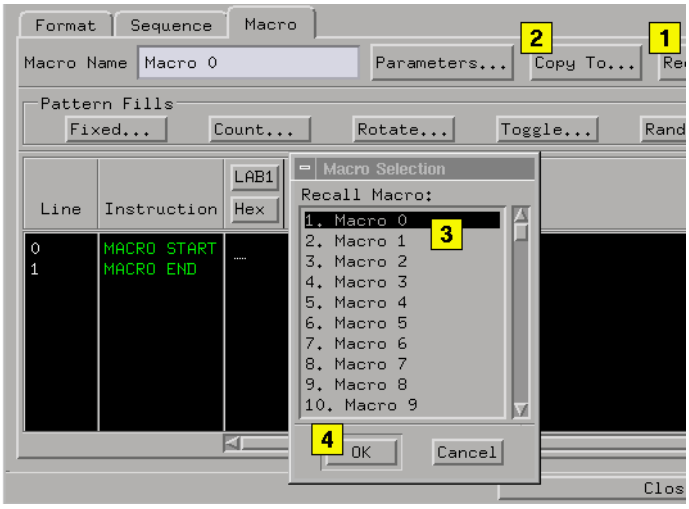
"Working with Labels and Pods" on page 88

Copying Macros

You can copy the currently displayed macro in the Macro display to any of the other available macros. Use this feature to create a new, but slightly different macro from the original macro.

To Copy a User Macro

1. From Macro, recall (see page 117) the macro you want to copy.
2. Select *Copy To*.
3. From the Macro Selection dialog that appears, select the desired macro name you want to copy to.
4. Select *OK*.



See Also

"Working with Macro Parameters" on page 101

"Working with Instruction Types" on page 81

"Editing Sequences" on page 76

"Working with Labels and Pods" on page 88

Viewing a Compiled Sequence

When a display tool is connected to the Pattern Generator icon, it displays the *compiled* sequence, rather than the actual sequence output at run time.

The compiled view of the sequence can be helpful when trying to understand the behavior of software instructions. The compiled view is not helpful when trying to understand the behavior of hardware instructions, since their behavior changes from run to run.

For more information on hardware and software instruction types, see Working with Instruction Types. (see page 81)

absolute Denotes the time period or count of states between a captured state and the trigger state. An absolute count of -10 indicates the state was captured ten states before the trigger state was captured.

acquisition Denotes one complete cycle of data gathering by a measurement module. For example, if you are using an analyzer with 128K memory depth, one complete acquisition will capture and store 128K states in acquisition memory.

analysis probe A probe connected to a microprocessor or standard bus in the device under test. An analysis probe provides an interface between the signals of the microprocessor or standard bus and the inputs of the logic analyzer. Also called a *preprocessor*.

analyzer 1 In a logic analyzer with two *machines*, refers to the machine that is on by default. The default name is *Analyzer<N>*, where N is the slot letter.

analyzer 2 In a logic analyzer with two *machines*, refers to the machine that is off by default. The default name is *Analyzer<N2>*, where N is the slot letter.

arming An instrument tool must be armed before it can search for its trigger condition. Typically, instruments are armed immediately when *Run* or *Group Run* is selected. You can set up one instrument to arm another using the *Intermodule Window*. In these setups, the second instrument cannot search for its trigger condition until it receives the arming signal from the first instrument. In some analyzer instruments, you can set up one analyzer *machine* to arm the other analyzer machine in the *Trigger Window*.

asterisk (*) See *edge terms*, *glitch*, and *labels*.

bits Bits represent the physical logic analyzer channels. A bit is a *channel* that has or can be assigned to a *label*. A bit is also a position in a label.

card This refers to a single instrument intended for use in the Agilent Technologies 16700A/B-series mainframes. One card fills one slot in the mainframe. A module may comprise a single card or multiple cards cabled together.

channel The entire signal path from the probe tip, through the cable and module, up to the label grouping.

click When using a mouse as the

pointing device, to click an item, position the cursor over the item. Then quickly press and release the *left mouse button*.

clock channel A logic analyzer *channel* that can be used to carry the clock signal. When it is not needed for clock signals, it can be used as a *data channel*, except in the Agilent Technologies 16517A.

context record A context record is a small segment of analyzer memory that stores an event of interest along with the states that immediately preceded it and the states that immediately followed it.

context store If your analyzer can perform context store measurements, you will see a button labeled *Context Store* under the Trigger tab. Typical context store measurements are used to capture writes to a variable or calls to a subroutine, along with the activity preceding and following the events. A context store measurement divides analyzer memory into a series of context records. If you have a 64K analyzer memory and select a 16-state context, the analyzer memory is divided into 4K 16-state context records. If you have a 64K analyzer memory and select a 64-state context, the analyzer memory will be divided into 1K 64-state records.

count The count function records periods of time or numbers of state transactions between states stored in memory. You can set up the analyzer count function to count occurrences of a selected event during the trace, such as counting how many times a variable is read between each of the writes to the variable. The analyzer can also be set up to count elapsed time, such as counting the time spent executing within a particular function during a run of your target program.

cross triggering Using intermodule capabilities to have measurement modules trigger each other. For example, you can have an external instrument arm a logic analyzer, which subsequently triggers an oscilloscope when it finds the trigger state.

data channel A *channel* that carries data. Data channels cannot be used to clock logic analyzers.

data field A data field in the pattern generator is the data value associated with a single label within a particular data vector.

data set A data set is made up of all labels and data stored in memory of any single analyzer machine or

instrument tool. Multiple data sets can be displayed together when sourced into a single display tool. The Filter tool is used to pass on partial data sets to analysis or display tools.

debug mode See *monitor*.

delay The delay function sets the horizontal position of the waveform on the screen for the oscilloscope and timing analyzer. Delay time is measured from the trigger point in seconds or states.

demo mode An emulation control session which is not connected to a real target system. All windows can be viewed, but the data displayed is simulated. To start demo mode, select *Start User Session* from the Emulation Control Interface and enter the demo name in the *Processor Probe LAN Name* field. Select the *Help* button in the *Start User Session* window for details.

deskewing To cancel or nullify the effects of differences between two different internal delay paths for a signal. Deskewing is normally done by routing a single test signal to the inputs of two different modules, then adjusting the Intermodule Skew so that both modules recognize the signal at the same time.

device under test The system under test, which contains the circuitry you are probing. Also known as a *target system*.

don't care For *terms*, a "don't care" means that the state of the signal (high or low) is not relevant to the measurement. The analyzer ignores the state of this signal when determining whether a match occurs on an input label. "Don't care" signals are still sampled and their values can be displayed with the rest of the data. Don't cares are represented by the *X* character in numeric values and the dot (.) in timing edge specifications.

dot (.) See *edge terms*, *glitch*, *labels*, and *don't care*.

double-click When using a mouse as the pointing device, to double-click an item, position the cursor over the item, and then quickly press and release the *left mouse button* twice.

drag and drop Using a Mouse: Position the cursor over the item, and then press and hold the *left mouse button*. While holding the left mouse button down, move the mouse to drag the item to a new location. When the item is positioned where you want it, release the mouse button.

Using the Touchscreen:
Position your finger over the item, then press and hold finger to the screen. While holding the finger down, slide the finger along the screen dragging the item to a new location. When the item is positioned where you want it, release your finger.

edge mode In an oscilloscope, this is the trigger mode that causes a trigger based on a single channel edge, either rising or falling.

edge terms Logic analyzer trigger resources that allow detection of transitions on a signal. An edge term can be set to detect a rising edge, falling edge, or either edge. Some logic analyzers can also detect no edge or a *glitch* on an input signal. Edges are specified by selecting arrows. The dot (.) ignores the bit. The asterisk (*) specifies a glitch on the bit.

emulation module A module within the logic analysis system mainframe that provides an emulation connection to the debug port of a microprocessor. An E5901A emulation module is used with a target interface module (TIM) or an analysis probe. An E5901B emulation module is used with an E5900A emulation probe.

emulation probe The stand-alone equivalent of an *emulation module*. Most of the tasks which can be performed using an emulation module can also be performed using an emulation probe connected to your logic analysis system via a LAN.

emulator An *emulation module* or an *emulation probe*.

Ethernet address See *link-level address*.

events Events are the things you are looking for in your target system. In the logic analyzer interface, they take a single line. Examples of events are *Label1 = XX* and *Timer 1 > 400 ns*.

filter expression The filter expression is the logical *OR* combination of all of the filter terms. States in your data that match the filter expression can be filtered out or passed through the Pattern Filter.

filter term A variable that you define in order to specify which states to filter out or pass through. Filter terms are logically OR'ed together to create the filter expression.

Format The selections under the logic analyzer *Format* tab tell the

logic analyzer what data you want to collect, such as which channels represent buses (labels) and what logic threshold your signals use.

frame The Agilent Technologies or 16700A/B-series logic analysis system mainframe. See also *logic analysis system*.

gateway address An IP address entered in integer dot notation. The default gateway address is 0.0.0.0, which allows all connections on the local network or subnet. If connections are to be made across networks or subnets, this address must be set to the address of the gateway machine.

glitch A glitch occurs when two or more transitions cross the logic threshold between consecutive timing analyzer samples. You can specify glitch detection by choosing the asterisk (*) for *edge terms* under the timing analyzer Trigger tab.

grouped event A grouped event is a list of *events* that you have grouped, and optionally named. It can be reused in other trigger sequence levels. Only available in Agilent Technologies 16715A or higher logic analyzers.

held value A value that is held until the next sample. A held value can exist in multiple data sets.

immediate mode In an oscilloscope, the trigger mode that does not require a specific trigger condition such as an edge or a pattern. Use immediate mode when the oscilloscope is armed by another instrument.

interconnect cable Short name for *module/probe interconnect cable*.

intermodule bus The intermodule bus (IMB) is a bus in the frame that allows the measurement modules to communicate with each other. Using the IMB, you can set up one instrument to *arm* another. Data acquired by instruments using the IMB is time-correlated.

intermodule Intermodule is a term used when multiple instrument tools are connected together for the purpose of one instrument arming another. In such a configuration, an arming tree is developed and the group run function is designated to start all instrument tools. Multiple instrument configurations are done in the Intermodule window.

internet address Also called Internet Protocol address or IP address. A 32-bit network address. It

is usually represented as decimal numbers separated by periods; for example, 192.35.12.6. Ask your LAN administrator if you need an internet address.

labels Labels are used to group and identify logic analyzer channels. A label consists of a name and an associated bit or group of bits. Labels are created in the Format tab.

line numbers A line number (Line #s) is a special use of *symbols*. Line numbers represent lines in your source file, typically lines that have no unique symbols defined to represent them.

link-level address Also referred to as the Ethernet address, this is the unique address of the LAN interface. This value is set at the factory and cannot be changed. The link-level address of a particular piece of equipment is often printed on a label above the LAN connector. An example of a link-level address in hexadecimal: 0800090012AB.

local session A local session is when you run the logic analysis system using the local display connected to the product hardware.

logic analysis system The Agilent Technologies 16700A/B-series mainframes, and all tools designed to work with it. Usually used to mean the specific system and tools you are working with right now.

machine Some logic analyzers allow you to set up two measurements at the same time. Each measurement is handled by a different machine. This is represented in the Workspace window by two icons, differentiated by a *1* and a *2* in the upper right-hand corner of the icon. Logic analyzer resources such as pods and trigger terms cannot be shared by the machines.

markers Markers are the green and yellow lines in the display that are labeled *x*, *o*, *G1*, and *G2*. Use them to measure time intervals or sample intervals. Markers are assigned to patterns in order to find patterns or track sequences of states in the data. The x and o markers are local to the immediate display, while G1 and G2 are global between time correlated displays.

master card In a module, the master card controls the data acquisition or output. The logic analysis system references the module by the slot in which the master card is plugged. For example, a 5-card Agilent Technologies 16555D would be referred to as *Slot C:*

machine because the master card is in slot C of the mainframe. The other cards of the module are called *expansion cards*.

menu bar The menu bar is located at the top of all windows. Use it to select *File* operations, tool or system *Options*, and tool or system level *Help*.

message bar The message bar displays mouse button functions for the window area or field directly beneath the mouse cursor. Use the mouse and message bar together to prompt yourself to functions and shortcuts.

module/probe interconnect cable

The module/probe interconnect cable connects an E5901B emulation module to an E5900B emulation probe. It provides power and a serial connection. A LAN connection is also required to use the emulation probe.

module An instrument that uses a single timebase in its operation. Modules can have from one to five cards functioning as a single instrument. When a module has more than one card, system window will show the instrument icon in the slot of the *master card*.

monitor When using the Emulation Control Interface, running the monitor means the processor is in debug mode (that is, executing the debug exception) instead of executing the user program.

panning The action of moving the waveform along the timebase by varying the delay value in the Delay field. This action allows you to control the portion of acquisition memory that will be displayed on the screen.

pattern mode In an oscilloscope, the trigger mode that allows you to set the oscilloscope to trigger on a specified combination of input signal levels.

pattern terms Logic analyzer resources that represent single states to be found on labeled sets of bits; for example, an address on the address bus or a status on the status lines.

period (.) See *edge terms*, *glitch*, *labels*, and *don't care*.

pod pair A group of two pods containing 16 channels each, used to physically connect data and clock signals from the unit under test to the analyzer. Pods are assigned by pairs in the analyzer interface. The number of pod pairs available is determined

by the channel width of the instrument.

pod See *pod pair*

point To point to an item, move the mouse cursor over the item, or position your finger over the item.

preprocessor See *analysis probe*.

primary branch The primary branch is indicated in the *Trigger sequence step* dialog box as either the *Then find* or *Trigger on* selection. The destination of the primary branch is always the next state in the sequence, except for the Agilent Technologies 16517A. The primary branch has an optional occurrence count field that can be used to count a number of occurrences of the branch condition. See also *secondary branch*.

probe A device to connect the various instruments of the logic analysis system to the target system. There are many types of probes and the one you should use depends on the instrument and your data requirements. As a verb, "to probe" means to attach a probe to the target system.

processor probe See *emulation probe*.

range terms Logic analyzer resources that represent ranges of values to be found on labeled sets of bits. For example, range terms could identify a range of addresses to be found on the address bus or a range of data values to be found on the data bus. In the trigger sequence, range terms are considered to be true when any value within the range occurs.

relative Denotes time period or count of states between the current state and the previous state.

remote display A remote display is a display other than the one connected to the product hardware. Remote displays must be identified to the network through an address location.

remote session A remote session is when you run the logic analyzer using a display that is located away from the product hardware.

right-click When using a mouse for a pointing device, to right-click an item, position the cursor over the item, and then quickly press and release the *right mouse button*.

sample A data sample is a portion of a *data set*, sometimes just one point. When an instrument samples the target system, it is taking a single

measurement as part of its data acquisition cycle.

Sampling Use the selections under the logic analyzer Sampling tab to tell the logic analyzer how you want to make measurements, such as State vs. Timing.

secondary branch The secondary branch is indicated in the *Trigger sequence step* dialog box as the *Else on* selection. The destination of the secondary branch can be specified as any other active sequence state. See also *primary branch*.

session A session begins when you start a *local session* or *remote session* from the session manager, and ends when you select *Exit* from the main window. Exiting a session returns all tools to their initial configurations.

skew Skew is the difference in channel delays between measurement channels. Typically, skew between modules is caused by differences in designs of measurement channels, and differences in characteristics of the electronic components within those channels. You should adjust measurement modules to eliminate as much skew as possible so that it does not affect the accuracy of your measurements.

state measurement In a state measurement, the logic analyzer is clocked by a signal from the system under test. Each time the clock signal becomes valid, the analyzer samples data from the system under test. Since the analyzer is clocked by the system, state measurements are *synchronous* with the test system.

store qualification Store qualification is only available in a *state measurement*, not *timing measurements*. Store qualification allows you to specify the type of information (all samples, no samples, or selected states) to be stored in memory. Use store qualification to prevent memory from being filled with unwanted activity such as no-ops or wait-loops. To set up store qualification, use the *While storing* field in a logic analyzer trigger sequence dialog.

subnet mask A subnet mask blocks out part of an IP address so that the networking software can determine whether the destination host is on a local or remote network. It is usually represented as decimal numbers separated by periods; for example, 255.255.255.0. Ask your LAN administrator if you need a the subnet mask for your network.

symbols Symbols represent patterns and ranges of values found on labeled sets of bits. Two kinds of symbols are available:

- Object file symbols - Symbols from your source code, and symbols generated by your compiler. Object file symbols may represent global variables, functions, labels, and source line numbers.
- User-defined symbols - Symbols you create.

Symbols can be used as *pattern* and *range* terms for:

- Searches in the listing display.
- Triggering in logic analyzers and in the source correlation trigger setup.
- Qualifying data in the filter tool and system performance analysis tool set.

system administrator The system administrator is a person who manages your system, taking care of such tasks as adding peripheral devices, adding new users, and doing system backup. In general, the system administrator is the person you go to with questions about implementing your software.

target system The system under test, which contains the microprocessor you are probing.

terms Terms are variables that can be used in trigger sequences. A term can be a single value on a label or set of labels, any value within a range of values on a label or set of labels, or a glitch or edge transition on bits within a label or set of labels.

TIM A TIM (Target Interface Module) makes connections between the cable from the emulation module or emulation probe and the cable to the debug port on the system under test.

time-correlated Time correlated measurements are measurements involving more than one instrument in which all instruments have a common time or trigger reference.

timer terms Logic analyzer resources that are used to measure the time the trigger sequence remains within one sequence step, or a set of sequence steps. Timers can be used to detect when a condition lasts too long or not long enough. They can be used to measure pulse duration, or duration of a wait loop. A single timer term can be used to delay trigger until a period of time after detection of a significant event.

timing measurement In a timing measurement, the logic analyzer samples data at regular intervals according to a clock signal internal to the timing analyzer. Since the analyzer is clocked by a signal that is not related to the system under test, timing measurements capture traces of electrical activity over time. These measurements are *asynchronous* with the test system.

tool icon Tool icons that appear in the workspace are representations of the hardware and software tools selected from the toolbox. If they are placed directly over a current measurement, the tools automatically connect to that measurement. If they are placed on an open area of the main window, you must connect them to a measurement using the mouse.

toolbox The Toolbox is located on the left side of the main window. It is used to display the available hardware and software tools. As you add new tools to your system, their icons will appear in the Toolbox.

tools A tool is a stand-alone piece of functionality. A tool can be an instrument that acquires data, a display for viewing data, or a post-processing analysis helper. Tools are represented as icons in the main window of the interface.

trace See *acquisition*.

trigger sequence A trigger sequence is a sequence of events that you specify. The logic analyzer compares this sequence with the samples it is collecting to determine when to *trigger*.

trigger specification A trigger specification is a set of conditions that must be true before the instrument triggers.

trigger Trigger is an event that occurs immediately after the instrument recognizes a match between the incoming data and the trigger specification. Once trigger occurs, the instrument completes its *acquisition*, including any store qualification that may be specified.

workspace The workspace is the large area under the message bar and to the right of the toolbox. The workspace is where you place the different instrument, display, and analysis tools. Once in the workspace, the tool icons graphically represent a complete picture of the measurements.

zooming In the oscilloscope or timing analyzer, to expand and contract the waveform along the time base by varying the value in the s/Div

field. This action allows you to select specific portions of a particular waveform in acquisition memory that will be displayed on the screen. You can view any portion of the waveform record in acquisition memory.

Index

Symbols

*, bit assignment, 94
., bit unassignment, 94

A

Agilent 16720A characteristics, 114
Agilent 16720A pattern generator, 2
Agilent 16720A specifications, 114
append labels command, 97
ASCDown command, 33
ASCII command, 32
ascii files, creating, 30
ascii files, loading, 28
assigning parameters, 102
automatic cursor wrap mode, 116

B

base, numeric base, 100
BEGIN, command, 56
beginnersexercise', 17
Binary Data, command, 56
binary files, creating, 40
binary files, loading, 39
Binary, commands, 42
bit assignment, 94
bit reference line, 94
bit significance, 94
bits, clearing, 94
bits, maximum per label, 94
bits, reordering, 92
blank lines, 78
break instruction, 82
BREAK, command, 51
breaks, 81

C

cables, connecting, 62
channel count, and modes, 12
channels, maximum per label, 94
characterisitic data inputs, 114
characterisitic duty cycle, 114
characterisitic logic levels, 114
characteristic memory depth, 114
characteristic number of channels, 114
clear format labels command, 93
clear lines, 76
clear pods command, 94
clk in, 13
clock characteristic, 114
clock out delay, 13
clock period, limited by, 13
clock pods, selecting, 62
clock ranges, setting, 13
clock source, external, 13
clock source, internal, 13
CLOCK, command, 43
color, labels, 99
column width, adjusting, 99
commands, ASCDown, 33
commands, BEGIN, 56
commands, Binary, 42
commands, Binary Data, 56
commands, BREAK, 51
commands, CLOCK, 43
commands, DELAY, 44
commands, DEPTH, 50
commands, EVENT, 52
commands, FORMat:CLOCK, 36
commands, FORMat:DELay, 37
commands, FORMat:MODe, 36
commands, LABEL, 46
commands, LABel, 33
commands, MAIN, 55
commands, MODE, 45
commands, ORDER, 48
commands, pattern generator, 32, 43
commands, RMODE, 46
commands, SIGNAL, 54
commands, VECTor, 34
commands, WAIT, 53
commands, WIDTH, 49
compiled vector sequences, viewing, 119
configuration files, loading, 61
configurations, storing, 61
copy, 118
count pattern fill, 105
cut lines, 76
cut lines vs delete lines, 76

D

data pods, selecting, 62
data sets, definition, 59
data sets, labels, 59
data sets, range, 60
data, deleting lines, 77
data, editing, 76
data, insert blank, 78
DELAY, command, 44
delete all labels command, 98
delete labels, 90
delete lines, 76, 77
delete lines vs cut lines, 77
deleting bit assignments, 94
deleting labels, 93
DEPTH, command, 50
ditto, inserting, 80

E

editing sequences, 76
erase lines, 76, 77
EVENT, command, 52
example, conceptual, 10
example, creating a sequence, 17
external clock, frequency, 12
external clock, programming, 36

F

file out tool, 58
files, importing, 28, 39
files, importing system data, 58
find labels feature, 95

finding labels, 93
fixed pattern fill, 104
fonts, setting the size, 98
format tab, 88, 91
format tab, labels, 89
FORMat:CLOCk command, 36
FORMat:DELay command, 37
FORMat:MODe command, 36
frequency, and clock source, 13

G

glossary of terms, 2
goto line, 79
group bit assignment, 94

H

hardcopy, of sequence, 112
hardware, instruction types, 81

I

IF conditions, maximum number, 114
if events, 81
import, 30, 40
import 16522A ASCII file, 28
import 16720 Binary file, 39
import system data file, 58
init sequence, creating, 21
initialization sequence, building, 21
insert after, 89, 91
insert all labels command, 98
insert before, 89, 91
insert ditto, 80
inserting blank lines, 78
inserting macro parameters, 102
instruction, signal IMB, 82
instructions, 81
instructions, break, 82
instructions, editing, 76
instructions, repeat loop, 85
instructions, user macro, 84
instructions, wait external event, 83
instructions, wait IMB event, 83
internal clock, frequency, 12
internal clock, programming, 36

L

label color, default, 99
LABel command, 33
LABEL, command, 46
LABEL, keywords, 48
labels, adjusting width, 99
labels, appending, 97
labels, assigning bits, 94
labels, clearing, 93
labels, creating, 89, 91
labels, deleting, 90
labels, deleting all, 98
labels, finding, 95
labels, font size, 98
labels, inserting, 89, 91
labels, inserting all, 98
labels, operations, 88
labels, polarity, 95
labels, rearranging order, 100
labels, rename, 91
labels, reordering bits, 92
labels, replacing, 96
labels, searching, 93
labels, setting color, 99
labels, turning off, 92
labels, turning on, 92
line, goto, 79
lines, blank, 78
lines, clearing, 76
lines, cutting, 76
lines, deleting, 76, 77
lines, editing, 76
lines, erasing, 76
lines, insert new, 78
lines, inserting blank, 78
lines, pasting, 76
loading ascii files, 28
loading binary files, 39
loops, 85

M

macros, 81, 118
macros, maximum number, 114
main sequence, creating, 23
MAIN, command, 55
MODE, command, 45

N

name parameters, 101
nested loops, 85
nested macros, 25
new lines, 78
note, maximum bits per label, 94
numeric base, setting, 100

O

ORDER, commands, 48
overview, 16720A pattern generator, 10

P

parameters, assigning values, 102
parameters, inserting, 102
parameters, macros, 25, 101
parameters, naming, 101
parameters, removing, 103
parameters, turning on, 101
paste lines, 76
pattern fills, count, 104, 105
pattern fills, fixed, 104
pattern fills, random, 104, 109
pattern fills, rotate, 104, 106
pattern fills, toggle, 104, 108
pattern fills, using, 104
pattern generator format tab, use, 11

Index

pattern generator interface, pods, 11
pattern generator pods, mapping, 11
pattern generator probes, mapping, 11
pattern generator window, hardcopy of, 111
pattern generator, connecting pods, 11
pattern generator, overview, 10
pattern generator, running, 16
pods, assigning bits, 94
pods, assinging bits, 11
pods, availability with mode, 12
pods, characteristics, 62
pods, clearing, 94
pods, connecting, 74
pods, equivalent circuits, 62
pods, mapping channel, 11
pods, operations, 88
pods, pinout, 62
pods, reordering bits, 92
pods, swapping, 93
polarity, labels, 95
printing pattern generator, 111
printing vector sequences, 112
probes, characteristics, 62
probes, connecting, 74

R

random pattern fill, 109
remove parameters, 103
removing labels, 93
rename labels command, 91
reorder bits feature, 92
repeat indicator, 34
repeat loop instruction, 85
repeat loops, 81
replace labels command, 96
RMODE, command, 46
roll, labels, 95
rotate patttern fill, 106

S

sequence, deleting lines, 77
sequence, importing, 39
sequence, initialization, 21
sequence, insert line, 78
sequence, main, 23
sequence, positioning, 79
sequences, editing, 76
sequences, importing, 28
sequences, importing data, 58
settings, saving, 61
shortcut, bit assignment, 94
signal IMB, 81
signal imb instruction, 82
SIGNAL, command, 54
software, instruction types, 81
swap pods command, 93
system help main page, 2

T

test vectors, building sequence, 15
toggle pattern fill, 108

U

user macro instruction, 84
user macros, 84, 118
user macros vs sequences, 25
user macros, creating, 25
user macros, parameters, 101
user macros, printing, 112
user macros, recalling, 117

V

values, parameters, 102
VECTor command, 34
vector output mode, 12
vector sequence, filling, 104
vector sequence, printing entire, 112
vectors, deleting, 77
vectors, editing, 76
vectors, generating, 104
vectors, inserting blank, 78

W

wait events, 81
Wait events, maximum number, 114
wait external event instruction, 83
wait IMB event instruction, 83
WAIT, command, 53
WIDTH, command, 49
wrap cursor, 116

Publication Number: 5988-9036EN
January 1, 2003

Agilent Technologies